Online Algorithms for Conversion Problems

An approach to conjoin
worst-case analysis and empirical-case analysis

von Esther Mohr
im März 2011

Informations- und
Technologiemanagement
Saarbrücken

Erstberichterstatter: Univ.-Prof. Dr.-Ing. Günter Schmidt
Zweitberichterstatter: Prof. Dr. Dr. h.c. mult. Kurt Mehlhorn
Tag des Kolloquiums: 18. Juli 2011
Dekan: Univ.-Prof. Dr. Holger Hermanns
Mitglieder des Prüfungsausschusses: Prof. Dr. Dr. h.c. mult. Reinhard Wilhelm
Univ.-Prof. Dr.-Ing. Günter Schmidt
Prof. Dr. Dr. h.c. mult. Kurt Mehlhorn
PD Dr. Rob van Stee
Acknowledgment

First and foremost I would like to express my sincere gratitude to my supervisor Günter Schmidt. His deep questions and ambition to solve problems inspired me to go on, in particular when facing difficulties. Similarly, I am grateful to Kurt Mehlhorn who has kindly agreed to be my co-supervisor. I would also like to thank the other members of the committee, Reinhard Wilhelm and Rob van Stee, for their guidance and support.

The lively and motivating research group at the Saarland University made frustrations a lot less worse, when yet another proof had crashed. Especially Mike Kersch and Iftikhar Ahmad deserve a special thanks for their comments and discussions on new ideas. I remember the fun and our weird calculations on anything that was handy. Further, the generous donations of cookies deserve an extra thanks. I had the pleasure to work with Benjamin Olshok, Hai Dung Dinh, Robert Dochow and Nadine Thiel as office mates. They, and all student assistants supporting the group, made research a great experience. Especially, I am indebted to Hedi Staub for her technical and organizational support.

One day, I started the sports of rowing, which enthused me immediately. Esther König deserves a special ‘thank you’. I am grateful for our numerous rowing and cooking sessions to compensate for writing this thesis. Moreover, I appreciate the thought-provoking discussions with Dieter Schmidtchen and the brain teasers at the bar counter.

Last but not at all least, I thank my family and friends. You have always been a source of motivation and encouragement.

Saarbrücken, March 2011
Esther Mohr
Summary

A conversion problem deals with the scenario of converting an asset into another asset and possibly back. This work considers financial assets and investigates online algorithms to perform the conversion. When analyzing the performance of online conversion algorithms, as yet the common approach is to analyze heuristic conversion algorithms from an experimental perspective, and to analyze guaranteeing conversion algorithms from an analytical perspective. This work conjoins these two approaches in order to verify an algorithms’ applicability to practical problems. We focus on the analysis of preemptive and non-preemptive online conversion problems from the literature. We derive both, empirical-case as well as worst-case results. Competitive analysis is done by considering worst-case scenarios. First, the question whether the applicability of heuristic conversion algorithms can be verified through competitive analysis is to be answered. The competitive ratio of selected heuristic algorithms is derived using competitive analysis. Second, the question whether the applicability of guaranteeing conversion algorithms can be verified through experiments is to be answered. Empirical-case results of selected guaranteeing algorithms are derived using exploratory data analysis. Backtesting is done assuming uncertainty about asset prices, and the results are analyzed statistically. Empirical-case analysis quantifies the return to be expected based on historical data. In contrast, the worst-case competitive analysis approach minimizes the maximum regret based on worst-case scenarios. Hence the results, presented in the form of research papers, show that combining this optimistic view with this pessimistic view provides an insight into the applicability of online conversion algorithms to practical problems. The work concludes giving directions for future work.
Zusammenfassung

Contents

List of Figures xiii
List of Tables xv
List of Variables xvii

1 Introduction 1
1.1 Preliminaries ......................................................... 1
1.2 Research Question ................................................... 5
1.3 Financial Markets and Online Conversion Algorithms .............. 6
    1.3.1 Heuristic Algorithms ....................................... 10
    1.3.2 Guaranteeing Algorithms .................................... 12
1.4 Trading Systems ...................................................... 13
References for Chapter 1 .............................................. 15

2 Competitive Analysis of Online Conversion Algorithms 23
2.1 Online and Offline Algorithms .................................... 23
2.2 Online Conversion Problems ....................................... 24
    2.2.1 Uni-directional Search ..................................... 28
    2.2.2 Bi-directional Search ....................................... 29
2.3 Competitive Analysis ............................................... 30
    2.3.1 Competitive Ratio for Uni-directional Search ............... 31
    2.3.2 Competitive Ratio for Bi-directional Search ............... 33
    2.3.3 Worst-case and Empirical-case Competitive Ratio ........... 35
2.4 Literature Review .................................................. 36
    2.4.1 Non-Preemptive Conversion .................................. 37
        2.4.1.1 Uni-directional Search ................................ 37
        2.4.1.2 Bi-directional Search ................................ 39
    2.4.2 Preemptive Conversion ....................................... 41
        2.4.2.1 Uni-directional Search ................................ 42
        2.4.2.2 Bi-directional Search ................................ 47
References for Chapter 2 .................................................. 48

3 Empirical Analysis of Online Conversion Algorithms .......................... 55
  3.1 Introduction .............................................................. 55
  3.2 Backtesting and Stylized Facts ....................................... 58
  3.3 Exploratory Data Analysis ............................................ 63
    3.3.1 Hypothesis Testing ............................................... 65
    3.3.2 Resampling: The Bootstrap Procedure .......................... 68
  3.4 Literature Review ..................................................... 71
References for Chapter 3 .................................................... 75

4 Selected Non-preemptive Algorithms ........................................... 81
  4.1 The Uni-directional Algorithm of El-Yaniv (1998) ..................... 81
    4.1.1 The Guaranteeing Algorithm .................................... 81
    4.1.2 Worst-Case Analysis ............................................ 82
  4.2 Extension to Bi-directional Search of Mohr and Schmidt (2008a) .... 83
    4.2.1 The Guaranteeing Algorithm .................................... 83
    4.2.2 Worst-Case Analysis ............................................ 84
  4.3 The Bi-directional Algorithms of Brock, Lakonishok and LeBaron (1992) .................................................. 85
    4.3.1 Moving Average Crossover ....................................... 86
    4.3.2 Worst-Case Analysis ............................................ 86
    4.3.3 Trading Range Breakout ......................................... 87
    4.3.4 Worst-Case Analysis ............................................ 87
References for Chapter 4 .................................................... 87

5 Selected Preemptive Algorithms ............................................... 91
  5.1 The Uni-directional Algorithm of El-Yaniv, Fiat, Karp and Turpin (1992) .................................................. 91
    5.1.1 The Guaranteeing Algorithm .................................... 92
    5.1.2 Worst-Case Analysis of Variant 1: Threat($m, M$) and Threat($m, M, q_1$) .................................................. 94
    5.1.3 Worst-Case Analysis of Variant 2: Threat($m, M, k$) .......... 96
    5.1.4 Worst-Case Analysis of Variant 3: Threat($\varphi, k$) ........ 100
    5.1.5 Worst-Case Analysis of Variant 4: Threat($\varphi$) ............ 100
    5.1.6 Numerical Examples for Variant 1 to 4 ........................ 101
  5.2 The Bi-directional Algorithm of El-Yaniv, Fiat, Karp and Turpin (1992) .................................................. 115
    5.2.1 The Guaranteeing Algorithm .................................... 115
    5.2.2 Worst-Case Analysis ............................................ 116
CONTENTS

5.3 Improvement Idea of Dannoura and Sakurai (1998) . . . . . . . . . 118
  5.3.1 The Guaranteeing Algorithm . . . . . . . . . . . . . . . . . . . 119
  5.3.2 Worst-Case Analysis . . . . . . . . . . . . . . . . . . . . . . . . 120
References for Chapter 5 . . . . . . . . . . . . . . . . . . . . . . . . . . . 122

6 Results 125
  6.1 Results of Mohr and Schmidt (2008) . . . . . . . . . . . . . . . . . 125
    6.1.1 Mohr and Schmidt (2008a) . . . . . . . . . . . . . . . . . . . 126
    6.1.2 Mohr and Schmidt (2008b) . . . . . . . . . . . . . . . . . . . 137
  6.2 Results of Schmidt, Mohr and Kersch (2010) . . . . . . . . . . . 145
  6.3 Results of Ahmad, Mohr and Schmidt (2010) . . . . . . . . . . . 155
  6.4 Results of Mohr and Schmidt (2010) . . . . . . . . . . . . . . . . . 193
References for Chapter 6 . . . . . . . . . . . . . . . . . . . . . . . . . . . 213

7 Conclusions and Future Work 215
  7.1 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 215
  7.2 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 217
References for Chapter 7 . . . . . . . . . . . . . . . . . . . . . . . . . . . 219

References xxi

A Copyright Permissions xlv
  A.1 Springer, Heidelberg . . . . . . . . . . . . . . . . . . . . . . . . . xlv
  A.2 Elsevier B.V. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xlv

B Eidesstattliche Versicherung xlix
List of Figures

3.1 Positive Skewness ................................................. 61
3.2 Kurtosis ............................................................ 62
List of Tables

3.1 Stylized facts of the German Dax-30 index for 1998-2007 .... 63

5.1 Numerical example for Variant 1 with \( q_1 = c^{\infty}(m, M) \cdot m \) .... 103
5.2 Numerical example for Variant 2 with \( q_1 = c^{k'}(q_1) \cdot m \) .... 104
5.3 Numerical example for Variant 3 with \( q_1 = c(\varphi, k) \cdot m \) .... 104
5.4 Numerical example for Variant 4 with \( q_1 = c(\varphi) \cdot m \) .... 105
5.5 Numerical example for Variant 1a with \( M > q_1 > c^{\infty}(m, M) \cdot m \) and \( q_1 \) assumed to be unknown a-priori .... 107
5.6 Numerical example for Variant 1b with \( M > q_1 > c^{\infty}(m, M) \cdot m \) and \( q_1 \) assumed to be known a-priori .... 108
5.7 Numerical example for Variant 2 with \( M > q_1 > c^{k'}(q_1) \cdot m \) .... 109
5.8 Numerical example for Variant 3 with \( M > q_1 > c(\varphi, k) \cdot m \) .... 109
5.9 Numerical example for Variant 4 with \( M > q_1 > c(\varphi) \cdot m \) .... 110
5.10 Numerical example for Variant 1 with \( M = q_1 \) .... 111
5.11 Numerical example for Variant 2 with \( M = q_1 \) .... 112
5.12 Numerical example for Variant 3 with \( M = q_1 \) .... 113
5.13 Numerical example for Variant 4 with \( M = q_1 \) .... 114
### List of Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>Conversion algorithm, with $X \in {OPT, ON}$</td>
</tr>
<tr>
<td>$ON$</td>
<td>Online conversion algorithm</td>
</tr>
<tr>
<td>$OPT$</td>
<td>Optimal offline algorithm</td>
</tr>
<tr>
<td>$i$</td>
<td>Counter</td>
</tr>
<tr>
<td>$j$</td>
<td>Asset</td>
</tr>
<tr>
<td>$J$</td>
<td>Number of assets, with $j = 1, \ldots, J$</td>
</tr>
<tr>
<td>$d(j, i)$</td>
<td>Number of dividends issued by $j$ within $i$-th time interval</td>
</tr>
<tr>
<td>$e$</td>
<td>Base of the natural logarithm</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>Epsilon</td>
</tr>
<tr>
<td>$z$</td>
<td>Constant</td>
</tr>
<tr>
<td>$c$</td>
<td>Competitive ratio</td>
</tr>
<tr>
<td>$c^{wc}$</td>
<td>Worst-case competitive ratio</td>
</tr>
<tr>
<td>$c^{ec}$</td>
<td>Empirical-case competitive ratio</td>
</tr>
<tr>
<td>$Q$</td>
<td>Worst-case time series</td>
</tr>
<tr>
<td>$a$</td>
<td>Level of risk, with $a \in [1, c]$</td>
</tr>
<tr>
<td>$M$</td>
<td>Upper bound of prices</td>
</tr>
<tr>
<td>$m$</td>
<td>Lower bound of prices</td>
</tr>
<tr>
<td>$\varphi$</td>
<td>Price fluctuation ratio, equaling $\frac{M}{m}$</td>
</tr>
<tr>
<td>$t$</td>
<td>Data point, e.g. day</td>
</tr>
<tr>
<td>$w$</td>
<td>Number of price maxima, with $i = 1, \ldots, w$</td>
</tr>
<tr>
<td>$p$</td>
<td>Number of time intervals, with $i = 1, \ldots, p$</td>
</tr>
<tr>
<td>$I$</td>
<td>Input sequence, with $t = 1, \ldots, T$ elements</td>
</tr>
<tr>
<td>$T$</td>
<td>Time interval length</td>
</tr>
<tr>
<td>$k$</td>
<td>Number of trading days, with $k \leq T$</td>
</tr>
<tr>
<td>$T'$</td>
<td>Remaining number of trading days, with $T' = T - t + 1$</td>
</tr>
<tr>
<td>$k'$</td>
<td>Remaining $k$ before the price drops to $m$, with $k' = k - t + 1$</td>
</tr>
<tr>
<td>$s_t$</td>
<td>Amount to be converted by $X$ on $t$, with $0 \leq s_t \leq 1$</td>
</tr>
</tbody>
</table>
Number of prices to be considered for conversion, with 
\( i = 1, \ldots, u \) and \( u \leq T \)

Initial amount of asset \( D \)

Initial amount of asset \( Y \)

Accumulated amount of asset \( Y \) on \( t \)

Remaining amount of asset \( D \) on \( t \)

Price on \( t \), with \( q_t \in [m, M] \)

Reservation price

Return

Arithmetic mean

Accepted price

Return for \( q' \)

Discrete return on \( t \) in \( i \)-th time interval, with \( i = 1, \ldots, p \)

Geometric return of \( X \) over \( p \)

Logarithmic return of \( X \) over \( p \), equaling \( \ln R_X(p) \)

Return function

Price function

Test statistic (\( t \)-test)

Significance level, with \( \alpha \in [0, 1] \)

Degrees of freedom

Critical value (\( t \)-statistic)

Return to be expected from \( X \)

Standard deviation, the square root of variance \( \sigma^2 \)

Weight of \( j \) within a portfolio

Skewness

Kurtosis

Probability of \( \cdot \)

Number of bootstrap samples, with \( i = 1, \ldots, S \)

Bootstrap block size, with \( 1 \leq l \leq S \)

Number of bootstrap blocks, equaling \( \frac{S}{l} \), with \( i = 1, \ldots, b \)

Observation on \( t \)

Bootstrap block, formed of \( l \) consecutive \( x_t \), with \( i = 1, \ldots, b \)

Arithmetic mean of \( X \) on \( i \)-th bootstrap sample, with \( i = 1, \ldots, S \)

Number of data points used, with \( t > n \)
$MA(n)_t$  
Moving average over $n$ on $t$, equaling $\frac{\sum_{i=t-n+1}^{t} q_i}{n}$

$\delta$  
Band lagging $q_t$, with $\delta \in [0.00, \infty]$  

$q_{t}^{\text{min}}(n)$  
Local minimum price over $n$ on $t$, equaling $\min \{q_i| i = t-n, \ldots, t-1\}$

$q_{t}^{\text{max}}(n)$  
Local maximum price over $n$ on $t$, equaling $\max \{q_i| i = t-n, \ldots, t-1\}$

$\text{iid}$  
Independent, identical distributed
Chapter 1

Introduction

This chapter introduces online problems and conversion algorithms in the context of conversion in financial markets and specifies how (not) to evaluate their quality. We give basic definitions and state the research questions to be answered. Then we focus on financial markets mentioning the relevant related work. The chapter concludes with an overview on trading systems as the 'tool' for evaluating online conversion algorithms.

1.1 Preliminaries

A conversion problem deals with the scenario of converting an asset $D$ into another asset $Y$ with the objective to get the maximum amount of $Y$ after time $T$. The process of conversion can be repeated in both directions, i.e. converting asset $D$ into asset $Y$, and asset $Y$ back into asset $D$. Within this work we consider financial assets and investigate online algorithms to perform the conversion.

In a typical problem setting, an investment horizon is considered and possibly divided into $i = 1, \ldots, p$ time intervals. Each $i$-th time interval is comprised of $t = 1, \ldots, T$ data points, e.g. days. On each day $t$, an algorithm $X$ is offered a price $q_t$ to convert asset $D$ into asset $Y$, and $X$ may accept the price $q_t$ or may decide to wait for a better price. The ‘game’ ends either when $X$ converts whole of the asset $D$ into $Y$, or on the last day $T$ where $q_T$ must be accepted.

In an offline scenario full information about the future is assumed, and so an optimal offline algorithm ($OPT$) is carried out. In an online scenario at each point of time an algorithm must take a decision based only on past information, i.e. with no knowledge about the future. Online conversion algorithms ($ON$) solve this problem. Typically, the quality of $ON$ is determined by the relation between the result generated by $ON$, and the optimal offline result generated by $OPT$ (Schmidt, 2006, p. 280). But in the work related two further approaches
Thus, before introducing online conversion algorithms, we must decide how (not) to evaluate their quality. Basically the performance analysis of a conversion algorithm $X \in \{OPT, ON\}$ can be carried out by three different approaches.

The first approach is to assume that input data is given according to a certain probability distribution, and to compute the expected behavior of an algorithm based on this distribution. This approach is called 'Bayesian Analysis', the traditional approach within the literature when analyzing conversion algorithms (Chou, 1994; Pásztor, 2000; Arakelian and Tzionas, 2008), and has been dominant over the last several decades (El-Yaniv, 1998, pp. 34-35). The objective is to optimize an algorithms empirical (average-case) performance under 'typical inputs' assuming a specific stochastic model (Karp, 1992a,b). Either assumptions about the distribution of the input data are made, or the distribution of the input data is assumed to be known beforehand (Babaioff et al., 2008). It is beyond the scope of this work to survey the 'Bayesian' work related. The reader is referred to Kakkade and Kearns (2005) and Fujiwara et al. (2011) analyzing various assumptions on the underlying price processes.

However, this approach can often not be applied as distributions are rarely known precisely. It is often extremely difficult to assume realistic statistical models for possible input sequences (which are always highly dependent on the particular application). Thus, distributional assumptions are often unrealistically crude (Borodin and El-Yaniv, 1998, p. xxiii). Moreover, even if the input in question follows a particular input distribution, it is often difficult to identify or construct a stochastic model that accurately reflects this distribution. For instance, a great deal of effort has been invested in attempt to identify the probability distributions of currency exchange rates, but there is still no evidence that such distributions exist (Chou, 1994). As a result, some research attempts to relax distributional assumptions. Rosenfield and Shapiro (1981) study the case where the price distribution itself is a random variable. In this regard Cover and Gluss (1986) consider online portfolio selection, reallocating their portfolio on the past behavior of the market. The goal is to perform just as well as if the empirical distribution of the prices is assumed to be known. Cover and Gluss (1986) show that an online algorithm not knowing the empirical distribution of the prices in advance can perform as well as an optimal algorithm. Thus, when analyzing conversion algorithms we wish to avoid making assumptions about input distributions or probabilities.

This leads to the second approach. Uncertainty about asset prices is assumed and conversion algorithms are analyzed considering worst-case scenarios.

\footnote{Also called probabilistic analysis (Borodin and El-Yaniv, 1998, p. xxiii) or distributional analysis (Chou, 1994, p. 9).}
This analytic approach is most frequently used in computer science as the empirical (average-case) performance is often roughly as bad as the worst-case performance, and worst-case measures additionally provide a definite upper bound (Cormen et al., 2001, p. 26). The approach does not demand that inputs come from some known distribution but instead compares the performance of an online algorithm to that of an adversary; the optimal offline algorithm. This notion of comparison is called competitive analysis. It is assumed that the online algorithm has no knowledge about future input data. Inputs are generated by the adversary who knows the entire future, and thus operates optimally (El-Yaniv et al., 1999). An online algorithm is called $c$-competitive, if its competitive ratio — the ratio between the performance of ON and OPT — is bounded by some constant $c$, which gives a worst-case performance guarantee. It is desirable to choose an algorithm with a preferably low competitive ratio. El-Yaniv et al. (1992) suggested to apply competitive analysis to online conversion algorithms where $c$ measures the quality of ON.

A lot of work related exists in the field of online algorithms and online optimization. Important results are presented in the book of Fiat and Woeginger (1998), as well as in the book of Borodin and El-Yaniv (1998). A survey on classical competitive analysis for online algorithms is given in Albers (2003). Further, within the work related, there are three different approaches to improve the competitive ratio of an online algorithm.

The first approach is to restrict the power of the adversary by allowing only certain input distributions. Raghavan (1992) and Chou et al. (1995) assume that the input sequence is generated by the adversary and has to satisfy specific statistical properties. The adversary is thus named ‘statistical adversary’. The approach may be considered as a hybrid of ‘Bayesian Analysis’ and competitive analysis. In this regard Koutsoupias and Papadimitriou (2000) consider a ‘partial knowledge’ of the input distribution by the online algorithm. Garg et al. (2008) study online algorithms under the assumption that the input is not chosen by an adversary, but consists of draws from a given probability distribution. All these approaches improve (lower) the competitive ratio by weakening the adversary, but do not lead to better online (conversion) algorithms, and thus are not considered here.

The second (most popular) approach is to relatively restrict the power of the adversary by using randomization. It is assumed that the adversary has relatively less power since the moves of an online algorithm are no longer certain (Fiat et al., 1991). We consider an optimal offline adversary knowing the entire future, even

\footnote{Chapter 2 shows how exactly to quantify the quality of ON by introducing the notion of competitive analysis.}
the random number generator. From this follows that randomization does not help (Borodin et al., 1992), and is also not considered here.

The third approach addresses ‘forecasts’ on the input sequence. The basic idea is that ON is allowed to make a forecast. In case the forecast comes true the competitive ratio improves, which is considered as a reward. In case the forecast comes not true, the best achievable worst-case ratio holds. Al-Binali (1997, 1999) provides a framework of ‘risk and reward’ in which investors may develop online algorithms based on their acceptable level of risk (‘risk tolerance’) and a ‘forecast’ on future price movements. Iwama and Yonezawa (1999) generalize this framework by introducing ‘forecast levels’ which forecast that prices $q_t$ will never increase (decrease) to some level, and present different online algorithms using these levels. In this regard Halldorsson et al. (2002) suggest to allow an online algorithm to maintain several different solutions, and to select one of them (the best one) at the end. As yet, these works have not been analyzed experimentally, and thus are potential new areas of research.

In case the input data processed by an online (conversion) algorithm does not represent the worst-case, its performance is considerably better than the competitive ratio tells. For this reason competitive analysis is criticized as being too pessimistic. Borodin and El-Yaniv (1998, p. xxiv) admit that in some application areas, especially in finance, worst-case performance guarantees are essential, e.g. in case of a stock market meltdown. But in terms of practical application the worst-case competitive ratio does not reveal which returns can be expected in practice, nor whether these returns are positive or not.

This leads to the third approach. In this experimental approach conversion algorithms $X \in \{OPT, ON\}$ are implemented, and the analysis is done on historic or artificial data by simulation runs. This approach is exploratory, since the empirical-case results suggest which hypotheses to test (statistically). From this follows that conversion algorithms can be evaluated using exploratory data analysis (EDA). The objective of EDA is to 1) suggest hypotheses to test (statistically) based on the results generated, 2) assess assumptions on the statistical inference, 3) support the selection of appropriate statistical tools and techniques for further analysis, and 4) provide a basis for further data collection through experiments. It is important to distinguish the EDA approach from classical hypothesis testing, which requires a-priori formulated hypotheses (Hoaglin et al., 2000). By applying EDA the observed empirical-case results are evaluated statistically, mainly by hypothesis tests, bootstrap methods, or Monte Carlo simulation (Brock et al., 1992; Steiglitz et al., 1996; Biais et al., 2005; Tabak and Lima, 2009; Schmidt et al., 2010). The classical question regarding the predictive ability of ON is to be answered: ‘Is it possible to forecast returns in a particular (future) time interval
by using the returns observed in a previous time interval?” (Pierdzioch, 2004).

To analyze online conversion algorithms, we apply the EDA (third) approach, and compare the results to those of the competitive analysis (second) approach. For the empirical-case the actually observed performance considering the experimental data is analyzed, and hypotheses to be evaluated statistically are derived. Further, competitive analysis is done by considering on the one hand worst-case scenarios, i.e. the worst possible input data which could have been occurred is used when calculating the worst-case competitive ratio $c^{wc}$. On the other hand, the actually observed input data is considered, i.e. the empirical-case performance on the experimental data is used when calculating the empirical-case competitive ratio $c^{ec}$. Hence, we aim to conjoin empirical-case analysis and worst-case analysis. This leads to the following research questions.

1.2 Research Question

When analyzing conversion algorithms, as yet the common approach is to experimentally analyze online conversion algorithms designed to achieve a possibly high empirical-case performance (heuristic conversion algorithms), and to mathematically analyze online conversion algorithms designed to give a worst-case performance guarantee (guaranteeing conversion algorithms). Our aim is to conjoin these two approaches in order to verify the applicability of both classes of online conversion algorithms to practical problems.

On the one hand we focus on the new field of worst-case analysis of heuristic conversion algorithms, and compare the results to the empirical-case results.

**Question 1**: Can the applicability of heuristic conversion algorithms be verified through competitive analysis, and which worst-case competitive ratio $c^{wc}$ do they achieve?

To answer Question 1 heuristic conversion algorithms from the literature are considered, and competitive analysis is done: The heuristic conversion algorithms of Brock et al. (1992) are analyzed, i.e. worst-case competitive ratios $c^{wc}$ are derived.

On the other hand we focus on the new field of experimental analysis of guaranteeing conversion algorithms, and compare the results to the analytical worst-case results.

**Question 2**: Can the applicability of guaranteeing conversion algorithms be verified through experiments, and which empirical-case performance do they achieve?
To answer Question 2 different guaranteeing conversion algorithms from the literature are considered, and experimental analysis is done.\(^3\) The guaranteeing conversion algorithms of El-Yaniv (1998); Dannoura and Sakurai (1998) and El-Yaniv et al. (1992, 2001) are analyzed, i.e. the empirical-case performance is derived through experiments. To measure the applicability of the algorithms considered the empirical-case competitive ratio $c^{ec}$ as well as the return to be expected $\mu$ is derived.

Summing up, we are interested in analyzing online conversion algorithms from an analytical and an experimental perspective in order to verify their applicability to practical problems.

The remainder of this work is organized as follows: The next section gives a brief introduction to financial markets, online conversion algorithms and trading systems. Chapter 2 introduces online financial search and conversion problems as well as the notion of competitive analysis. Further, a detailed overview on work related to online conversion problems is given. Chapter 3 presents the approach to experimental analysis of online conversion algorithms. Exploratory data analysis (EDA) is introduced, and the steps how to empirically analyze online conversion algorithm using this data analysis approach are provided. A detailed overview on the work related is given. Chapter 4 presents the new field of worst-case analysis of heuristic conversion algorithms. We focus on the Moving Average and Trading Range Breakout algorithms introduced by Brock et al. (1992). Chapter 5 presents the guaranteeing conversion algorithms introduced by El-Yaniv (1998); Dannoura and Sakurai (1998) and El-Yaniv et al. (1992, 2001) in detail. Chapter 6 presents empirical-case results of the guaranteeing conversion algorithms reviewed in Chapter 5 as well as analytical worst-case results of the heuristic conversion algorithms reviewed in Chapter 4. The results are given in the form of research papers published in/submitted to different journals. Prior to each publication a preface is given linking the topic of the paper to this thesis. Chapter 7 concludes and gives some directions for future work.

1.3 Financial Markets and Online Conversion Algorithms

In general, algorithms used in financial markets aim different objectives. They are designed to (cf. Bertsimas and Lo (1998)):

1. Optimize the trade execution,

\(^3\)Experimental analysis in other fields can be found in Karlin (1998); Albers and Jacobs (2010).
2. maximize the return to be expected $\mu$,
3. exploit different price patterns or price dynamics,
4. minimize the expected transaction costs,
5. give a performance guarantee under worst-case conditions,
6. minimize the risk,
7. balance the trade-off between the return to be expected and the incurred risk,
8. convert fixed blocks of assets,
9. convert over a fixed finite number of time intervals $p$.

Assets are 'things' owned by an individual. They can be physical, financial or intellectual. Stocks are a shares of a company. As a financial asset, stocks can be bought and sold by the help of conversion algorithms. These algorithms aim either to buy at possibly low prices or to sell at possibly high prices, or both. The goal is to automatically determine entry point(s) before a market increase, and exit point(s) before a market downturn, often based on historic or predicted price movements. Hence, every conversion algorithm consists of at least one buying rule and one selling rule represented by (source program) statements specifying the exact entry and exit points. A typical example for a buying rule is the IF-THEN statement, for example BUY IF $q_t \leq x_t$. Here a buying signal is generated if the price $q_t$ is smaller than or equal to some observation $x_t$. As an order, these signals can be executed on the stock market. Further, buying and selling rules of different algorithms can be combined to more complex algorithms, e.g. by using genetic programming (Potvin et al., 2004).

We focus on algorithms aiming the objectives 2 and 5. Based on the design pattern of these algorithms, we can broadly classify them into two classes, a) online conversion algorithms — developed to give a performance guarantee under worst-case conditions, and referred to as guaranteeing conversion algorithms, and b) heuristic conversion algorithms — developed to achieve a preferably high empirical-case performance.

**a) Guaranteeing conversion algorithms** are developed to give a performance guarantee under worst-case conditions. The worst-case performance guarantee is usually evaluated using competitive analysis (second approach), assuming uncertainty about the future input sequence $I$ (El-Yaniv, 1998). The performance guarantee is measured in terms of the competitive ratio (Fiat and Woeginger, 1998, p. 4).
b) **Heuristic conversion algorithms** are developed to achieve a preferably high empirical-case performance. Very often these algorithms are based on data from technical analysis (Brock et al., 1992; Vanstone and Finnie, 2009), artificial intelligence (Palmer et al., 1994; Kumar et al., 1997; Feng et al., 2004), neural networks (Schulenberg and Ross, 2002; Chavarnakul and Enke, 2008), genetic algorithms/programming (Dempster and Jones, 2001; Korczak and Roger, 2002; Potvin et al., 2004), or software agents (Silaghi and Robu, 2005). The empirical-case performance is usually evaluated either using ‘Bayesian Analysis’ (first approach) or EDA (third approach), and measured in terms of the return to be expected \( \mu \).

Using the competitive ratio, the behavior of heuristic conversion algorithms is found similar to guaranteeing conversion algorithms, as both classes work without any knowledge of future input. We conclude heuristic conversion algorithms are also online conversion algorithms, and can be analyzed using competitive analysis. Thus, both classes of algorithms are referred to as online conversion algorithms (ON).

Irrespective of the application area, online algorithms are related to approximation algorithms. Both seek to obtain a good approximation to some optimal solution, i.e. guarantee a specific fraction of the optimal offline result. The difference lies in that approximation algorithms (also known as computational complexity algorithms) deal with the question what resources would be needed to compute a solution, namely the computational complexity. The goal is to determine the trade-off between the computational complexity and the quality of the solution the algorithm computes. As the computational resources available are limited, approximation algorithms deal with complexity measurement. In contrast, online algorithms focus on the limitations caused by a lack of information, and not on the limitations caused by a lack of running time (approximation algorithms). Thus, competitive analysis is an information theoretic measure, not a computational complexity measure (Fiat and Woeginger, 1998, p. 5).

For evaluating online conversion algorithms the **order type** is irrelevant. But in case ON is considered for practical use the order type is essential as it is superior to the signals generated by ON. Hence, the most frequently used order types are briefly presented in the following.

A **market order** is an order to buy or sell an asset at the current market price. Unless specified otherwise, orders are entered as a market order, e.g. by a broker. The advantage of a market order is that it is almost always guaranteed that the order will be executed. The disadvantage is that when a market order is placed, the price at which the order will be executed cannot be controlled. To avoid buying
or selling an asset at a price higher or lower than a certain level, a limit order must be placed. A limit order is an order to buy or sell at a predefined reservation price or ‘better’: A buy limit order can only be executed at the limit price or lower, a sell limit order can only be executed at the limit price or higher.

Example 1. Assume an investor wants to buy an asset that was initially offered at $9, but does not want to end up paying more than $10. Then a limit order to buy the asset at any price up to $10 should be placed.

The advantage of using a limit order is that the investor protects himself from buying (selling) the asset at a too high (low) price. The disadvantage is that a limit order may never be executed because the market price may surpass the investors limit before the order can be filled.

A stop order is an order to buy or sell an asset once it reaches a specified price, namely the stop price. A buy stop order is used to invest in case of a trend reversal. In case of short selling\(^4\), it is used to limit a loss or to protect a profit. A buy stop order is entered at a stop price that is always above the current market price. A sell stop order avoids further losses or protects a profit that exists if a price drops. A sell stop order is always placed below the current market price. The advantage of a stop order is that the price movement must not be monitored. The disadvantage is that the stop price could be activated by a short-term price fluctuation. Once a stop price is reached the stop order becomes a market order. The received price may differ from the stop price, especially in markets with high volatility. An investor can avoid the risk of a stop order not guaranteeing a specific price by placing a stop-limit order. A stop-limit order combines the features of stop and limit order. Once the stop price is reached, the stop-limit order becomes a limit order.

The computerized execution of financial instruments following prespecified rules and guidelines is called algorithmic trading (Kissel and Malamut, 2006). Like Grossman (2005) and Domowitz and Yegerman (2006), we define the term algorithmic trading as the automated, computer-based execution (submission and canceling) of orders via direct market-access channels. Usually, the goal is to meet a particular benchmark, e.g. the volume-weighted average price (VWAP) over the execution interval (Coggins et al., 2006). In contrast to online conversion algorithms, algorithmic trading defines certain aspects of an order, but never the points of time to take a buying or selling decision. Algorithmic trading strategies execute orders and typically determine order type, timing, routing and quantity, while dynamically monitoring market conditions across different market places. To reduce the market impact by optimally (or randomly) breaking large orders

\(^4\)The selling of an asset the seller does not own.
into smaller pieces, and to track benchmarks are the main tasks. The aim is to optimize the trade execution (Nevmyvaka et al., 2006). Often a mix of active and passive strategies is used, employing different order types. The scope of this work are online conversion algorithms solving the financial search problem. Thus, algorithmic trading is not considered and the reader is referred to the surveys by Gomber et al. (2005); Fraenkle and Rachev (2009), and Hendershott et al. (2010).

Every stock market investor has an own idea of how the most profitable stocks can be found, and at what time they should be bought and sold. First, a decision must be taken which class of online conversion algorithms (heuristic or guaranteeing) should be applied. In the following heuristic conversion algorithms as well as guaranteeing conversion algorithms are presented in detail.

1.3.1 Heuristic Algorithms

Many practical problems are unlikely to admit exact (optimal) solutions in a reasonable amount of time. Hence heuristics are sought for these problems – these algorithms try to find a possibly ‘good’ solution, not necessarily the best one, in a small amount of time. Heuristic conversion algorithms attempt to identify and exploit winners or trends and are designed to achieve a preferably high empirical-case performance. The starting point for the creation of a heuristic conversion algorithm is the selection of input variables likely to influence the desired outcome, i.e. to maximize the return to be expected $\mu$. There is a great number of methods used and they broadly fall in the area of either Fundamental Analysis, or Technical Analysis. It is essential to have an understanding of these two complementary forms of analysis and their possible effect, so that an ‘intelligent’ choice of input variables can be made (Vanstone and Finnie, 2009).

Fundamental Analysis uses economic data to forecast prices or to determine whether the markets are over- or undervalued. The goal is to use so-called financial ratios produced from business ratios as predictors of a company’s future stock price, return or price direction. Financial ratios can for instance be 1) the stock price compared to its actual earning, 2) the actual value of an asset compared to the book value, 3) balance sheets, or 4) the last development of consumption spending in a specified country. For a detailed overview on Fundamental Analysis and work related the reader is referred to Vanstone and Finnie (2009, pp. 6670-6672) and the books of Murphy (1999) and Malkiel (2003).

Technical Analysis seeks to identify price patterns and trends in financial markets. The goal is to exploit those patterns, and to forecast future price directions through the study of past market data, primarily price and volume (Murphy, 1999). Technical Analysis is composed of four techniques (cf. Schmidt
1. Charting, the study of price charts, typically done by pattern matching.

2. Elliott waves, the study of mathematical properties of waves and patterns, based on Fibonacci numbers.

3. Heuristic conversion algorithms, the calculation of indicators and oscillators, typically mathematical transformations of price or volume.

4. Esoteric approaches, e.g. weather-based strategies.

Charting is usually highly subjective and without ‘rigorous’ mathematical definition. Malkiel (2003) concludes that ‘under scientific scrutiny, chart-reading must share a pedestal with alchemy’, and thus is not considered here. Nevertheless, several academic studies suggest charting for extracting useful information about market prices (Lo et al., 2000, p. 1706). The Elliott wave principle by R.N. Elliott (1871-1948) analyzes the mathematical properties of waves and patterns based on Fibonacci numbers. These numbers are closely connected to the Golden ratio (0.618), as the quotient of neighboring Fibonacci numbers is 0.618. Practitioners commonly use the Golden ratio to forecast levels of future market waves based on their relation to past market waves (Schmidt, 2006, pp. 218-219). Elliott waves are not considered here. Esoteric approaches are also excluded, as they have no scientific justification (cf. Hirshleifer and Shumway, 2003). The remainder of this work will only consider research support for the use of heuristic conversion algorithms. However, these algorithms are not considered by many researchers. The main reason is the Efficient Market Hypothesis (EMH), which supports the random-walk theory (RWT). The intuition behind the EMH is simple: Market prices follow a random walk and cannot be predicted based on their past behavior. Hence, markets efficiently process all relevant information into a single price. In essence, the RWT states that price changes in stock markets are independent, identical distributed (iid) random variables. This implies that a time series of prices has no ‘memory’, which further implies that the study of past prices cannot provide a useful contribution to predicting future prices or price movements. As main method to determine the return to be expected is backtesting, the conclusion is that heuristic conversion algorithms cannot work (see e.g. Fama, 1965; Leigh et al., 2002; Tabak and Lima, 2009). Of course, there are also numerous works questioning various aspects of the EMH, or fail to confirm it (see e.g. Leigh et al., 2002; Findlay et al., 2003). Thus, regardless of the EMH, a large number of practitioners use heuristic conversion algorithms as their main method to determine transaction points (Taylor and Allen, 1992).
In general, heuristic conversion algorithms are reservation price (RP) algorithms. Reservation price(s) \( q^* \) are calculated for each day \( t \) based on the offered price \( q_t \). Using the \( q^* \), the RP algorithm determines transaction points specifying when to buy or sell. The majority of work related concerns the empirical analysis of simple RP algorithms. 'Truly' effective algorithms are usually kept secret (Vanstone and Finnie, 2009, p. 6673). We limit to the heuristic RP algorithms introduced by Brock et al. (1992), namely Moving Average Crossover (MA) and Trading Range Breakout (TRB), which are based on technical indicators. These algorithms are of major interest in the literature and have been analyzed by several researchers, cf. Bessembinder and Chan (1995); Hudson et al. (1996); Mills (1997); Ratner and Leal (1999); Parisi and Vasquez (2000); Gunasekarage and Power (2001); Kwon and Kish (2002); Chang et al. (2004); Bokhari et al. (2005); Marshall and Cahan (2005); Ming-Ming and Siok-Hwa (2006); Hatgioannides and Mesomeris (2007); Lento and Gradojevic (2007); Lagoarde-Segot and Lucey (2008) and Tabak and Lima (2009). These works on MA and TRB are restricted to empirical-case results, and do not take into account worst-case results (which we derive in Chapter 4).

1.3.2 Guaranteeing Algorithms

Decision making can be considered in two different contexts: Making decisions with complete information, and making decisions based on incomplete (partial) information. Known the entire future, an optimal offline decision can be computed. As we do not want to make any assumptions on future prices, worst-case scenarios are of main interest. Competitive analysis deals with the question whether the decisions taken were reasonable given partial information, and calculates the ratio between the worst-case behavior of an online algorithm and the corresponding optimal algorithm on the same problem instance. This ratio, the competitive ratio, is the worst-case performance guarantee. In the context of financial markets these online algorithms are referred to as guaranteeing conversion algorithms, and the guarantee is to be determined analytically. The main application of guaranteeing conversion algorithms is the search for best prices. Here, an online investor is searching for the maximum (resp. minimum) price(s) in a sequence of prices that unfolds sequentially. Each point of time \( t \) the investor obtains a price quotation \( q_t \), after which (s)he must immediately decide whether to accept \( q_t \) or to continue observing prices. The goal is to buy at low prices and to sell at high prices with no knowledge about the future (El-Yaniv, 1998; Mohr and Schmidt, 2008; Kakade et al., 2004; Lorenz et al., 2009; Schmidt et al., 2010).

\[^5\text{A detailed literature overview on these heuristic RP algorithms is given in Chapter 3.}\]
Most authors apply guaranteeing conversion algorithms to solve the currency conversion problem (El-Yaniv et al., 1992, 2001; Iwama and Yonezawa, 1999; El-Yaniv et al., 2001; Chen et al., 2001; Kakade et al., 2004; Hu et al., 2005; Chang and Johnson, 2008; Fujiwara et al., 2011). In this problem, a fixed amount of dollars must be converted into yen, and possibly back. The goal is to compare well with any conversion algorithm; even with \( OPT \). Selected online conversion algorithms to solve this problem are presented in detail in Chapter 4 and 5.

Other applications of guaranteeing algorithms in literature are the search for jobs, and the search for employees where the goal is to choose the best position, applicant or expert (Freeman, 1983; Ferguson, 1989; Kalai and Vempala, 2005; Babaioff et al., 2008). Further, Ajtai et al. (1995) develop an algorithm to choose an appropriate sample from a population for the purpose of a study.

By the help of trading systems online conversion algorithms can be implemented, evaluated and, if promising, used for real-time trading on a stock market. An overview on trading systems is given in the following. We consider a trading system as the ‘tool’ for evaluating online conversion algorithms.

1.4 Trading Systems

In practice, a great variety of trading systems exists. Practitioners use these systems driven by a profit motive. These systems are not considered here. Details on the functionality of most important commercial trading systems available on the market can be found in Kersch and Schmidt (2011). Within the scientific community the term trading system is used in different ways:

First, the term trading system is used to describe electronically organized markets. Examples are the German XETRA market, the German XONTRO trading system, or the United States NASDAQ system. These markets mostly replaced the phone-based order flow, and are organized in the form of auctions (Kim, 2007, p. 2).

Second, the term trading system is used to describe algorithmic trading, namely computer-based algorithms, and autonomous programs to determine the market timing of orders. For example Gomber (2000, p. 28) defines an (electronic) trading system as a computer system for the electronic order specification and order routing, which enables the electronic concentration of compatible orders. These systems are mainly used by institutional investors. For example in 2009 42% of the trades on the XETRA market were submitted via algorithmic trading (Teske, 2010, p. 23). Further, Gomber et al. (2005) claim that algorithmic trading will replace as much as 90% of todays human traders within the next years.
Third, the term trading system is used to describe so-called trading machines, namely the computer-based implementation and execution of online conversion algorithms, and their corresponding orders by a software system. These machines decide whether or not to convert financial instruments in the matter of a split second. Mostly without human interference. A (electronic) trading machine is an environment where users define and adjust trading models for real-time execution, i.e. algorithms can not be evaluated using historical data (Ignatovich, 2006, p. 1).

Fourth, the term trading system is used to describe a collection of rules which are used to generate buy and sell signals including risk and money management (Vanstone and Finnie, 2009).

In contrast, within this work, collections of rules are defined as online conversion algorithm, and the term trading system indicates a software system. By the help of a trading system these algorithms can be 1) designed, e.g. using an (XML) editor, 2) simulated, e.g. on historical or artificial data, 3) evaluated, e.g. using statistical tests, and 4) executed on a stock exchange if the results are promising, e.g. via direct market-access channels. In addition, supporting functions such as charts or an information system offer the possibility to interpret historical and real-time data, known as ‘charting’.

In order to design, evaluate and execute conversion algorithms an appropriate software system – providing the desired functionality – is required. In the following, we give a brief overview on different classes of trading systems based on their functionality. In contrast, practitioners classify trading systems based on the user type (Kim, 2007, p. 119). Three classes of trading systems exist (Kersch and Schmidt, 2011):

1. An Execution System (ES) is the superordinate concept for trading systems or online brokerage systems. Execution systems are used by banks, direct banks, online banks, financial service providers, or by service providers specializing in online brokerage. With an ES the user has the possibility to generate and submit orders to be executed on the stock market. The implementation and evaluation of conversion algorithms is not supported.

2. A Planning System (PS) allows to implement and test conversion algorithms. The algorithms can be evaluated and optimized in terms of return maximization. The execution of orders and the order routing is not supported.

3. A Planning and Execution System (PES) combines the characteristic features of both ES and PS. With a PES the investor has the possibility to 1) implement, 2) evaluate, and 3) execute conversion algorithms supported...
by one single system.

Independent from its classification, a trading system should contain the following components: Graphical tools, development tools, test environment (backtesting), real-time environment (portfolio management and order management). For evaluating online conversion algorithms the development tools are essential, as they must be easy to use and, at the same time, powerful to describe complex algorithms. For that purpose, within this work, we use the LifeTrader System, a PES providing the required functionality.\(^6\)

An approach to evaluate the performance of online conversion algorithms is presented in the following: Chapter 2 introduces the notion of competitive analysis, and Chapter 3 gives the steps to empirically analyze online conversion algorithms.

References for Chapter 1


\(^6\)LifeTrader is a software system to evaluate online conversion algorithms, details can be found in Kersch and Schmidt (2011).
REFERENCES FOR CHAPTER 1


REFERENCES FOR CHAPTER 1


REFERENCES FOR CHAPTER 1


REFERENCES FOR CHAPTER 1


REFERENCES FOR CHAPTER 1


Chapter 2

Competitive Analysis of Online Conversion Algorithms

This chapter reviews fundamental concepts and results in the area of online algorithms and competitive analysis. We present the classical online problem and introduce the notion of competitive analysis mentioning the related work relevant to the specific problem. Then we focus on online algorithms for conversion problems and provide a comprehensive review of the literature addressing the existing problems. The chapter concludes with an overview on competitive search algorithms in the context of conversion in financial markets. We limit to the search for best prices in order to buy or/and sell assets.

2.1 Online and Offline Algorithms

A standard assumption in traditional optimization techniques is the complete knowledge of all data of a problem instance in advance (Borodin and El-Yaniv, 1998). However in reality, decisions often have to be made online, i.e. without knowing future data relevant for the current choice, or before complete information is available. Such scenarios are called online problem. Each decision must be made based on the already appeared data of the problem instance, and without any information about future data (Fiat and Woeginger, 1998).

Online algorithms represent the theoretical framework for solving online problems. An online algorithm computes a partial solution whenever input data requests an action. No assumptions about the input data are made. Even worse, input data may be produced by an adversary in such way that the online algorithm is always confronted with the worst possible input sequence (cf. Section 1.1). The worst possible adversary is an algorithm that always achieves an optimum solution, the optimal offline algorithm \(OPT\) (Albers, 2003).
More formally, each input can be represented as a finite sequence $I$ with $t = 1, \ldots, T$ elements, and a feasible output can also be represented as a finite sequence with $T$ elements. An algorithm computes \textit{online} if for each $t = 1, \ldots, T - 1$, it computes an output for $t$ before the input for $t + 1$ is given. An algorithm computes \textit{offline} if it computes a feasible output given the entire input sequence $I$ in advance.

An online algorithm may not produce an optimum result. It is nevertheless desired to evaluate its quality. The technique to evaluate the performance of an online algorithm is called competitive analysis and compares the performance of an online algorithm to that of an adversary, e.g. $OPT$. Within this work we consider online conversion algorithms ($ON$) — to compute a solution $ON$ must solve the online conversion problem. Thus, before introducing the notion of competitive analysis, the online conversion problem and its solutions from the literature are presented.

2.2 Online Conversion Problems

An online conversion problem deals with the scenario of converting an asset $D$ into another asset $Y$, and possibly back. As mentioned in Section 1.3 these assets can be physical, financial, or intellectual. Hence, every online conversion problem is a variant or an application of the elementary problem of optimal stopping (Chow et al., 1971). The key example of an optimal stopping problem is the well known secretory problem. In its simplest form the problem can be stated as follows (Ferguson, 1989, p. 282):

1. There is a single secretarial position to fill.
2. There are $T$ applicants for the position, and the value of $T$ is known.
3. The applicants can be ranked from best to worst with no ties.
4. The applicants are interviewed sequentially in a random order, with all $T!$ possible orders being equally likely.
5. After each interview, the applicant must be accepted or rejected.
6. The decision to accept or reject an applicant can be based only on the relative ranks of the applicants interviewed so far.
7. Rejected applicants cannot be recalled.
8. The last applicant must be accepted.
9. The payoff is 1 for selecting the best applicant and 0 otherwise.
Online Conversion Problems

Clearly, the objective is to select the best applicant. Only an applicant who, when interviewed, is better than all the applicants interviewed previously will be considered for acceptance. The optimal policy (the stopping rule) for a large number of applicants $T$ is to (interview and) reject the first $\frac{T}{e}$ applicants, and then to accept the first applicant who is better than all the rejected. The secretary problem has received much attention because the stopping rule has a surprising feature: For $T \to \infty$, the probability of selecting the best applicant from the pool goes to $\frac{1}{e}$, which is around 37%. Hence, the stopping rule picks the single best applicant in about 37% of the cases (Ferguson, 1989; Babaioff et al., 2008). Work on the problem and its extensions is reviewed in Freeman (1983); Ferguson (1989), and Ajtai et al. (1995).

In the following we limit to online conversion problems in a financial context. These problems are a special case to the theory of optimal stopping. It is assumed that $ON$ observes a sequence of $t = 1, \ldots, T$ price quotations $q_t$ and must decide which $q_t$ to pick, i.e. when to stop observing. Instead of picking the best applicant, the objective is to pick the best price(s) $q_t$ for conversion. Further, in case $ON$ picks a price $q_t$ $ON$ must specify which fraction $s_t$ of asset $D$ is to be converted into asset $Y$ at $q_t$. Depending on the possible values of $s_t$ two classes of online conversion problems exist:

**Preemptive (pmtn).** Search for more than one price in the time interval of length $T$ in order to convert asset $D$. $ON$ is allowed to convert sequentially in parts at different prices $q_t$, i.e. the whole amount available is converted ‘little by little’, and $s_t \in [0, 1]$. Typically, the number of prices considered for conversion is determined by $ON$. Except in one special case where $ON$ desires to convert at a specific number of prices, denoted by $u$. This is referred to as $u$-preemptive ($u$-pmtn). In the work related algorithms for preemptive conversion are denoted as constant rebalancing algorithms or threat-based algorithms (cf. Section 2.4.2).

**Non-preemptive (non-pmtn).** Search for one single price in the time interval of length $T$ in order to convert asset $D$. $ON$ is allowed to convert ‘all or nothing’, i.e. the whole amount available is converted at one price $q_t$, and $s_t \in \{0, 1\}$. In the work related algorithms for non-preemptive conversion are denoted as reservation price algorithms (cf. Section 2.4.1). Non-preemptive conversion is a special case of preemptive conversion.

Preemptive as well as non-preemptive algorithms solving the online conversion problem either aim cost minimization or profit maximization, or both. Stated this way, the problem is very similar to the famous *secretary problem:* Designing an
algorithm for picking an element out of a (ordered) sequence, in order to maximize the probability of picking the ‘best’ element of the entire sequence (Awerbuch et al., 1996). In the finance related literature three main fields of application solving this problem can be found: 1) Replacement problems, 2) investment planning, and 3) the search for best prices. In the following we state each problem in short and give a brief literature overview.\(^7\)

1) Replacement Problem. In the basic setup of this problem some equipment is needed during an unknown number of time intervals. How long the equipment is needed is made known online: At the start of each time interval \(ON\) gets the information whether the equipment will be needed in the current time interval or not. \(ON\) must immediately decide whether to buy the equipment for a price \(q_b\) or to rent it for a price \(q_r\), with \(q_r < q_b\). The ‘game’ ends with the purchase of the equipment, or if the equipment is no longer needed. The total cost incurred by algorithm \(ON\) is the sum of all renting fees, and perhaps one purchase. The goal is to choose the optimal point of time for buying (El-Yaniv and Karp, 1997, p. 815). The optimal decision must be determined such that the ratio of the money which was spent for the equipment (\(q_r\) and \(q_b\)), and the minimum money which had to be spent is minimized. The solution of the replacement problem is to rent until the period of amortization ends, and to buy then. Karp (1992a,b) shows that in practice people buy equipment earlier than this optimal point, or keep renting forever. Typical practical applications addressed in the literature are ski-rental (Karlin et al., 1994; al-Binali, 1997; El-Yaniv et al., 1999; Seiden, 2000; Fujiwara and Iwama, 2002), selling a car (Babaioff et al., 2008), and buying a BahnCard\(^8\) (Fleischer, 2001; Ding et al., 2005). For a detailed review on the problem and its extensions the reader is referred to El-Yaniv and Karp (1997) and El-Yaniv et al. (1999). The replacement problem is not discussed here.

2) Investment Planning. In the basic setup of this problem an algorithm \(ON\) must decide how to reallocate among different available investment opportunities; e.g. assets, commodities, securities, and their derivatives. The value of each investment opportunity changes from time interval

---

\(^7\)Some authors state a fourth main field called leasing problems, e.g. algorithms to decide whether to buy or lease a car. Those problems are considered as rudimentary forms of replacement problems (El-Yaniv, 1998, p. 30).

\(^8\)A BahnCard is a loyalty card offered by Deutsche Bahn AG, the German national railway company. It entitles the passenger to a discount price, and must be purchased prior to travel; see [www.bahn.de](http://www.bahn.de)
Online Conversion Problems

27
to time interval in an uncertain manner. The goal is to maximize the terminal wealth (Cover, 1991). Typical applications in literature are ‘universal portfolios’ proposed by Cover (1991), and later studied in Cover and Ordentlich (1996); Helmbold et al. (1998); Blum and Kalai (1999); Cover and Ordentlich (1998); Kalai and Vempala (2003) and Agarwal and Hazan (2006). In this setting the goal is to design an online algorithm running an ‘universal portfolio’ that is competitive against any constant rebalancing portfolio which keeps the same distribution of wealth among a set of assets from day to day. In this regard other (non-universal) online portfolio selection algorithms are presented by Cover and Gluss (1986) and Borodin et al. (2000, 2004). Option pricing (Lorenz et al., 2009; DeMarzo et al., 2006) and asset allocation (Raghavan, 1992) are further fields. The investment planning problem is not discussed here.

3) Search for Best Prices. In the basic setup of this problem ON is given the task of converting an asset into another asset, and possibly back. The goal is to convert at best prices, i.e. to search for the maximum (resp. minimum) price in a sequence of prices that unfolds sequentially (El-Yaniv, 1998; Kakade et al., 2004; Lorenz et al., 2009; Schmidt et al., 2010). Thus, converting assets is a direct application of the elementary problem of optimal stopping. Consider ON must convert an asset D into another asset Y, and starts with the initial amount $d_0 = 1$ ($y_0 = 0$) of asset $D$ ($Y$). In its simplest form, an online conversion algorithm solving search for best prices can be stated as follows.

**Algorithm 1.**

*Step 1:* Obtain price quotations $q_t \in [m, M]$ at points of time $t = 1, \ldots, T$.

*Step 2:* Every point of time $t$ take a decision whether or not to accept the current price $q_t$.

*When*

*Step 2a:* Price $q_t$ is accepted convert an amount $s_t$ of asset D into Y.

*Step 2b:* Price $q_t$ is not accepted, obtain the next price quotation $q_{t+1}$.

*Step 2c:* Asset D is converted completely, or $T$ is reached, the ‘game’ ends.

*Step 3:* If there is some amount of D left on $T$ then accept the last price $q_T$ (which might be the worst-case, i.e. $m$ for selling or $M$ for buying).
Some authors assume ON must pay a commission to get a price quotation, called sampling costs (El-Yaniv, 1998, p. 33). Further, the search for best prices is often considered as currency conversion, or as elementary search problem (El-Yaniv, 1998, p. 32). Several authors suggest algorithms to solve the currency conversion problem, cf. El-Yaniv et al. (1992, 2001); al-Binali (1997, 1999); Iwama and Yonezawa (1999); Chou et al. (1995); Chen et al. (2001); Kakade et al. (2004); Hu et al. (2005); Chang and Johnson (2008) and Fujiwara et al. (2011). In this problem, a fixed amount of dollars must be converted into yen, and possibly back. The goal is to perform well under worst-case assumptions, i.e. to achieve a possibly low competitive ratio $c$.

Within this work we limit to online conversion algorithms solving the search for best prices. The work related addresses on the one hand algorithms that aim profit maximization, denoted as max-search problem, or cost minimization, denoted as min-search problem. These algorithms are uni-directional. On the other hand, algorithms are addressed that aim return maximization solving both problems. These algorithms are bi-directional (El-Yaniv et al., 2001). A short overview on uni- and bi-directional search problems addressed in the literature is given in the following.

### 2.2.1 Uni-directional Search

Uni-directional search assumes that within one time interval conversion can only be performed in one direction. When carrying out uni-directional search to solve the online conversion problem, the objective is always to choose a point of time to take a decision, in order to maximize an expected profit or to minimize an expected cost, but never both (Kalai and Vempala, 2005). Hence, the resulting min-search problem or max-search problem is considered as uni-directional (or one-way) (El-Yaniv et al., 2001, p. 101).

**Uni-directional Search.** Here, ON is given the task of converting an asset $D$ into another asset $Y$ within a given time interval in order to achieve financial gain. The conversion back from $Y$ into $D$ is forbidden. To convert $D$ back into $Y$ a new ‘search game’ must be carried out. The classical example of uni-directional search is currency conversion, e.g. converting dollars $D$ into yen $Y$: $ON$ may convert $D$ into $Y$ as often as possible (at different prices $q_t$) until the whole of asset $D$ is converted into $Y$. There is no restriction on the number of conversions, and conversion can either be preemptive or non-preemptive. In other words, $ON$ searches for the maximum or the minimum price(s) in order to carry out either a buying or a selling transaction.
within one time interval of length $T$. A transaction is completed when the whole of asset $D$ is converted into $Y$.

Some authors consider randomized online search as uni-directional search. The goal is also to convert $D$ into $Y$. It is assumed that the price (or the exchange rate from $D$ to $Y$) varies unpredictably (El-Yaniv, 1998; El-Yaniv et al., 2001; Chen et al., 2001). The transformation of randomized online search to uni-directional search is as follows (Damaschek et al., 2009, p. 620): The initial amount of $D$, denoted by $d_0$, corresponds to a probability of 1. Converting $d_0$ means to stop converting with exactly that probability (randomized online search). Thus, any uni-directional search algorithm is equivalent to a randomized search algorithm that converts the entire $d_0$ at once (non-preemptive) at some randomly chosen price, cf. Borodin and El-Yaniv (1998, p. 265) and El-Yaniv (1998, p. 36).

Algorithms to solve the uni-directional search problem are suggested by El-Yaniv et al. (1992, 2001); El-Yaniv (1998); al-Binali (1997, 1999); Iwama and Yonezawa (1999); Chen et al. (2001); Kakade et al. (2004); Hu et al. (2005); Chang and Johnson (2008); Fujiwara et al. (2011). An experimental analysis of the uni-directional algorithms of El-Yaniv (1998); El-Yaniv et al. (2001) assuming different settings, such as dividing the investment horizon into time intervals, can be found in Schmidt et al. (2010).

In case min-search and max-search are combined bi-directional search is carried out. A short overview on bi-directional search problems is given in the following.

### 2.2.2 Bi-directional Search

Bi-directional search assumes that within one time interval conversion can be performed in both directions. When carrying out bi-directional search to solve the online conversion problem, the objective is to achieve a possibly high return. When converting assets, uni-directional search is extended to bi-directional search, and bi-directional search is a synonym for trading.

**Bi-directional Search.** Here, $ON$ is given the task of converting an asset $D$ back and forth. Converting asset $D$ into asset $Y$, then back into asset $D$, and back into asset $Y$, etc. is allowed within the same time interval. The relative price between $D$ (resp. $Y$) and $Y$ (resp. $D$) is used to determine the units converted, and thus becomes the exchange rate. There is no restriction on the number of conversions, conversion can either be preemptive or non-preemptive. In contrast to uni-directional search $ON$ searches for maximum and minimum prices to carry out both a buying and a selling transaction within one time interval of
length $T$. Chou et al. (1995); Dannoura and Sakurai (1998); El-Yaniv et al. (1992, 2001); Mohr and Schmidt (2008a) suggest algorithms to solve the bi-directional search problem under various limitations. The classical example of bi-directional search is currency conversion converting dollars $D$ into yen $Y$ and back as often as possible.

**Run Search.** A special case of bi-directional search. Here, $ON$ is also given the task of converting an asset $D$ back and forth as often as possible. But when carrying out run search, the algorithm $ON$ divides the considered sequence of prices into *upward runs* and *downward runs* depending on the price movement. Search is carried out depending on the direction of the runs: *Max-search* is carried out if prices are moving up, and *min-search* is carried out if prices are moving down. In other words, uni-directional search is carried out depending on the direction of a run, and each run equals one time interval of length $T$. Dannoura and Sakurai (1998); El-Yaniv et al. (1992, 2001); Damaschke et al. (2009) suggest algorithms to solve the run search problem.

Irrespective whether an algorithm converts preemptive or non-preemptive, uni-directional or bi-directional it may not produce an optimum result. Hence, it is desired to evaluate its effectiveness, e.g. against the performance of another algorithm for the same problem. This technique is called competitive analysis. In the following we introduce notion of competitive analysis as a performance measure for online conversion algorithms investigating worst-case scenarios.

### 2.3 Competitive Analysis

Firstly, competitive analysis was used in the 1970s by computer scientists in connection with approximation algorithms for $NP$-hard problems (Graham, 1966; Johnson, 1973; Johnson et al., 1974; Yao, 1980). In 1985, the work of Sleator and Tarjan (1985), on list access and paging algorithms, put forth the use of the competitive ratio as a general performance measure for online decision making. Three years later, the term *competitive ratio* was formed by Karlin et al. (1988).\(^9\) The main idea is to assume the worst possible input sequence $I$, and to compare the performance of an online algorithm to the performance of an adversary on this sequence. The competitive ratio $c$ measures the quality of the online algorithm with respect to the adversary. Within the scope of this work, unless otherwise stated,

---

\(^9\)In the literature, the competitive ratio is also called the worst-case ratio or the worst-case performance guarantee (Fiat and Woeginger, 1998, p. 4).
the performance of $ON$ is always compared to the worst possible adversary: $OPT$ computes an output given the entire input sequence $I$ in advance. $ON$ is called $c$-competitive if for any $I$ (El-Yaniv et al., 2001, Formula (1))

$$ON(I) \geq \frac{1}{c} \cdot OPT(I).$$

(2.1)

In other words, $ON$ is called strictly $c$-competitive, if its competitive ratio – the ratio between the performance of $ON$ and $OPT$ – is bounded by some constant $c$, which gives a worst-case performance guarantee (Albers, 2003). We want to remark that the definition of $c$-competitiveness varies in the literature. $ON$ is called weakly $c$-competitive if there exists a constant $z$ such that (Karlin et al., 1994, p. 302)

$$ON(I) \geq \frac{1}{c} \cdot OPT(I) + z$$

(2.2)

holds for any input sequence $I$. Some authors even allow $z$ to depend on problem or instance specific parameters (Albers, 1997; Krumke, 2002). We assume the constant $z$ to be zero and will stick to the definition given in equation (2.1). Hence, any $c$-competitive $ON$ is guaranteed a value of at least the fraction $\frac{1}{c}$ of the optimal offline result, no matter how uncertain the future will be (El-Yaniv et al., 2001, p. 104). This holds for bounded problems (El-Yaniv, 1998).

We consider online conversion algorithms with bounded profit function, e.g. by assuming $q_t \in [m, M]$, where $M$ and $m$ are upper and lower bounds of prices $q_t$. Further, we differ between the competitive ratio for uni-directional search, and the competitive ratio for bi-directional search. Algorithms denoted as uni-directional only convert in one direction (asset $D$ into asset $Y$). Thus, their competitive ratio is measured by the amount of (accumulated) $Y$ achieved on the last day $T$. Algorithms denoted as bi-directional convert in both directions (asset $D$ into asset $Y$, and back to $D$). Thus, their competitive ratio is measured by the amount of (accumulated) $D$ achieved on the last day $T$.

### 2.3.1 Competitive Ratio for Uni-directional Search

We assume $ON$ is either allowed to carry out a selling or a buying transaction within each $i$-th time interval of length $T$ ($i = 1, \ldots, p$). Overall, within the whole investment horizon, $ON$ is allowed to carry out $p \geq 1$ buying or selling transactions, solving either the min-search problem or the max-search problem. The performance of $ON$ is measured using the competitive ratio as given in equation (2.1).

**Min-Search.** To minimize costs the min-search problem must be solved in order to buy at a possibly low price(s). Assume $ON$ buys $p \geq 1$ times at price(s)
\[ q^{\text{min}}(i) \geq m(i) \geq m \text{ with } i = 1, \ldots, p. \]

Solving equation (2.1) to compute the competitive ratio for each \( i \)-th buying transaction equals

\[
\begin{align*}
    c^{\text{min}}(i) &= \frac{\text{OPT}}{\text{ON}} \\
    &= \frac{m(i)}{q^{\text{min}}(i)} \\
    &\leq 1,
\end{align*}
\]

and results in an overall competitive ratio after the \( p \)-th buying transaction of

\[
\begin{align*}
    c^{\text{min}}(p) &= \prod_{i=1}^{p} \frac{m(i)}{q^{\text{min}}(i)} \\
    &\leq 1.
\end{align*}
\]

Assuming \( q^{\text{min}}(i) = q^{\text{min}} \) and \( m(i) = m \) to be constants for each \( i \)-th buying transaction the overall competitive ratio (after the \( p \)-th transaction) then equals

\[
\begin{align*}
    c^{\text{min}}(p) &= \left( \frac{m}{q^{\text{min}}} \right)^p \\
    &\leq 1.
\end{align*}
\]

As buying is a minimization problem \( c^{\text{min}}(p) \leq 1 \), and measures the competitive ratio for buying under worst-case assumptions. The greater \( c \) the more effective is \( \text{ON} \).

**Max-Search.** To maximize profit the *max-search problem* must be solved in order to sell at a possibly high price. Assume \( \text{ON} \) sells \( p \geq 1 \) times at possibly high prices \( q^{\text{max}}(i) \leq M(i) \leq M \) with \( i = 1, \ldots, p \). Solving equation (2.1) to compute the competitive ratio for each \( i \)-th selling transaction then equals

\[
\begin{align*}
    c^{\text{max}}(i) &= \frac{\text{OPT}}{\text{ON}} \\
    &= \frac{M(i)}{q^{\text{max}}(i)} \\
    &\geq 1,
\end{align*}
\]

and results in an overall competitive ratio after the \( p \)-th selling transaction of

\[
\begin{align*}
    c^{\text{max}}(p) &= \prod_{i=1}^{p} \frac{M(i)}{q^{\text{max}}(i)} \\
    &\geq 1.
\end{align*}
\]

Assuming \( q^{\text{max}}(i) = q^{\text{max}} \) and \( M(i) = M \) to be constants for each \( i \)-th selling transaction the overall competitive ratio (after the \( p \)-th transaction) then equals

\[
\begin{align*}
    c^{\text{max}}(p) &= \left( \frac{M}{q^{\text{max}}} \right)^p \\
    &\geq 1.
\end{align*}
\]
As selling is a maximization problem $c_{\text{max}}(p) \geq 1$, and measures the competitive ratio for selling under worst-case assumptions. The smaller $c$ the more effective is ON.

In the above section it is assumed that ON either buys $p$ times at possibly low prices or sells $p$ times at a possibly high prices ($p \geq 1$), resulting in the worst-case competitive ratios given in equation (2.4) and (2.7). To trade assets $p \geq 1$ times sequentially in a row this assumption does not hold. In the context of financial markets online conversion algorithms are designed to buy and sell (trade) in order to achieve a possibly high return. We assume each trade consists of exactly one buying transaction and one selling transaction. In other words, first the min-search problem has to be solved for buying, and later the max-search problem has to be solved for selling, resulting in $p$ trades (equaling the number of returns). Thus, instead of using maximum or minimum prices, the competitive ratio for bi-directional search is calculated using the returns achieved by OPT and ON.

### 2.3.2 Competitive Ratio for Bi-directional Search

We assume ON is allowed to carry out more than one buying and selling transaction within each $i$-th time interval of length $T$ ($i = 1, \ldots, p$). Further, we assume each $i$-th time interval is initiated by a buying transaction, and terminated by a selling transaction. Hence, within the whole investment horizon overall $p$ trades, equaling the number of time intervals, are carried out. Thus, the competitive ratio for bi-directional search measures the performance of ON in terms of the achieved return, when carrying out $p \geq 1$ trades. Online conversion algorithms are either designed to trade once ($p = 1$), or to trade sequentially in a row ($p > 1$), defined as follows:

**Single Bi-directional Conversion.** Within $T$ an asset is traded exactly once. Thus, the objective is to buy one single asset at best at its minimum price $q^{\text{min}} \geq m$, and to sell it later at best at its maximum price $q^{\text{max}} \leq M$.

**Multiple Bi-directional Conversion.** Within $T$ an asset is traded more than once. The objective is to trade $p > 1$ times sequentially in a row: Buy an asset $p > 1$ times at local minimum prices $q^{\text{min}}(i) \geq m(i) \geq m$, and sell it $p > 1$ times at local maximum prices $q^{\text{max}}(i) \leq M(i) \leq M$, where $i = 1, \ldots, p$ buying transactions and $i = 1, \ldots, p$ selling transactions are carried out. Further, the single asset problem trading one single asset $p > 1$

\footnote{Short-selling is not considered here as it is forbidden in some countries, e.g. in Germany since May 19th, 2010.}
times, and the *multiple asset problem* trading several different assets \( p > 1 \) times can be distinguished.

For both variants the calculation of the competitive ratio is identical. Let \( X \in \{OPT, ON\} \) be a bi-directional conversion algorithm. Assume the algorithms \( X \) trade sequentially in a row, and each \( i \)-th trade consists of one buying and one selling transaction with \( p \geq 1 \), and \( i = 1, \ldots, p \). Further assume algorithm \( X \) buys \( p \geq 1 \) times at a possibly low price(s) \( q_{\text{min}}^{\text{max}}(i) \geq m(i) \), and sells at possibly high price(s) \( q_{\text{max}}^{\text{max}}(i) \geq m(i) \). Then the return of \( X \) for each \( i \)-th trade with \( i = 1, \ldots, p \) equals

\[
R_X(i) = \frac{q_{\text{max}}(i)}{q_{\text{min}}(i)},
\]

and results in an overall return after the \( p \)-th trade of

\[
R_X(p) = \prod_{i=1}^{p} \frac{q_{\text{max}}(i)}{q_{\text{min}}(i)}. \tag{2.10}
\]

Note that \( ON \) solving the bi-directional conversion problem in order to maximize the return to be expected \( \mu \) is called *money-making* if it is guaranteed to be profitable when \( OPT \) is profitable, i.e. the achieved return \( R_X(p) > 1 \) (Chou et al., 1995, p. 469).

The overall competitive ratio for bi-directional conversion \( c(p) \) with \( p \geq 1 \) can be derived in two ways. First, the competitive ratio for \( \text{min-search} \) and \( \text{max-search} \), as given in Section 2.3.1, can be used. For each \( i \)-th trade from equation (2.3) and (2.6) we get

\[
c(i) = \frac{c_{\text{max}}(i)}{c_{\text{min}}(i)} \geq 1,
\]

resulting in an overall competitive ratio

\[
c(p) = \prod_{i=1}^{p} \frac{c_{\text{max}}(i)}{c_{\text{min}}(i)} \geq 1.
\]

\[
= \prod_{i=1}^{p} \left( \frac{M(i)}{q_{\text{max}}^{\text{max}}(i)} \cdot \frac{q_{\text{min}}(i)}{m(i)} \right)
\]

\[
\geq 1.
\]
Assuming $q^{\text{max}}(i), q^{\text{min}}(i), M(i)$ and $m(i)$ to be constants from equations (2.3) and (2.6) we get the overall competitive ratio after the $p$-th trade

\[
c(p) = \frac{c^{\text{max}}(p)}{c^{\text{min}}(p)} = \left( \frac{M}{q^{\text{max}}(i)} \cdot \frac{q^{\text{min}}}{m(i)} \right)^p \geq 1.
\]

Second, the overall returns $R_X(p)$ achieved by $X \in \{\text{OPT, ON}\}$ as given in equation (2.10) can be used to calculate $c(p)$. Assuming $p \geq 1$ the overall return $R_{ON}(p)$ of an algorithm $ON$ equals

\[
R_{ON}(p) = \prod_{i=1}^{p} \frac{q^{\text{max}}(i)}{q^{\text{min}}(i)},
\]

and the overall return $R_{OPT}(p)$ of algorithm $OPT$ equals

\[
R_{OPT}(p) = \sup R_{ON}(p) = \prod_{i=1}^{p} \frac{M(i)}{m(i)}.
\]

In case $M(i) = M$ and $m(i) = m$ are constants the overall return of $OPT$ equals (Mohr and Schmidt, 2008a)

\[
R_{OPT}(p) = \left( \frac{M}{m} \right)^p.
\]

Assuming and identical number of $p \geq 1$ trades for $OPT$ and $ON$ from equation (2.14) and (2.15) we get an overall competitive ratio

\[
c(p) = \frac{OPT}{ON} = \frac{R_{OPT}(p)}{R_{ON}(p)} = \prod_{i=1}^{p} \left( \frac{M(i)}{m(i)} \cdot \frac{q^{\text{min}}(i)}{q^{\text{max}}(i)} \right) = \prod_{i=1}^{p} \frac{c^{\text{max}}(i)}{c^{\text{min}}(i)}.
\]

### 2.3.3 Worst-case and Empirical-case Competitive Ratio

When analyzing online conversion algorithms we differ between the worst-case competitive ratio $c^{wc}$ considering the performance of $ON$ on a worst possible
sequence of inputs, and the empirical-case competitive ratio $c_{ec}$ considering the performance of $ON$ on an observed time series of prices. Assuming $p \geq 1$ trades both ratios can be calculated using equation (2.17). To calculate $c_{wc}$ a constructed worst-case time series of prices is considered and the return of $ON$ is derived analytically. In contrast, to calculate $c_{ec}$ an observed time series of prices is considered, and the return of $ON$ is derived experimentally through backtesting. Thus, the worst-case competitive ratio $c_{wc}(p)$ for $p \geq 1$ equals

$$c_{wc}(p) = \sup c(p).$$

(2.18)

In the worst-case $ON$ might, for example, buy $i$ times at the highest possible price $M(i)$, and sell $i$ times at the lowest possible price $m(i)$.

Further, the empirical-case competitive ratio $c_{ec}(p)$ for $p \geq 1$ equals

$$c_{ec}(p) = \frac{R_{OPT}(p)}{R_{ON}(p)}$$

(2.19)

where $OPT$ achieves the best possible return $OPT = \frac{M(i)}{m(i)}$ on the time series considered, and $ON$ achieves a return according to the buying and selling signals generated. Note that $c_{ec}(p) \leq c_{wc}(p)$, and the best achievable $c \in \{c_{wc}(p), c_{ec}(p)\}$ equals $1$.

In the following we give an overview on online conversion algorithms analyzed using competitive analysis – in terms of $ON$ ‘playing’ against an adversary while considering worst-case scenarios. Typically, these reviewed online conversion algorithms are categorized as reservation price algorithms, constant rebalancing algorithms, threat-based algorithms, and risk-rewarded algorithms. For the literature overview, we present a new approach to classify online conversion algorithms based on the type of search (uni-directional or bi-directional), and the amount to be converted ($pmtn$ or $non-pmtn$). Within Chapter 6 this classification is refined by the ‘amount of information’ assumed to be known a-priori (about the future) to $ON$ in order to compute the amount to be converted $s_t$.

2.4 Literature Review

We give a literature overview of work on online conversion problems, focusing on worst-case performance measures as given in equation (2.18). As we are interested in online algorithms related to financial decision making we restrict the literature overview to algorithms in the context of financial markets, solving the search for best prices as given in Algorithm 1 in order to convert assets. The majority of the work related considers online conversion problems in Forex Markets.\(^{11}\)

\(^{11}\text{Foreign exchange market; a worldwide decentralized over-the-counter financial market for the trading of currencies, also denoted as FX or currency market.}\)
We do not consider related applications like algorithmic trading and online auctions. The reader is referred to Kleinberg (2005); Blum et al. (2006) and Chang and Johnson (2008).

Based on the amount to be converted $s_t$, when presenting the work related, we distinguish the two classes of online conversion algorithms: a) non-preemptive online conversion algorithms – designed to search for one single price within the time interval to convert the asset, and b) preemptive online conversion algorithms – designed to search for more than one price within the time interval to convert the asset.

### 2.4.1 Non-Preemptive Conversion

Non-preemptive conversion allows the search for one single price in the time interval to convert an asset $D$. Typically, the whole amount available is converted at one single price $q_t$, i.e. $s_t \in \{0, 1\}$. Non-preemptive algorithms define limit price(s) (the market participant is willing to accept) to avoid buying or selling at a price higher (lower) than a specific level. That is the lowest price (per asset) an algorithm might accept for buying, and the highest price an algorithm might accept for selling. Such limit prices are denoted as reservation prices (RP), denoted by $q^*$. As a non-preemptive algorithm converts ‘all or nothing’ one $q_t \geq (\leq) q^*$ must be accepted within one time interval. Thus, the online conversion algorithms presented in the following are denoted as RP algorithms. We differ between works on uni-directional search and bi-directional search.

#### 2.4.1.1 Uni-directional Search

In the following non-preemptive conversion algorithms for uni-directional search are presented. Here an algorithm on is allowed to convert an asset $D$ into another asset $Y$ but conversion back to $D$ is forbidden. Unfortunately, the work related is limited to guaranteeing conversion algorithms – the performance of the RP algorithms is evaluated using competitive analysis.

The two early works of Pratt et al. (1979) and Rosenfield and Shapiro (1981) assume different price distributions, and study the question when an RP algorithm should stop searching for a lower (higher) price.

Pratt et al. (1979) assume two cases. First, it is assumed that the underlying price distribution is known. Second, no knowledge is assumed, and the underlying distribution must be learned by the RP algorithm while observing prices. Pratt et al. (1979) develop RP algorithms to decide whether to observe further price quotations or not. The goal is to balance the chance of achieving a
lower (higher) price against greater incurred constant search costs, and to find a buyer-to-seller price equilibrium.

Rosenfeld and Shapiro (1981) determine search policies in case of incomplete information. Different assumptions on the a-priori knowledge about the future are made, e.g. that the price distribution is known or unknown to the RP algorithm, or itself is a random variable. Further, the RP algorithm is either allowed to accept prices previously quoted (recall) or not (no recall). Rosenfeld and Shapiro (1981) derive conditions under which the following reservation price policy (RPP) is optimal: \textit{Accept a price for buying if and only if it is below the RP.} The goal is to find an equilibrium distribution of prices (Rosenfeld and Shapiro, 1981, p. 190).

Awerbuch et al. (1996) assume the following setting: An RP algorithm must choose one out of $J$ assets for conversion. The goal is to pick a ‘winner’ that will have the best future performance. This task is made difficult by the constraint that the RP algorithm has no way to predict the future performance of any of the $J$ assets. The decision is irreversible, once an asset is chosen search is closed. For each asset $j$ ($j = 1, \ldots, J$) the value $d(j, i)$ is the number of dividends issued by asset $j$ within the $i$-th time interval. The suggested RP algorithm is: \textit{At the $(i+1)$-th time interval choose the $j$-th asset with probability $\rho^{(3d(j,i)))/(r-2)}$. Where $r$ is the a-posteriori performance (in terms of the return achieved) of the best asset, and assumed to be known. Awerbuch et al. (1996) find that their proposed RP algorithm can pick a winner with high probability.

El-Yaniv (1998) (and El-Yaniv et al. (2001)) assume that the upper and lower bounds of prices, $M$ and $m$, are known. An RP algorithm is suggested to solve the \textit{max-search problem} (El-Yaniv et al., 2001, p. 107): \textit{Accept the first price greater than or equal to $q^* = \sqrt{(M \cdot m)}$ for selling.} El-Yaniv et al. (2001) prove that if the prices $q_t \in [m,M]$ the RP algorithm is optimal, and the competitive ratio is $\sqrt{M/m}$. The RP algorithm is presented in detail in Section 4.1.

The original RP algorithm of El-Yaniv (1998) was modified by Kakade et al. (2004) and Chang and Johnson (2008) to solve the \textit{max-search problem} in modern financial markets considering the ‘Volume Weighted Average Price’ (VWAP) and limit order books (markets). Both authors assume that the price fluctuation ratio $\varphi = M/m$ is known. The modified RP algorithm places sell orders in order to maximize the total return (Chang and Johnson, 2008, p. 45): \textit{Pick an integer $i$ uniformly at random between 0 and $\lfloor \ln \varphi \rfloor$, and place an order to sell the asset at reservation price $q^* = e^i \cdot q_{min}$.} In addition Kakade et al. (2004) suggest a second RP algorithm that seeks to sell all assets at the average price of the market, the VWAP. Kakade et al. (2004) and Chang and Johnson (2008) make no assumptions on the price distribution.
Xu et al. (2011) present two RP algorithms. The first algorithm is based on the assumption that \( m \) and \( M \), as well as the return function \( f(q_t) \) are known. The second RP algorithm is based on the knowledge of \( m, M, f(q_t) \), and \( T \). The model extends the RP algorithm of El-Yaniv (1998) by introducing sampling costs for observing prices \( q_t \). It is assumed that the achievable return \( r \) when accepting a price \( q_t \) on day \( t \) is not exactly the price itself, but a function of the price (such as accepted price \( q' \) minus the accumulated sampling costs). In contrast to the RP algorithm of El-Yaniv (1998) the considered RP is not constant but varies with time, and thus is denoted by \( q^*_t \). After the player accepts one specific price \( q' \) the ‘game’ ends. It is assumed that a larger price results in a larger return \( r' \) for \( q' \). Further, the achieved return \( r' \) is higher when accepting \( q' \) earlier, as less sampling costs occur. Xu et al. (2011) present two provable optimal RP algorithms, and competitive analysis is done.

Recent work extends the algorithms for uni-directional search of El-Yaniv et al. (2001); El-Yaniv (1998) assuming that every two consecutive prices are interrelated. The motivation of Zhang et al. (2010) is the stock market in China, which empirically shows a bounded movement by 10% of every two interrelated closing prices.

Damaschke et al. (2009) assume \( M \) and \( T \) are known and prices \( q_t \in [\frac{M}{T}, M] \). A RP algorithm for max-search is presented: Accept the first price greater than or equal to \( q^* = \frac{M}{\sqrt{T}} \), with \( t = 1, \ldots, T \). Numerical examples are presented showing that the RP algorithm achieves a better (smaller) competitive ratio than previous algorithms. Damaschke et al. (2009) prove the optimality of their RP algorithm, and show that the competitive ratio equals \( \sqrt{T} \).

### 2.4.1.2 Bi-directional Search

In the following non-preemptive conversion algorithms for bi-directional search are presented. Here, \( ON \) is allowed to convert asset \( D \) into asset \( Y \), and back into \( D \) within \( T \). The work related is comprised of guaranteeing as well as heuristic RP algorithms.

**Guaranteeing Algorithms.** In the following we give a brief overview on guaranteeing RP algorithms from the literature using the competitive ratio as performance measure.

Kao and Tate (1999) consider online difference maximization, and do not make any assumptions regarding knowledge about the future. Low prices and high prices are selected from a sequence of prices in a random order by the following RP algorithm: A price is selected as low (high) if it is less (greater) than or equal
to a predefined lower (upper) bound \( m(M) \). If no price is chosen before the last day, the last price \( q_T \) must be accepted. The goal is to maximize the difference in final ranks (the expected gain) of the selected low/high price pairs (Kao and Tate, 1999, p. 88). Single and multiple conversion problems are considered. In case of single conversion one high/low pair must be chosen. In case of multiple conversion the selection of arbitrarily many high/low pairs is possible. When proving the optimality of their \( RP \) algorithm Kao and Tate (1999) assume that the inputs (prices) come from a probabilistic source such that all inputs are equally likely. Kao and Tate (1999) prove the optimality of their \( RP \) algorithm, and show that for single (multiple) pair selection the competitive ratio equals \( 1 \left( \frac{4}{3} \right) \).

Mohr and Schmidt (2008a,b) extended the uni-directional \( RP \) algorithm for selling of El-Yaniv (1998) to buying and selling, i.e. introduce a rule for min-search. The resulting bi-directional \( RP \) algorithm is: *Buy the asset at the first price smaller than or equal to, and sell the asset at the first price greater than or equal to reservation price \( q^* = \sqrt{(M \cdot m)} \).* It is shown that, in terms of achieved return, the competitive ratio \( c(i) = \frac{m(i)}{M(i)} \) for each \( i \)-th trade with \( i = 1, \ldots, p \). In addition to worst-case analysis, empirical-case analysis of the suggested \( RP \) algorithm is done assuming different settings, such as dividing the investment horizon into time intervals of different length \( T \). The original reservation price algorithm suggested by El-Yaniv (1998) and its extension by Mohr and Schmidt (2008a,b) is presented in detail in Section 4.1.

**Heuristic Algorithms.** A large number of practitioners uses heuristic conversion algorithms as their main method to determine buying and selling points using reservation prices (Taylor and Allen, 1992). The performance of these \( RP \) algorithms is usually evaluated through experiments (cf. Chapter 1). We limit to two heuristic conversion algorithms suggested by Brock et al. (1992), namely *Moving Average Crossover (MA)* and *Trading Range Breakout (TRB)*, which are based on technical indicators. These bi-directional algorithms are of major interest in the literature, and the comparison to a passive buy-and-hold (\( BH \)) algorithm is of prime interest. Brock et al. (1992, p. 1736) distinguish two variants of the \( MA \) algorithm, namely Variable-length Moving Average (\( VMA \)) and Fixed-length Moving Average (\( FMA \)). Both variants *buy if the short MA crosses the long MA from below, and sell if the short MA crosses the long MA from above.* Let \( MA(S)_t \) be a short moving average, and \( MA(L)_t \) a long moving average \( (S < L) \). The value \( n \in \{ S, L \} \), with \( t > n \), defines the number of previous data points (days) used to calculate \( MA(n)_t = \frac{\sum_{i=t-n+1}^{t} q_i}{n} \). The algorithms \( VMA \) and \( FMA \) differ in the way their performance is measured: In case of \( VMA \) every signal is considered, i.e. after a sell signal the \( RP \) algorithm goes out of the market or takes a short
position (Brock et al., 1992, p. 1738, b.8). In case of FMA fixed T-day time intervals following a buy (sell) signal are defined where \( T = 10 \) (Brock et al., 1992, p. 1740, t.3). Other signals during these T-day time intervals are ignored, i.e. in case of a buying signal a T-day long position is taken, and in case of a selling signal a T-day short position (Brock et al., 1992, p. 1736, t.11). In other words, FMA only carries out *min-search*. Brock et al. (1992) suggested different variants \((S,L)\) of the MA algorithm: \((1,50), (1,150), (5,150), (1,200)\) and \((2,200)\). Further prices might be lagged by a band \( \delta \in [0.00, \infty] \).

The TRB algorithm *buys if the price cuts the local maximum price from below, and sells if the price cuts the local minimum price from above* (Brock et al., 1992, p. 1736, t.20). The performance of TRB is calculated for fixed T-day time intervals following a buy (sell) signal, where \( T = 10 \) (Brock et al., 1992, p. 1742, b.7). Similar to FMA other signals during the T-day time intervals are ignored. Local minimum prices \( q_{t}^{\text{min}}(n) = \min \{ q_{i} | i = t-n, \ldots, t-1 \} \) and maximum prices \( q_{t}^{\text{max}}(n) = \max \{ q_{i} | i = t-n, \ldots, t-1 \} \) are calculated over the past \( n \in \{50, 150, 200\} \) days. Further prices might be lagged by a band \( \delta \in [0.00, \infty] \).

Unfortunately, within the work related only empirical-case analysis is considered. Thus, in Chapter 4.3 worst-case competitive analysis of the heuristic conversion algorithms VMA, FMA and TRB is done. Chapter 3 presents empirical-case analysis and work related to VMA, FMA and TRB.

### 2.4.2 Preemptive Conversion

Preemptive algorithms allow the search for more than one price in the time interval to convert the asset. Typically, a specific fraction of the whole amount available is converted at points of time \( t \) during \( T \). Let \( s_{t} \) be the amount to be converted at time \( t \), then \( s_{t} \in [0, 1] \). The only restriction is that during \( T \) an asset must be completely converted into another asset, i.e. \( \sum_{t=1}^{T} s_{t} = 1 \), and that at most \( T \) prices can be accepted for conversion.

Not all, but a great amount of algorithms addressed in the work related can be classified dependent on the calculation of \( s_{t} \). If possible, we classify the algorithms as follows:\footnote{In case the classification is not clear, the algorithms are presented at the beginning of the section.} The class of *threat-based algorithms* converts different amounts \( s_{t} \in [0, 1] \) of an asset at different points of time \( t \) during the time interval of length \( T \) \( (t = 1, \ldots, T) \) while assuming that the worst possible price occurs on day \( t+1 \). The class of *constant rebalancing algorithms* converts fixed fractions \( s_{t} = \frac{1}{T} \) of an asset at every point of time \( t \) during \( T \). The class of *risk-rewarded algorithms* algorithms
converts different amounts $s_t \in [0, 1]$ of an asset at different points of time $t$ during $T$ dependent on the acceptable level of risk $a \in [1, c]$. The amount to be converted $s_t$ is calculated such that the more risk is taken, the smaller the competitive ratio gets.

Raghavan (1992) analyze the performance of $ON$ under a statistical restriction on the input sequence(s) considered. Raghavan (1992) addresses a simple version of the asset allocation problem. Here $ON$ can invest in two assets: A risky and a risk-free asset. Based on the observed asset prices, $ON$ must decide at each point of time how to divide the available wealth among these two assets. The problem is analyzed using a statistical adversary.\(^{13}\)

Inspired by Raghavan (1992), DeMarzo et al. (2006) design an asset allocation algorithm to distribute the current wealth among a risky and a risk-free asset. At each point of time $t$ $ON$ converts an amount $s_t$ into a risky asset, and $1 - s_t$ into a risk-free asset. $ON$ converts using different assets $j = 1, \ldots, J$, and the goal is to achieve the performance of the best asset ($OPT$). $ON$ maintains weights $\omega_{j,t}$ for each $j$ at time $t$ and updates the weights each day. Each point of time $t$ $ON$ forms a portfolio where $s_t$ converted into asset $j$ equals $s_{j,t} = \frac{\omega_{j,t}}{W_j}$ with $W_j = \sum_{t=1}^{T} \omega_{j,t}$.

The authors show how to use the proposed algorithm to price the current value of an option.

In the following we differ between works on uni-directional and bi-directional search.

### 2.4.2.1 Uni-directional Search

Preemptive conversion algorithms for uni-directional search are presented in the following. Here, $ON$ is allowed to convert an asset $D$ into asset $Y$ but conversion back into $D$ is forbidden. Unfortunately, the work related is limited to guaranteeing conversion algorithms and the performance of the algorithms is evaluated using competitive analysis.

Chen et al. (2001) assume that the price function $g(q_t)$ and the number of days $T$ are known. Each ‘next’ price $q_{t+1}$ depends on the current price $q_t$ in a geometric manner: $q_t/\beta \leq q_{t+1} \leq q_t \cdot \alpha$, where $\alpha, \beta > 1$ (cf. the bounded daily return model in Chen et al. (2001, p. 448)). Some initial wealth to be invested according to a $T$-day investment plan is assumed. $ON$ runs the so called balanced strategy ($BAL$). Each day $t$, the amount to be converted $s_t$ is determined by $BAL$ such that the performance of $ON$ is balanced on all market downturns (downward runs). The results of $BAL$ are compared to constant rebalancing ($CR$) while carrying out

\(^{13}\)The input sequence generated by a statistical adversary has to satisfy specific statistical properties, cf. Chapter 1.
simulation runs using daily closing prices of the Taipei Stock Exchange (TSE) for
the year 1997. BAL and CR are money-making except in September, October,
and December 1997. Overall BAL outperforms CR.

Hu et al. (2005) suggest two algorithms. The static mixed strategy depends on
$T$ and the price fluctuation ratio $\varphi = \frac{M}{m}$. The dynamic mixed strategy depends on
the remaining trading days $T' = T - t + 1$, $\varphi$, and the remaining wealth. In both
cases, at the start of each day $t$, ON has some initial wealth. For each observed
price $q_t$ ON converts some amount $s_t \in [0,1]$ of the wealth. The amount to be
invested $s_t$ is (re)calculated on each day, and all remaining wealth on day $T - 1$
must be converted on day $T$. The performance of both algorithms is compared to
a special variant of CR (constant rebalancing) based on Nash Balances.\footnote{For the
definition of Nash Balances see Rubinstein and Osborne (1994).}

Results show that CR is outperformed by both algorithms on data of the China Merchants
Bank Co., Limited (CMB) for the year 2003.

Lorenz et al. (2009) assume that $m$ and $M$ are known. Further, the number
conversions is limited by the value $u$, i.e. not more than $u$ preemptions are allowed.
Two different RP algorithms are given, one for max-search and one for min-search. It is
assumed that ON may convert $u \geq 1$ times (originally denoted as $k$-search problem).
At each point of time $t$ it must be immediately decided whether or not to convert
two unit of the asset for the observed price $q_t$. At the start of the ‘game’ $u$
different reservation prices $q_t^i$, where $i = 1, \ldots, u$, and $u \leq T$ are calculated: For
min-search $q_t^i = m \cdot \left[1 + (c_{\text{max}} - 1) \cdot (1 + \frac{c_{\text{max}}}{u})^{i-1}\right]$, and for max-search $q_t^i = M \cdot \left[1 - (1 - \frac{1}{c_{\text{min}}} \cdot \frac{1}{u}) \cdot (1 + \frac{1}{u \cdot c_{\text{min}}})^{i-1}\right]$ where $c_{\text{max}}$ is a competitive ratio for max-search
and $c_{\text{min}}$ a competitive ratio for min-search (Lorenz et al., 2009, pp. 280-281). The
suggested algorithm is: Accept a price $q_t$ for selling (buying) iff $q_t \geq (<) q_t^i$. Hence,
the algorithm accepts the first price that is at least (lower) $q_t^i$ for selling (buying)
to convert for the first time. Then the algorithm waits for the first price that is at
least (lower) $q_t^u$, etc. Lorenz et al. (2009) make no assumptions on the price path
except that prices $q_t \in [m, M]$. The suggested algorithm may be forced to convert
at the last price $q_T$ of the sequence in order to meet the constraint of converting
the whole asset within $T$, with $q_T > (\leq) q_t^i$.

\textbf{Constant Rebalancing Algorithms.} Constant rebalancing (CR) algorithms
are a popular method to carry out uni-directional search. A CR algorithm does
not convert the entire asset at one single point of time. Rather, a fixed fraction of
asset $D$ is converted at regular increments across time (El-Yaniv et al., 2001, pp.
117; 135). Given $J$ assets, the amount to be converted $s_t = \frac{j}{T}$, with $t = 1, \ldots, T$
days, and $j = 1, \ldots, J$ assets (Butenko et al., 2005, p. 9). Suppose uni-directional
preemptive conversion: Asset $D$ is to be irreversibly converted into asset $Y$ within a given number of days $T$. Then a $CR$ algorithm converts equal amounts of $D$ on each day $t$, i.e. $s_t = \frac{1}{T}$, with $t = 1, \ldots, T$. Thus, the overall accumulated amount of asset $Y$ achieved by the $CR$ algorithm, denoted by $y_T$, equals

$$y_T = \sum_{t=1}^{T} \frac{q_t}{T} = \frac{1}{T} \sum_{t=1}^{T} q_t. \tag{2.20}$$

The $CR$ method ensures that an algorithm does not convert the whole asset at a market high (low), and thus the investor regrets the decision ex-post. Instead, the goal is to keep the same distribution of wealth among an asset from day to day, resulting in an average price.\footnote{Constant rebalancing is also known as ‘dollar-cost averaging’ or ‘average price trading’.} In the following we give a brief overview on the work related. $CR$ algorithms are often used as a benchmark when empirical-case analysis of preemptive conversion algorithms is done, see e.g. Chen et al. (2001); Hu et al. (2005).

Constantinides (1979) firstly demonstrate that $CR$ algorithms are suboptimal theoretically. Later, many empirical studies have compared $CR$ algorithms other conversion algorithms, and also found $CR$ to be suboptimal.

Bertsimas and Lo (1998) derive conditions on price dynamics under which a $CR$ algorithm for converting $j = 1, \ldots, J$ assets minimizes the cost of execution. Works on optimal trade execution are not discussed here, and the reader is referred to the overview in Bertsimas and Lo (1998) and Leggio and Lien (2003).

Blum and Kalai (1999) present a $CR$ algorithm that rebalances monthly under transaction costs, and compare its performance to $OPT$. On all data sets considered the $CR$ algorithm achieves inferior returns to $OPT$ but still outperforms the market when the transaction costs are less than 2%.\footnote{Blum and Kalai (1999) use the data sets suggested by Cover and Ordentlich (1996); Ordentlich and Cover (1998); Helmhold et al. (1998).} Blum and Kalai (1999) show that rebalancing less frequently, i.e. monthly instead of daily, is beneficial when transaction costs are high.

Almgren and Chriss (2000); Almgren (2003) propose different predefined (sequences of) constant fractions $s_t \in [0, 1]$ to be converted on each day $t = 1, \ldots, T$. The value of $s_t$ depends on assumptions on different parameters, such as risk tolerance, transaction costs, or price volatility.

Borodin et al. (2004) suggest to exploit the market volatility. The goal is to benefit from statistical relations between different assets by ‘trying to learn the winners’. The first approach is to learn from experts, i.e. to design a (reward-based)
CR algorithm which computes the weighted average of expert ratings. An update rule is used to gradually increase the relative weights of more successful experts. Three different learning CR algorithms are presented which rebalance a portfolio each day depending on yesterday’s weighted expert advices. The second approach is a CR algorithm that considers the market history: Two consecutive time intervals of equal length $T$ are considered to model statistical relations between different pairs of assets. The suggested CR algorithm takes advantage when an asset outperforms other assets especially if this outperformance is anti-correlated with the performance of the other assets. Thus, the CR algorithm is called AntiCor. An experimental study of the three learning CR algorithms and the AntiCor algorithm is presented. The results are compared to classical CR, to the bi-directional algorithm of Cover (1991), to the universal portfolio of Cover and Ordentlich (1996), and to $BH$.\footnote{The bi-directional algorithm of Cover (1991) is presented in Section 2.4.2.2.} The AntiCor algorithm outperforms all algorithms.

**Threat-based Algorithms.** Unlike CR algorithms, threat-based algorithms partition the amount to be converted $s_t$ where each $s_t$ has a different value ($0 \leq s_t \leq 1$) depending on the price $q_t$ offered to $ON$.

El-Yaniv et al. (1992, 2001) consider currency conversion in *Forex Markets*. Dollars $D$ must be converted into yen $Y$ to solve the *max-search problem*. The optimal performance is obtained by Algorithm 8, p. 92, commonly referred to as the threat-based policy (El-Yaniv et al., 1992, 2001, p. 3; p. 109).

The authors develop different variants of the threat-based algorithm; for each of these variants the achievable competitive ratio $c$ depends on the assumptions on the a-priori knowledge about the future of $ON$. Four variants are suggested, assuming:

1. Variant: Bounds $M$ and $m$, and number of days $k \leq T$
2. Variant: Bounds $M$ and $m$
3. Variant: Price fluctuation ratio $\varphi = \frac{M}{m}$, and number of days $k \leq T$
4. Variant: Price fluctuation ratio $\varphi = \frac{M}{m}$ are/is known. El-Yaniv et al. (1992, 2001) show that these variants of the threat-based algorithm gain the optimal (minimum) competitive ratio, and further suggest to repeat the uni-directional algorithm for bi-directional search. In addition, El-Yaniv et al. (1992, 2001) and Dannoura and Sakurai (1998) addressed the scenario where $m$ and $M$, as well as the first price $q_1$ are assumed to be known. The basic rules of the threat-based strategy remain the same.
different variants of the uni-directional algorithm of El-Yaniv et al. (1992, 2001) and Dannoura and Sakurai (1998) are presented in detail in Section 5.1.

Damaschek et al. (2009) assume $m$, $M$, and $T$ are known. The threat-based algorithm of El-Yaniv et al. (1992, 2001) is improved by assuming that the upper bound is a decreasing function of time, with $M_t = \frac{M}{t}$, and the lower bound $m$ is constant. The authors theoretically derive the best achievable worst-case competitive ratio $c^*$ (the lower bound) for the case search is repeated over several downward runs. The ratio $c^*$ is found by computing a competitive ratio for each downward run and then choosing the maximum as $c^*$ (Damaschek et al., 2009, equation 23, p. 639). Numerical examples are presented showing that the algorithm achieves a better (smaller) competitive ratio than the original algorithm of El-Yaniv et al. (1992, 2001).

**Risk-Rewarded Algorithms.** This class of algorithms includes a flexible risk management mechanism to competitive analysis. This means that a forecast, in particular a (partial) probabilistic input model, can be included. ON is allowed to make a ‘forecast’. If the forecast comes true, then a better (smaller) ratio $c_1$ than the worst-case competitive ratio $c^{wc}$ is achieved. Otherwise the worst-case competitive ratio $c^{wc}$ holds, where $c_1 \leq c^{wc}$. The result are algorithms with a bounded loss within a pre-specified tolerance.

The risk-rewarded competitive analysis contains two approaches. The first approach is to allow ON to benefit from the investors capability in correctly forecasting the future sequence(s) of prices. The second approach is to allow the investor to control the risk by selecting ‘near optimal’ algorithms subject to personal the risk tolerance.

Al-Binali (1997, 1999) extend threat-based algorithm of El-Yaniv et al. (2001) by a framework in which investors may develop online conversion algorithms based on their acceptable level of risk (risk tolerance), and on forecasts on price rate fluctuations. The algorithm ON is allowed to make a ‘forecast’. If the forecast comes true ON gets a competitive ratio $c_1$, otherwise ON suffers the worst-case ratio $c^{wc}$. The important factor is, that the risk can be controlled by a factor of $a \in [1, c]$. Assume the forecast is that the price will increase to at least $M_1$. ON takes this forecast (rate $M_1$), and the risk-tolerance factor $a$. If the forecast comes true, the algorithm achieves a competitive ratio $c_1 = \frac{c}{a} \leq c \cdot a$, and is optimal under the following condition: If the forecast comes not true, the worst-case competitive ratio is not worse than $c^{wc} = c \cdot a$. In other words, in case ON takes some amount of risk ON gets an optimal reward ‘for’ this risk.

that the price will increase to some level. Iwama and Yonezawa (1999) also allow the opposite, i.e. the forecast is that the price will never decrease to some level. 2) Iwama and Yonezawa (1999) provide a scheme which enables including several forecasts. During conversion forecasts can be 'updated' (corrected). ON can make a forecast and then 'update' it by a second forecast, etc. Results show that the suggested algorithms are not optimal for the entire investment horizon considered, but for different time intervals.

2.4.2.2 Bi-directional Search

In the following preemptive conversion algorithms for bi-directional search are presented. Here, an algorithm on is allowed to convert an asset \( D \) into another asset \( Y \), and back into \( D \) within one time interval. The work related is only comprised of guaranteeing conversion algorithms.

Cover (1991) investigates the portfolio selection problem. An algorithm that dynamically determines the amount of asset \( D \) to be converted \( s_t \) among \( J \) different assets \( j = 1, \ldots, J \) is presented. The goal is get the maximum value of asset \( D \) after time \( T \) based on the market history.

**Threat-based Algorithms.** El-Yaniv et al. (1992, 2001) assume \( M \) and \( m \) to be known and consider run search. ON divides the time series of prices into upward runs and downward runs, and then repeats the uni-directional algorithm suggested by El-Yaniv et al. (1992, 2001). Within one time interval of length \( T \) asset \( D \) is converted into asset \( Y \) if the price is moving up, and \( Y \) into \( D \) if the price is moving down. Though the uni-directional algorithm proposed in El-Yaniv et al. (1992, 2001) is shown to be optimal, the bi-directional algorithm is not. Therefore, the problem of designing an optimal threat-based algorithm for bi-directional search remains unanswered (El-Yaniv et al., 1992, p. 7). The bi-directional algorithm is presented in detail in Section 5.2.

Chou et al. (1995) provide a framework to analyze the bi-directional algorithm of El-Yaniv et al. (1992, 2001) considering a statistical adversary, i.e. by allowing only certain input distributions.

Dannoura and Sakurai (1998) improve the bi-directional algorithm suggested by El-Yaniv et al. (1992). The authors use the fact that the uni-directional algorithm of El-Yaniv et al. (1992) induces an optimal algorithm for bi-directional search under certain restrictions on the sequence of prices, such that the price increases from \( m \), then drops again to \( m \), and repeats such fluctuations (Dannoura and Sakurai, 1998, Figure 2, p. 30). As El-Yaniv et al. (1992) suggested, the improved uni-directional algorithm is repeated for bi-directional
search. Dannoura and Sakurai (1998) claim that an investor using the algorithm of El-Yaniv et al. (1992) faces too much of a threat and therefore make the threat smaller. The threat assumed by El-Yaniv et al. (1992) is that the price might drop to $m$ and will remain there until the last day $T$. Dannoura and Sakurai (1998) observed that the algorithm suggested by El-Yaniv et al. (1992) does not convert at all unless the price is as large as $c \cdot m$, i.e. the ‘real’ threat is at most $c \cdot m$ (not $m$) and shall not go beyond this point. Dannoura and Sakurai (1998) prove that their proposed threat-based algorithm achieves a better worst-case competitive ratio than the algorithm of El-Yaniv et al. (1992). The improved bi-directional algorithm suggested by Dannoura and Sakurai (1998) is presented in detail in Section 5.3.

In case the input data processed by an online conversion algorithm does not represent the worst-case input, its performance is often considerably better than the worst-case competitive ratio tells. For this reason competitive analysis is criticized as being too pessimistic (see, for example, Koutsoupias and Papadimitriou, 2000). Hence, the traditional approach to analyze online conversion algorithms is backtesting. The algorithms are implemented, and the analysis is done on historic data by simulation runs. Empirical-case analysis of online conversion algorithms is presented in the next chapter.

References for Chapter 2


REFERENCES FOR CHAPTER 2


REFERENCES FOR CHAPTER 2


Chapter 3

Empirical Analysis of Online Conversion Algorithms

This chapter gives an approach to empirically analyze online conversion algorithms. First, we present the idea of backtesting and introduce stylized facts. Then we present exploratory data analysis and provide the steps how to empirically analyze online conversion algorithm using this data analysis approach. We give the work related relevant for each step. Further, we focus on hypothesis testing and present the resampling method bootstrapping. The chapter concludes with an overview on heuristic conversion algorithms analyzed using hypothesis tests and/or a bootstrap procedure.

3.1 Introduction

There is a lack of consensus on a generally accepted performance evaluation model for online conversion algorithms. Several approaches exist, most common is to analyze the performance of \( ON \) using returns, or by different measures estimating (risk) adjusted returns (Tezel and McManus, 2001, pp. 177-181). We suggest evaluate the quality of \( ON \) by the three following criteria:

1. The worst-case competitive ratio \( c^{wec} \) assuming the worst possible sequence of inputs,

2. the empirical-case performance (in terms of the return to be expected \( \mu \)) on an observed time series of prices, and

3. the empirical-case competitive ratio \( c^{ec} \) on an observed time series of prices.

Classical (worst-case) competitive analysis, as presented in Chapter 2, derives the \( c^{wec} \) of \( ON \) assuming a constructed worst-case time series of prices. In contrast,
classical empirical-case analysis considers an observed time series of prices, and carries out experiments on this data set, e.g. using historical data. On the one hand \( \mu \) is derived, and on the other hand \( C_{ec} \). In order to clarify the difference between the above three criteria suppose two different online conversion algorithms, denoted by \( A_1 \) and \( A_2 \). Both algorithms \( ON \in \{A_1, A_2\} \) solve the search for best prices as presented in Section 2.2, Algorithm 1, p. 27. The question is how to decide which is the better algorithm.

The worst-case competitive analysis approach is to evaluate \( A_1 \) and \( A_2 \) on a constructed data set representing the worst-case scenario. To decide which is the better algorithm, each algorithm \( ON \in \{A_1, A_2\} \) is compared to \( OPT \) by calculating its worst-case competitive ratio \( C_{ec} \) as given in equation (2.19). The algorithm which achieves the smaller \( C_{ec} \), is considered as the better one. If the worst-case occurs \( ON \) is then guaranteed \( 1/C_{ec} \) of the result achieved by \( OPT \) (cf. equation (2.1)). A great deal of literature focuses on the worst-case performance analysis of online conversion algorithms; an overview can be found in Section 2.4.

The leading experimental approach to decide which algorithm \( ON \in \{A_1, A_2\} \) is the better one is backtesting. The aim of backtesting is to make assumptions about the future performance of an algorithm (in terms of \( \mu \)) based on its performance in the past. \( A_1 \) and \( A_2 \) are run on data sets comprised of historical time series of prices.\(^{18}\) The empirical-case performance of \( ON \) is measured in terms of the overall (excess) return generated.\(^{19}\) The algorithm which achieves a (significantly) higher return is considered as the better one. Typically, \( ON \) is compared to a passive benchmark algorithm \((B)\), and not to \( OPT \) (see for example Zontos et al., 1998; El-Yaniv et al., 1999; Schulenberg and Ross, 2002; Shen, 2003; Siganos, 2007; Larsen (Jr.) and Resnick, 2008; Chavarrakul and Enke, 2008).

To test for significance, the (distributions of the) returns generated by \( ON \in \{A_1, A_2, B\} \) are analyzed statistically, e.g. using hypotheses tests (Brock et al., 1992). Based on these statistical results a decision is taken which algorithm \( ON \) is the 'best' one, and thus should be applied in practice as it generates the 'highest' (excess) return (resp. \( \mu \)): It is assumed that the return generated in the past can be expected in the future. A great deal of experimental studies in the literature use this standard approach, especially in the field of heuristic conversion algorithms; an overview is given at the end of this chapter.

Following the above experimental approach, different algorithms are either compared directly to each other, or to a benchmark algorithm. This approach might

---

\(^{18}\)We do not consider artificial stock markets, an overview can be found in Palmer et al. (1994); LeBaron et al. (1999).

\(^{19}\)An excess return is the amount by which the return of \( ON \) is greater than the risk-free rate of return over a time interval of length \( T \).
be misleading. When comparing the algorithms directly to each other a mutual basis of comparison is missing, and when comparing to a benchmark, $B$ might not be suitable. This problem is solved by the competitive analysis approach presented in Chapter 2. Each $ON$ is compared to $OPT$, and the worst-case competitive ratio $c^{wc}$ determines the quality of $ON$. But this approach is often considered to be too pessimistic as – instead of an observed historic time series of prices – worst-case scenarios are assumed. We suggest to solve this problem by calculating the empirical-case competitive ratio $c^{ec}$ which takes the data of the problem instance into account.

The empirical-case competitive ratio $c^{ec}$ is calculated in the same manner as the worst-case competitive ratio $c^{wc}$. But instead of a constructed worst-case time series of prices the data set used in the experiments is considered. To decide which is the better algorithm, the observed performance of $ON \in \{A1, A2, B\}$ is compared to $OPT$ through backtesting. The quality of $ON$ is determined by $c^{ec}$ (cf. equation (2.19)) and by $\mu$. The algorithm which achieves the smallest (highest) $c^{ec} (\mu)$ is considered as the ‘best’ one.

Each of the above three criteria is useful when evaluating online conversion algorithms but in case they are used independently the results might be misleading. When considering an online conversion algorithm for practical application, worst-case performance guarantees are essential, e.g. in case of a stock market meltdown. But in terms of converting assets the worst-case competitive ratio $c^{wc}$ does not reveal which returns can be expected, nor whether these returns are positive or not. Hence, experiments should be carried out. An elegant solution is to combine the competitive analysis approach with the experimental approach when analyzing online conversion algorithms. On the one hand, the worst-case performance of $ON$ is determined and analyzed mathematically. On the other hand, the empirical-case competitive ratio $c^{ec}$ and the return to be expected $\mu$ are essential to determine whether $ON$ is considerably better than the pessimistic worst-case competitive ratio $c^{wc}$ tells. Thus, we suggest the following approach:

1. Step: Analyze $ON$ assuming a worst-case sequence of prices, and analytically derive its worst-case competitive ratio $c^{wc}$.

2. Step: Implement and backtest $ON$ (in a sufficient test environment) using historical time series of prices.

3. Step: Determine the return to be expected $\mu$ from $ON$. Analyze the empirical-case performance of $ON$ compared to a benchmark $B$ for the purpose of formulating hypotheses worth testing, and test these hypotheses statistically.
4. Step: Determine and analyze the empirical-case competitive ratio $c^{ec}$ of $ON$.

5. Step: If necessary, carry out further experiments on different data sets in order to evaluate the empirical-case performance achieved by $ON$ on the original data set.

How competitive analysis analysis of $ON$ (1. Step) is done is shown in Chapter 2. Experimental analysis according to steps 2. to 5. is presented in the following.

## 3.2 Backtesting and Stylized Facts

The implementation and simulation of an online conversion algorithm, also known as backtesting, is the concept of taking $ON$ and going back in time in order to see what would have happened if $ON$ had been followed (Ni and Zhang, 2005). The assumption is that if $ON$ has performed well previously, it has a good (but not certain) chance of performing well again in the future. Conversely, if $ON$ has not performed well in the past, it will probably not perform well in the future.

The backtesting of online conversion algorithms is important for practitioners as well as researchers to judge if $ON$ is profitable under certain circumstances. It helps to ‘learn’ how $ON$ is likely to perform in the marketplace, and also provides the opportunity to improve $ON$. The purpose of the backtesting is to answer the following questions:

1. Is $ON$ profitable when applied to certain stocks and time intervals?

2. If $ON$ generates (excess) returns for a certain stock, for what parameter values $ON$ achieves the highest ones?

3. Can these parameter values also generate a reasonable (excess) returns during future time intervals?

The outcome of a backtesting procedure are the returns generated by $ON$. In general, when converting assets, discrete (time interval) returns and continuous returns must be distinguished (Spremann, 2006, pp. 410-411).\(^{20}\) Let $q_t$ be the price of an asset on day $t$, then for a time interval $i$ of length $T$ days, the discrete return equals

$$R_t(i) = \frac{q_t}{q_{t-T}}$$

(3.1)

assuming $T < t$. Each time interval $i = 1, \ldots, p$ is initiated by a buying transaction at price $q_{t-T}$, and terminated by a selling transaction at price $q_t$. Thus, at the end

\(^{20}\)Discrete returns are also called holding period or time interval returns.
of the investment horizon overall $p$ trades (equaling the overall number of time intervals) are carried out. Most common is $T = 1$, resulting in the daily return

$$R_t(i) = \frac{q_t}{q_{t-1}},$$

and the percentage return is calculated by $R_t(i) - 1$.

When calculating the empirical-case competitive ratio $c_{ec}$ of an algorithm $X \in \{ON, OPT\}$, the time interval return of $X$ for each $i$-th trade is required. Thus, equation (3.1) equals equation (2.9), p. 34. Further, from equation (3.1) we get the continuous return

$$r_t(i) = \ln R_t(i)$$

$$= \ln q_t - \ln q_{t-T}. $$

Equations (3.1), (3.2) and (3.3) calculate the returns of single time intervals $i$. The return of an algorithm $X \in \{OPT, ON\}$ over multiple time intervals $p$ must be calculated in a geometric manner using equation (3.2); denoted as geometric return

$$R_X(p) = R_t(i) \cdot R_{t-1}(i) \cdot \ldots \cdot R_{t-p+1}(i)$$

$$= \frac{q_t}{q_{t-1}} \cdot \frac{q_{t-1}}{q_{t-2}} \cdot \ldots \cdot \frac{q_{t-p+1}}{q_{t-p}},$$

and for a constant time interval length $T$

$$R_X(p) = \prod_{i=1}^{p} \frac{q_t(i)}{q_{t-T}(i)}. $$

Using discrete returns, we get the overall logarithmic return

$$r_X(p) = \ln \left( R_t(i) \cdot R_{t-1}(i) \cdot \ldots \cdot R_{t-p+1}(i) \right)$$

$$= \ln R_t(i) + \ln R_{t-1}(i) + \ldots + \ln R_{t-p+1}(i)$$

$$= r_t(i) + r_{t-1}(i) + \ldots + r_{t-p+1}(i)$$

$$= \ln R_X(p).$$

In case continuous returns $r_t(i)$ are used, they can simply be added to get the logarithmic return $r_X(p)$ over multiple time intervals (instead of multiplying the discrete returns $R_t(i)$ to get the geometric return $R_X(p)$). But continuous returns $r_t(i)$ suffer from a drawback: They can not be used to calculate portfolio returns. Let $\omega_j$ be the weight of an asset $j = 1,\ldots,J$ within a portfolio, then

$$\omega_1 \cdot \ln R_{(1,t)}(i) + \ldots + \omega_J \cdot \ln R_{(J,t)}(i) \neq \ln \left( \omega_1 \cdot R_{(1,t)}(i) + \ldots + \omega_J \cdot R_{(J,t)}(i) \right).$$

The logarithmic return over multiple time intervals $p$ can not be calculated directly by the continuous return of single time intervals $i = 1,\ldots,p$. Thus, we use the geometric return, as given in equation (3.4), within this work.
Trading systems enable a user to develop and backtest online conversion algorithms. Simple algorithms are relatively easy to implement and test. But the more complex the investigated algorithms get, the more data must be processed. Further, some algorithms use multiple stocks, and even multiple markets. All these factors make backtesting very time-consuming, and many ready-for-use commercial products become incapable of dealing with them (Ni and Zhang, 2005, p. 127). Thus, within this work, we use the LifeTrader system as it provides the required functionality for backtesting the considered online conversion algorithms. LifeTrader is a PES (planning and execution system) developed at the Saarland University; an overview on its functionality can be found in Kersch and Schmidt (2011). Further, the above suggested steps to evaluate an algorithm are covered by the LifeTrader system.

The aim of backtesting is to make assumptions about the return to be expected $\mu$ based on the performance of $ON$ in the past. In the work related it is assumed that future asset returns are independently distributed random variables drawn from the same probability distribution. Further, it is assumed that the returns generated by $ON$ are normal distributed (Spremann, 2006, p. 123). Within this work, we assume that these assumptions are close to reality, but must not always be true for a specific data set considered. Thus, when empirically analyzing the performance of $ON$ the properties of the discrete returns generated by $ON$ – in case the algorithm is invested – must be analyzed. These properties are called ‘empirical stylized facts’, and characterize a data set from a statistical point of view. Stylized facts are usually formulated in terms of qualitative properties of daily returns $R_t(i)$ calculated using equation (3.2) (Cont, 2001, p. 224). The stylized facts are summary statistics, and contain (Brock et al., 1992, p. 1737):

1. The number $p$ – also known as the sample size,

2. the arithmetic mean

$$\bar{r} = \frac{1}{p} \cdot \sum_{i=1}^{p} R_t(i), \quad (3.8)$$

3. the standard deviation

$$\sigma = \sqrt{\frac{1}{p-1} \cdot \sum_{i=1}^{p} (R_t(i) - \bar{r})} \quad (3.9)$$

defined as the square root of the variance $\sigma^2$,

4. the skewness

$$\gamma = \frac{p}{(p-1) \cdot (p-1)} \cdot \sum_{i=1}^{p} \left( \frac{R_t(i) - \bar{r}}{\sigma} \right)^3, \quad (3.10)$$
5. the kurtosis

\[ \beta = \left[ \frac{p \cdot (p - 1)}{(p - 1) \cdot (p - 2) \cdot (p - 3)} \cdot \sum_{i=1}^{p} \left( \frac{R_t(i) - \bar{r}}{\sigma} \right)^4 \right] - \frac{3 \cdot (p - 1)^2}{(p - 2) \cdot (p - 3)} \]  

(3.11)

of the observed daily returns (Spremann, 2006, Formula (5-12), (5-13) and (5-14)). The arithmetic mean \( \bar{r} \) is commonly used as the estimator for the (unknown) return to be expected \( \mu \) in the future. The standard deviation \( \sigma \) shows the variation from the mean \( \bar{r} \). A low standard deviation indicates that the observed returns tend to be very close to the mean \( \bar{r} \), whereas a high standard deviation indicates that the returns are spread out over a large range of values.

The skewness \( \gamma \) measures the (a)symmetry in the probability distribution of the observed returns. In case of normal distributed data \( \gamma = 0 \). In case \( \gamma > 0 \) (positive skewness) the right tail of the distribution is longer, i.e. the mass of the distribution is concentrated on the left, and relatively few high returns exist. In case \( \gamma < 0 \) (negative skewness) the left tail of the distribution is longer, i.e. the mass of the distribution is concentrated on the right, and relatively few low returns exist. Figure 3.1 gives an example for positive skewness and \( \bar{r} = 0 \) in case of a normal distribution.

The kurtosis \( \beta \) measures with which probability extremely low or extremely high returns might occur. In case of normal distributed data \( \beta = 3 \).\(^{21}\) In case

\(^{21}\)The excess kurtosis is defined as \( \beta - 3 \), i.e. the excess kurtosis of the normal distribution equals 0.
\( \beta > 3 \) (leptokurtosis) both tails of the probability distribution are ‘fat’, i.e. the mass of the distribution is concentrated on the left and on the right. Relatively may high and low returns exist. Figure 3.2 gives an example for \( \bar{r} = 0 \).

![Figure 3.2: Kurtosis](image)

The stylized facts, especially the skewness and the kurtosis, are used to check the assumption that the returns generated by ON are normal distributed. The Jarque-Bera \((JB)\) test is a non-parametric hypothesis test to check the null hypothesis \( H_0 \) that ‘the returns achieved by ON are normal distributed’ (Jarque and Bera, 1987). In particular two hypotheses are tested, the first one is that \( \gamma = 0 \), and the second one is that \( \beta = 3 \). In case the value of \( \beta \) (\( \gamma \)) is ‘not close enough’ to 3 (0) \( H_0 \) is rejected. The range of tolerance not to reject \( H_0 \) is given by the variances of \( \gamma \) and \( \beta \). For the skewness the variance equals \( \frac{6}{p} \), and for the kurtosis \( \frac{24}{p} \) (Spremann, 2006, p. 145).

Within this work as data set we consider the German Dax-30 index for the investment horizon 01-01-1998 to 12-31-2007, resulting in \( T = 2543 \) closing prices. We refrained from considering the year 2008 as it marks a major structural break in the markets worldwide. The common benchmark algorithm when backtesting online conversion algorithms is a passive buy-and-hold algorithm \((BH)\) (Brock et al., 1992).

**Example 2.** The stylized facts of the daily returns achieved by \( BH \) for the 10-year sample 1998-2007 are given in Table 3.1. As \( BH \) is invested in the Dax-30 index from the first trading day (01-02-1998) until the last day trading (12-28-2007) of the investment horizon we get a sample size of \( p = T - 1 \) daily returns. Using the
Exploratory Data Analysis

<table>
<thead>
<tr>
<th>Sample Size</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Skewness</th>
<th>Kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>( \bar{r} )</td>
<td>( \sigma )</td>
<td>( \gamma )</td>
<td>( \beta )</td>
</tr>
<tr>
<td>2542</td>
<td>1.0004</td>
<td>0.0157</td>
<td>-0.0676</td>
<td>5.7064</td>
</tr>
</tbody>
</table>

Table 3.1: Stylized facts of the German Dax-30 index for 1998-2007

values given in Table 3.1 a JB test is performed. Results show that \( H_0 \) must be rejected, i.e. the daily returns of BH are not normal distributed.

Summing up, stylized facts give the qualitative properties of the analyzed returns. As shown in Example 2 the common assumption of normal distributed data must not always be true. Instead of making assumptions on the underlying structure of the data set considered our goal is to ‘let the data speak for themselves’ as much as possible. As a result, the approach to empirically analyze online conversion algorithms must be exploratory. To solve a problem, the exploratory data analysis (EDA) technique makes (little or) no assumptions on the data. Rather, results are immediately analyzed with the goal to infer what model would be appropriate. The EDA approach allows the data to suggest models that fit best.

3.3 Exploratory Data Analysis

Two popular data analysis approaches are (Hoaglin et al., 2000):

1. Bayesian Analysis, and

2. Exploratory Data Analysis (EDA).

These approaches are similar in that both start with a problem, and both yield conclusions. The difference lies in the sequence of processing the input data in order to solve the problem. The following elements are covered by both data analysis approaches: 1) Problem – the performance of ON, 2) Data – the returns generated by ON on the considered time series of prices, 3) Stochastic Model – an abstraction of reality; the stochastic process generating the data 4) Distribution – the (assumed) underlying structure of the data, 5) Analysis – the discussion of the data, 6) Conclusions – the inference on the performance of ON. For Bayesian Analysis the sequence of processing the input data is

Problem \( \rightarrow \) Data \( \rightarrow \) Stochastic Model \( \rightarrow \) Prior Distribution \( \rightarrow \) Analysis \( \rightarrow \) Conclusions
To solve a problem, data collection is followed by the imposition of a model (assumed) to fit the input data. The analysis that follows is focused on the parameters of that model. Further, assumptions about the distribution of the input data are made, or the distribution of the input data is known beforehand. The objective is to compute and analyze the empirical-case performance of ON under ‘typical inputs’ with respect to these stochastic assumptions. Unfortunately, most currently existing models fail to reproduce the underlying data structure (Cont, 2001, p. 233). Thus, the ‘Bayesian’ approach is criticized from both a technical, and a conceptual perspective. Technically, for many real-life problems, an adequate stochastic model is extremely difficult or costly to devise. Conceptually, the validity of the conclusions becomes dependent on the validity of the underlying (distributional) assumptions (El-Yaniv et al., 1999). Worse yet, the exact underlying assumptions may be unknown, or if known, untested. For this reason the ‘Bayesian Analysis’ approach is not considered here (cf. Section 1.1). Instead, we focus on exploratory data analysis (EDA). The main difference is that the distribution and the stochastic model are derived from the data, and not assumed a-priori. Thus, for Exploratory Data Analysis the sequence of processing the input data is

Problem → Data → Distribution → Analysis → Stochastic Model → Conclusions

In case online conversion algorithms are evaluated using EDA the focus is not on the process or model generating the data, but on the analysis of the data generated by ON. EDA is used analyze the computed empirical-case returns, and to suggest how to further analyze them. A variety of graphical and quantitative techniques might be employed in order to

- maximize the insight into the returns generated, e.g. to detect outliers and anomalies,
- assess assumptions on the stochastic model,
- uncover underlying data structures, e.g. distributions,
- support the selection of appropriate statistical tools and techniques for further analysis,
- suggest hypotheses to test (statistically) based on the returns generated,
- provide a basis for further data collection through experiments, e.g. by resampling methods like bootstrapping.
Exploratory Data Analysis

The EDA approach an attitude (philosophy) about how data analysis should be carried out. The stylized facts of an algorithm give an insight into the returns generated, and uncover the underlying structure of the achieved returns. This enables to select the appropriate statistical tools for further analysis, i.e. the adequate statistical test. The returns generated are analyzed for the purpose of formulating hypotheses worth testing. This distinguishes EDA from classical hypothesis testing, which requires a-priori formulated hypotheses (Oldenburger, 1996, pp. 71-72). Hypothesis tests are used to decide which algorithm under investigation is the better one on a specific time series of prices. In case the chosen hypothesis test does not provide a result, i.e. there is no statement possible which algorithm is the better one, further data sets must be considered. In the following we present two standard approaches from the literature used to evaluate the performance of an online conversion algorithm. First, we present the student $t$-test for testing hypotheses, and second the bootstrapping procedure for generating further data sets if required.

### 3.3.1 Hypothesis Testing

Before describing the student $t$-test in detail we first give some preliminaries on statistical tests. A statistical test which uses hypotheses is called hypothesis test. Two types of hypothesis tests exist (Cont, 2001, p. 223):

1. **Parametric tests:** Assume that the data to be analyzed belongs to a prespecified parametric family, for example require a certain distribution.

2. **Non-parametric tests:** Make only qualitative assumptions about the properties of the stochastic process generating the data, for example the JB test.

Cont (2001) states that non-parametric tests have the great theoretical advantage of being model-free, but in a financial context they can only provide qualitative information about a data set under investigation. Thus, non-parametric tests are less exact, and should only be used when parametric tests are not applicable.

A statistical hypothesis is a statement about the properties of one or several random variables, e.g. about the stylized facts or the distribution of the returns generated by ON. To confirm a hypothesis statistically a so-called null hypothesis ($H_0$) is defined which must to be rejected in order to confirm the (alternative) hypothesis ($H_1$) indirectly. Two types of hypotheses, based on the parameters of a distribution, can be distinguished:

1. Two-tailed: It is tested whether two parameter values are equal (unequal), e.g. $H_0 : \mu_1 = (\neq) \mu_2$ must be rejected.
2. One-tailed: It is tested whether one parameter value is greater (smaller) than or equal to another parameter value, e.g. \( H_0 : \mu_1 \geq (\leq) \mu_2 \) must be rejected.

A null hypothesis \( H_0 \) cannot be confirmed or rejected with certainty. Therefore a significance level \( \alpha \in [0,1] \) has to be specified. The value of \( \alpha \) describes the amount of evidence required to accept that an event is unlikely to have occurred by chance. The smaller the chosen significance level, the fewer the null hypothesis \( H_0 \) is rejected. The most established significance levels are 5% (0.05), 1% (0.01), and 0.1% (0.001). Next we present the student \( t \)-test as the standard parametric test applied by almost all the contributions to empirical evaluation methods for online conversion algorithms in the literature (Brock et al., 1992; Mills, 1997; Hudson et al., 1996; Gunasekara and Power, 2001).

**Student \( t \)-test**

The (student) \( t \)-test is a parametric one-tailed two-sample hypothesis test to show that the mean of one sample (of returns) is significantly greater than the mean of another sample. The \( t \)-test implies the following assumptions regarding the sample under consideration, i.e. the returns generated by \( ON \):

1. The returns generated by \( ON \) are (stochastically) independent, to be tested by the *Ljung-Box test* (Ljung and Box, 1978).

2. The underlying distribution of the returns under consideration is normal, to be tested by the *JB test* (Jarque and Bera, 1987).

3. The variances of the returns are homogeneous, to be tested by the *Bartlett test* for normal distributed samples, otherwise by the *Levene test* (Levene, 1960; Layard, 1973).

These assumptions have to be met if the \( t \)-test is to be valid. Within this work we do not discuss these limitations of the \( t \)-test, the reader is referred to Kumar et al. (1997, p. 341) and Wolfinger (1996, pp. 207-208). Further, we do not present the tests to verify the 1. and 3. assumption. The reader is referred to Levene (1960); Layard (1973) and Ljung and Box (1978).

The test statistic \( \Gamma \) used by the \( t \)-test follows a \( t \)-distribution if \( H_0 \) is not rejected. The shape of the \( t \)-distribution is specified by the degrees of freedom \( v \), and passes into the standard normal distribution with increasing \( v \). Thus, a normal distribution can be assumed in case the sample size \( p \) is greater than 30. In case the variances of the two samples are not equal an alternative to the \( t \)-test is the *Welch-test*. The only difference between the two-sample \( t \)-test and the *Welch-test* is the different calculation of \( v \) and \( \Gamma \) (Welch, 1947; Satterthwaite, 1946).
The $t$-test algorithm for evaluating the performance of $ON$ is given in the following. The $t$-test is significant when $H_0: \mu_1 \leq \mu_2$ is rejected at a significance level of $\alpha\%$. The value $\mu_1$ and $\mu_2$ specify the returns to be expected from $ON \in \{A1, B\}$, which are normally unknown. Therefore, when analyzing the performance of the two algorithms the means of the observed discrete returns generated by $A1$, denoted by $\bar{r}_1$, and generated by $B$, denoted by $\bar{r}_2$, must be used. To answer the question whether $A1$ is significantly better than a benchmark $B$ through backtesting, the values of $\bar{r}_1$ (with sample size $p_1$) and $\bar{r}_2$ (with sample size $p_2$) are calculated using equation (3.8), compared, and their difference is tested for significance using $t$-test. The steps of the $t$-test algorithm are (Ruppert, 2004, p. 64):

**Algorithm 2.**

**Step 1:** Specify the level of significance $\alpha$ in %.

**Step 2:** Formulate the one-tailed null hypothesis: It is tested whether $\mu_1$ is significantly greater than $\mu_2$ ($H_0: \mu_1 \leq \mu_2$ must be rejected).

**Step 3:** Specify two samples ($P_1$, $P_2$) and determine their size ($p_1$, $p_2$): Usually samples are comprised of (discrete) returns generated by $A1$ and $B$.

**Step 4:** Calculate the arithmetic mean $\bar{r}_1$ of $P_1$ and $\bar{r}_2$ of $P_2$ using equation (3.8).

**Step 5:** Calculate the variances $\sigma^2_1$ of $P_1$ and $\sigma^2_2$ of $P_2$ by squaring the standard deviation given in equation (3.9), and test for variance homogeneity.

**When the variances are equal:**

**Step 6a:** Calculate the degrees of freedom

$$v = p_1 + p_2 - 2.$$  \hspace{1cm} (3.12)

**Step 7a:** Calculate the test statistic

$$\Gamma = \frac{\bar{r}_1 - \bar{r}_2}{\sqrt{\frac{(p_1-1)\sigma^2_1+(p_2-1)\sigma^2_2}{v}\cdot\left(\frac{1}{p_1} + \frac{1}{p_2}\right)}}.$$  \hspace{1cm} (3.13)

**When the variances are not equal:**

**Step 6b:** Calculate the degrees of freedom

$$v = \left|\frac{\left(\frac{\sigma^2_1}{p_1}\right)^2 + \left(\frac{\sigma^2_2}{p_2}\right)^2}{\frac{\sigma^2_1}{p_1-1}\left(\frac{\sigma^2_1}{p_1}\right)^2 + \frac{\sigma^2_2}{p_2-1}\left(\frac{\sigma^2_2}{p_2}\right)^2}\right|.$$  \hspace{1cm} (3.14)
CHAPTER 3  Empirical Analysis of Online Conversion Algorithms

68

Step 7b: Calculate the test statistic

$$\Gamma = \frac{\bar{r}_1 - \bar{r}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}.$$  \hfill (3.15)

Step 8: Calculate critical value $t_{cr} = t_{1-\alpha}$ from the $t$-distribution.

Step 9: Take a decision; if

1) $\Gamma \geq t_{cr}$ then $H_0$ is rejected,

2) $\Gamma < t_{cr}$ then $H_0$ can not be rejected.

That $H_0$ cannot be rejected does not signify $H_1$ is valid; backtesting on further time series of prices is essential in case $\Gamma < t_{cr}$. The result of Algorithm 2 then is that there is no statement possible on the performance of $A_1$. This might be due to sample problems, or the implied $t$-test assumptions are violated. The $t$-test is robust, meaning it is quite insensitive to deviations from normality in the data. The most serious sample problem is that the variances are not homogeneous, called heteroskedasticity, meaning that the volatility of the returns evolves over time (Ruiz and Pascual, 2002, p. 1). To deal with this problem a number of recent papers has suggested to use resampling methods to generate further data sets for backtesting. The most common method is the so-called bootstrap procedure as it is robust to heteroskedasticity (Tabak and Lima, 2009, p. 816). Further, bootstrapping is a way of finding the 'most likely' sample distribution by generating many new random samples from the original sample. In the following we present the bootstrap procedure.

3.3.2 Resampling: The Bootstrap Procedure

Hypothesis testing using a $t$-test rests on the implied $t$-test assumptions. In case these assumptions are violated – when evaluating $ON$ – the bootstrap idea is based on asking: ‘What would happen if we applied $ON$ many times?’.

Efron (1979) suggested the name ‘bootstrap procedure’ (Wu, 1986, p. 1265). The main idea of a bootstrap procedure is to resample new data sets from the original sample creating $S$ bootstrap samples of the same size as the original sample: $S$ samples are created by repeatedly sampling with replacement. Sampling with replacement means that after an observation is randomly drawn from the original sample it is ‘put back’ before drawing the next observation. This classic bootstrap procedure suggested by Efron (1979) is the simplest version, and only valid for identically distributed data. If this assumption is violated, or in case the classic procedure is applied directly to dependent data, the resampled data will not preserve the properties of the original data set. As a result inconsistent statistical
Exploratory Data Analysis

results are provided (Ruiz and Pascual, 2002, p. 3). Consequently, alternative approaches have been developed.

Künsch (1989) proposes the Moving Block Bootstrap (MBB) that divides the original data set into overlapping blocks of fixed length, and resamples with replacement from these blocks. Within this work, we limit to the MBB procedure, for an overview on other bootstrap approaches the reader is referred to Ruiz and Pascual (2002). The MBB is a widely used non-parametric approach preserving the properties of the original data set (Künsch, 1989; Hall et al., 1995; Leivich and Thomas, 1993; Tabak and Lima, 2009). Let $S$ be the number of bootstrap samples to be generated, with $i = 1, \ldots, S$. Let $l$ be the block size $(1 \leq l \leq S)$, and $b_i(i) = \{x_{t}, \ldots, x_{t+l-1}\}$ a block formed with $l$ consecutive observations beginning with $x_t$. Where $b$ equals the number of blocks, with $i = 1, \ldots, b$. When evaluating the performance an online conversion algorithm the length of the original data set $T$ equals the number of prices $q_t$, and results in $T - 1$ daily returns within each $i$-th time interval ($i = 1, \ldots, p$). Then, the MBB algorithm for resampling $T - 1$ daily returns $R_t(i)$ generated by ON is comprised of the following steps (Hall et al., 1995; Tabak and Lima, 2009):

**Algorithm 3.**

Step 1: Determine the optimal block size $l^*$ according to the rule given in Hall et al. (1995).\(^{22}\)

Step 2: Calculate the number of blocks $b = \frac{S}{T}$ to be resampled.

Step 3: Split the sample of observed returns into $S - l + 1$ overlapping blocks $b_i(i) = \{R_t(i), R_{t+1}(i), \ldots, R_{t+(l-1)}(i)\}$.

Step 4: Resample the blocks $b_i(i)$ with replacement generating $S$ new bootstrap samples of length $T$.

Step 5: Calculate $S$ ‘pseudo’ time series of prices from the resampled (blocks of) returns using $S$ randomly chosen first prices $q_{1}^{\text{min}}(i) \in [q_{\text{min}}^\text{max}(i), q_{\text{max}}^\text{max}(i)]$ as a starting value, and $q_t = R_t(i) \cdot q_{t-1}$ for $t = 2, \ldots, T$ and $i = 1, \ldots, p$.

It is assumed that the blocks $b_i(i)$ are iid random variables with conditional probability $\rho(b_i(i)) = \frac{1}{S-l+1}$ (Tabak and Lima, 2009, p. 817). Further, Hall et al. (1995) show that the optimal block size $l^*$ depends significantly on the context, being equal to $\sqrt{T-1}$, $\frac{1}{\sqrt{T-1}}$ and $\sqrt{T-1}$ in the cases of variance or bias estimation, estimation of an one-sided distribution function, and estimation of a two-sided distribution function, respectively. The result of a bootstrap procedure are $S$ ‘pseudo’ time series. On each $i$-th bootstrap sample algorithms $X \in$

\(^{22}\text{For } l^* = 1 \text{ the } MBB \text{ is similar to the classic bootstrap procedure suggested by Efron (1979).} \)
\{OPT, ON\} are run, and the resulting in \( S \) arithmetic means \( \bar{r}(i)_X \) are commonly used as the estimator for the (unknown) rate of return to be expected \( \mu_X \) in the future, with \( i = 1, \ldots, S \). Thus, a typical bootstrap procedure to evaluate an algorithm \( X \in \{OPT, ON\} \) requires to:

1. Randomly resample from the original sample, creating \( S \) bootstrap samples of the same size as the original sample, according to Algorithm 3.

2. Run algorithm \( X \) on each of the \( S \) bootstrap samples to get \( S \) different arithmetic means \( \bar{r}(i)_X \) for each algorithm \( X \).

3. Statistically evaluate the performance of algorithm \( X \) on each of the \( S \) bootstrap samples according to Algorithm 2.

4. Combine the \( S \) statistical \( t \)-test results into one summary statistic for each algorithm \( X \).

5. For each algorithm \( X \) estimate the return to be expected \( \mu_X \) by calculating the mean \( \bar{r}^S_X \) of all arithmetic means \( \bar{r}(i)_X \), with \( i = 1, \ldots, S \).

The distribution of the \( i = 1, \ldots, S \) different arithmetic means \( \bar{r}(i)_X \) per algorithm \( X \) shows the 'most likely' stylized facts, and the 'most likely' performance of \( OPT \) and \( ON \). Summing up, when analyzing the empirical-case performance of an algorithm \( X \) the bootstrap procedure can be used to estimate the true but unknown (Ruiz and Pascual, 2002, p. 2)

1. distribution, or

2. probability distribution

of the population of the returns \( \bar{r}(i)_X \) generated by algorithm \( X \in \{OPT, ON\} \) from which the return to be expected \( \mu_X \) can be estimated through \( \bar{r}^S_X \). This ensures that the online conversion algorithms considered are compared \( S \) times on a mutual basis.

In the following we give an overview on online conversion algorithms evaluated using stylized facts, hypothesis testing as well as a bootstrap procedure. Unfortunately, the work related is limited to heuristic conversion algorithms. By carrying out Algorithm 2 and Algorithm 3 the question whether the (back) tested algorithms have predictive ability or not is to be answered. Most authors study the Efficient Market Hypothesis (EMH): The EMH states that in a (weakly) efficient financial market returns are not predictable (cf. Section 1.3.1). The predictability of returns is usually measured by the first-order autocorrelation coefficient, measuring the similarity between observations as a function of the time
separation between them. If a sufficiently large proportion of all traders acting in a
stock market behave ‘irrationally’, then the stock prices can, at least temporarily,
deviate from economic fundamentals (DeLong et al., 1990). This deviation of stock
prices from economic fundamentals can imply autocorrelation and, hence, the
predictability of returns: Repeating price patterns occur. Returns to be expected
\( \mu_X \) are considered to be ‘predictable’ in the sense that it is possible to forecast
returns in a particular time interval by using the returns observed in a previous
time interval (Pierdzioch, 2004). In addition, most authors employ a bootstrap
procedure to test for predictability.

### 3.4 Literature Review

We limit our overview to the two heuristic conversion algorithms suggested in
the work of Brock et al. (1992), namely Moving Average Crossover (\( MA \)) and
Trading Range Breakout (\( TRB \)). Brock et al. (1992, p. 1736) distinguish two
variants of the \( MA \) algorithm, namely Variable-length Moving Average (\( VMA \))
and Fixed-length Moving Average (\( FMA \)). The definition of \( VMA, FMA \) and
\( TRB \) can be found in Section 2.4.1.2. These three bi-directional algorithms are
of major interest in the literature, and have been analyzed experimentally by
several researchers (Vanstone and Finnie, 2009, p. 6673). Here, the comparison
to a passive buy-and-hold (\( BH \)) algorithm (as benchmark \( B \)) is of prime interest
using either hypothesis tests, a bootstrap procedure or both. The deviation of
stock prices from economic fundamentals is measured in terms of the return to be
expected: \( \mu_{ON} \) of \( ON \in \{ VMA, FMA, TRB \} \) is estimated and compared to \( \mu_B \)
of benchmark \( B \) through backtesting. The predictive ability of \( ON \) is based on
the assumption that if \( H_0 : \mu_{ON} \leq \mu_B \) is rejected, there is good (but not certain)
chance that \( ON \) performs better than algorithm \( B \) again in the future. In case
results show that the (excess) returns generated by \( ON \) are not significant, this
suggests that predictability is not economically significant.

Brock et al. (1992) suggest the algorithms \( VMA, FMA \) and \( TRB \) and conduct
experiments with a price-weighted index on an investment horizon of approximately
90 years from the first day 1897 to the last day 1986 (exactly 25036 trading days)
using the Dow Jones Industrial Average (DJIA) index (Brock et al., 1992, p. 1734).
Experiments are carried out for five different time intervals of length \( T \):

1. January 1897- December 1986 (‘90 Years’), \( T=25036 \),
2. January 1897 - July 1914 (‘World War I’), \( T=5255 \),
3. January 1915 - December 1938 (‘Depression’), \( T=7136 \),
4. January 1939 - June 1962 (‘World War II’), $T=6442$,

DJIA buy-and-hold ($BH$) is the benchmark considered, called ‘unconditional returns’. The performance is measured using logarithmic returns (cf. equation (3.6)) as they are time additive and approximate discrete returns if calculated on a daily basis (Brock et al., 1992, p. 1737). The returns on buy (sell) signals on the DJIA are compared to returns from simulated comparison series generated by the following models: Autoregressive (AR(1)), generalized autoregressive conditional heteroskedasticity in mean (GARCH-M), and exponential GARCH. The results provide empirical support for utilizing the heuristic conversion algorithms as they outperform not only $BH$ but also the AR(1), the GARCH-M, and the exponential GARCH model. The returns obtained from the algorithms are not likely to be generated by these three models. Brock et al. (1992) conclude that $VMA$, $FMA$ and $TRB$ have predictive ability. The suggested algorithms are presented and analyzed in detail in Section 4.3.

Bessembinder and Chan (1995) test whether $VMA$, $FMA$ and $TRB$ can predict stock price movements in Asian markets. The first result is that the algorithms are ‘quite successful’ in the emerging markets of Malaysia, Thailand and Taiwan, but have less predictive power in more developed markets such as Hong Kong and Japan. Transactions costs which could eliminate gains are estimated to be 1.57%. The second result is that buying and selling signals emitted by U.S. markets have substantial forecast power for Asian stock returns beyond that of own-market signals.

Hudson et al. (1996) test whether the finding by Brock et al. (1992) – that $VMA$, $FMA$ and $TRB$ have predictive ability – is replicable on the FT30 (Financial Times Ordinary) Index from July 1935 to January 1994. Further, the authors test whether the algorithms generate excess returns in a costly trading environment. Hudson et al. (1996) conclude that although $VMA$, $FMA$ and $TRB$ do have predictive ability in terms of UK data, their use would not generate excess returns in the presence of costs. In general, the results presented are remarkably similar to those of Brock et al. (1992). Thus, one conclusion to be drawn from both studies is that $VMA$, $FMA$ and $TRB$ have predictive ability if sufficiently long investment horizon is considered.

Mills (1997) also compares $VMA$, $FMA$ and $TRB$ to $BH$ by conducting experiments on the FT30 index for the time intervals 1935-1954 and 1975-1994. In addition, trading signals generated by a geometric MA are considered. The geometric MA gave an almost identical set of buying and selling signals as the conventional (arithmetic) MA. Until 1980 all algorithms outperform $BH$. The
results of Mills (1997) are consistent, in almost every respect, with those of Brock et al. (1992) and Hudson et al. (1996). But from 1980 on BH clearly dominates all other algorithms. The sample used in Brock et al. (1992) ends in 1986; so Mills (1997) concludes that there was not enough data to analyze structural shifts that might have taken place starting in 1982.

Ratner and Leal (1999) compare VMA and FMA to BH by investigating ten emerging equity markets in Latin America and Asia from 1982 to 1995 under transaction costs using the S&P500 and Nikkei225 indices. Results show that VMA and FMA applied to emerging markets do not have the ability to outperform BH.

Parisi and Vasquez (2000) test VMA, FMA and TRB in the Chilean stock market using the Indice de Precio Selectivo de Acciones (ISPA) from January 1987 to September 1998. The results are similar to the ones of Brock et al. (1992), providing strong support for VMA, FMA and TRB.

Gunasekarage and Power (2001) test VMA and FMA in four emerging South Asian capital markets from January 1990 to March 2000, i.e. the Bombay Stock Exchange, the Colombo Stock Exchange, the Dhaka Stock Exchange and the Karachi Stock Exchange. The findings indicate that the algorithms have predictive ability in these markets, and reject $H_0: \mu_X = \mu_{BH}$ with $X \in \{VMA, FMA\}$. Gunasekarage and Power (2001) conclude that VMA and FMA are able to generate excess returns in South Asian markets.

Kwon and Kish (2002) extend the work of Brock et al. (1992) in two ways. First, by investigating the predictive ability of VMA, FMA and TRB on the New York Stock Exchange (NYSE) index from July 1962 to December 1996, as well as on the National Association of Security Dealers Automatic Quotations (NASDAQ) index from January 1972 to December 1996. Second, by including a further MA algorithm, called Moving Average with Trading Volume (MAV). The results support the results of Brock et al. (1992) showing that the suggested algorithms outperform BH.

Chang et al. (2004) test whether returns generated by VMA, FMA and TRB are predictable in eleven emerging stock markets in the US and Japan considering data from January 1991 to January 2004. Predictability is analyzed by means of multivariate variance ratios using bootstrap procedures. VMA, FMA and TRB are employed and compared to BH. Results show that there is some evidence of forecasting power but no significance. When trading costs are taken into account only a few variants of the algorithms generate excess returns. Chang et al. (2004) conclude that although the algorithms show some predictive ability this is not statistically significant. Hence, Chang et al. (2004) check for robustness by analyzing returns from 1559 different variants of the algorithms, testing different sub-samples, and analyzing bear and bull markets. Overall the algorithms do not
seem to have predictive power for the recent sample used.

Bokhari et al. (2005) investigate the predictive ability and profitability of \textit{VMA}, \textit{FMA} and \textit{TRB} for different company sizes considering different indices form January 1987 to July 2002. Results on different Financial Times Stock Exchange (FTSE) indices, namely FTSE 100, FTSE 250 and FTSE Small Cap, show that the algorithms have a progressively higher predictive ability the smaller the size of the company, but are not profitable assuming transaction costs.

Marshall and Cahan (2005) test the profitability of twelve variants of \textit{VMA}, \textit{FMA} and \textit{TRB} on the New Zealand equity market. The nature and regulations suggest that the New Zealand equity market may be less efficient than large markets in Europe or the US. This raises the possibility that the algorithms are profitable in New Zealand. Using a bootstrap procedure, the results show that the returns achieved in New Zealand follow a similar pattern than those in large markets.

Ming-Ming and Siok-Hwa (2006) test the profitability of \textit{VMA}, \textit{FMA} and \textit{TRB} on nine Asian stock market indices from January 1988 to December 2003. The results provide strong support for \textit{VMA} and \textit{FMA} in China, Thailand, Taiwan, Malaysia, Singapore, Hong Kong, Korea, and Indonesia.

Hatgioannides and Mesomeris (2007) aim to characterize the stock return dynamics of four Latin American and four Asian emerging capital market economies and test the profitability of \textit{VMA} and \textit{TRB}. Using the Morgan Stanley Capital International (MSCI) index \textit{BH} is outperformed in all markets before transaction costs, and in Asian markets after transaction costs.

Lento and Gradojevic (2007) test the profitability of different algorithms by evaluating their ability to outperform \textit{BH}. Different \textit{VMA}, \textit{FMA}, Filter rule, Bollinger Band, and \textit{TRB} algorithms are tested on the S&P/TSX 300 Index, the DJIA, the NASDAQ Composite Index, and the Canada/U.S. spot exchange rate. A bootstrap procedure is used to determine the statistical significance of the results. Considering transaction costs, excess returns are generated by \textit{VMA}, \textit{FMA} and \textit{TRB} for all markets except DJIA.

Lagoarde-Segot and Lucey (2008) test the Efficient Market Hypothesis (\textit{EMH}) in seven emerging Middle-Eastern North African (MENA) stock markets from January 1998 to December 2004. The results of a random-walk test, and the returns of \textit{VMA}, \textit{FMA} and \textit{TRB} are aggregated into a single efficiency index. The impact of market development, corporate governance and economic liberalization on the latter using a multinomial ordered logistic regression is to be analyzed. The results highlight heterogeneous levels of efficiency in the MENA stock markets. The efficiency index seems to be affected by market depth, although corporate governance factors also have predictive power. By contrast, the impact of overall economic liberalization does not appear significant.
Tabak and Lima (2009) investigate the predictive power of VMA, FMA and TRB for the Brazilian exchange rate from 2003 to 2006. A bootstrap procedure is employed to test for predictability. Furthermore, the ability of the algorithms to generate significant higher returns compared to BH is tested. Results show that the excess return generated by the algorithms is not significant, suggesting that predictability is not economically significant. Their results are consistent with those of Chang et al. (2004).

In the next two chapters a selection of preemptive and non-preemptive online conversion algorithms is presented in detail. The results of the empirical evaluation of those algorithms are given in Chapter 6.

References for Chapter 3


REFERENCES FOR CHAPTER 3


Welch, B.: 1947, The generalization of ‘student’s’ problem when several different population variances are involved, Biometrika 34(1/2), 28–35.


Chapter 4

Selected Non-preemptive Algorithms

Non-preemptive conversion algorithms are represented by one single number which specifies when to buy or sell an asset. For each observed price the algorithm must decide to convert ‘all or nothing’. In the following one guaranteeing algorithm and two heuristic algorithms from the literature are presented in detail. This chapter is used as the theoretical basis for the implementation and the experimental analysis of the algorithms presented.

4.1 The Uni-directional Algorithm of El-Yaniv (1998)

El-Yaniv (1998) suggests an uni-directional algorithm to solve the max-search problem presented in Section 2.2.1. Mohr and Schmidt (2008a,b) extend this algorithm to bi-directional search in order to buy at low prices and to sell at high prices. The original algorithm and its extension are presented in the following.

4.1.1 The Guaranteeing Algorithm

El-Yaniv (1998) provides an elegant algorithm for uni-directional non-preemptive conversion with \( m \) and \( M \) known. The algorithm is called reservation price policy (RPP) (El-Yaniv, 1998, p. 34).\(^\text{23}\)

Algorithm 4. Accept the first price greater than or equal to \( q^* = \sqrt{M \cdot m} \).

El-Yaniv (1998) assumes that prices \( q_t \) \( (t = 1, \ldots, T) \) are chosen by \( \text{OPT} \) from the real interval \([m, M]\) with \( m \leq q_t \leq M \), \( \varphi = \frac{M}{m} \), and \( 0 \leq m < M \). To solve the max-search problem, \( \text{ON} \) is searching for the maximum price in a sequence of prices of unknown length \( T \) that unfolds sequentially. Each point of time \( t \) \( \text{ON} \) obtains a

\(^{23}\)The RPP can also be found in El-Yaniv et al. (2001, p. 107).
price quotation \( q_t \) after which he must immediately decide whether to accept the price \( q_t \), or to continue observing prices. Search is closed when \( ON \) accepts some price.

We call \( q^* \) the reservation price (RP), and its deviation is done by the ‘error balancing argument’ (Borodin and El-Yaniv, 1998, p. 267). The optimal \( q^* \) under worst-case assumptions should balance the ratio ‘best-case to worst-case’. Two cases must be considered: 1) the computed \( q^* \) is too low, or 2) the computed \( q^* \) is too high. A clever adversary with complete knowledge of the future, and \( q^* \), can use this information to exploit the algorithm making the RPP perform worse, as shown in the following. Two errors, concerning the maximum price encountered, might occur in case of \textit{max-search}:

1) \textit{Too-early error}: If \( q^* \) is too low, then \( OPT \) provides an input sequence in such format that prices \( q_t \in [q^*, M] \), and thus \( ON \) may suffer from the so called ‘too early error’: \( ON \) could have achieved \( M \) but gets \( q^* \) in the worst-case. The competitive ratio achieved thus will be \( c_1 = \frac{M}{q^*} \).

2) \textit{Too-late error}: If \( q^* \) is too high, then \( OPT \) provides an input sequence in such format that prices \( q_t \in [m, q^*] \), and thus \( ON \) may suffer from the ‘too late error’: \( ON \) could have achieved \( q^* \), and gets \( m \) in the worst-case. The competitive ratio achieved thus will be \( c_2 = \frac{q^*}{m} \).

\( ON \) must choose a \( q^* \) while balancing the two errors, i.e. to ensure that

\[
\frac{c_1}{M} = \frac{c_2}{q^*} = \frac{q^*}{m} = \sqrt{M \cdot m}.
\]

The above reservation price policy is optimal for both finite and infinite time horizons, and when duration \( T \) is known or unknown (El-Yaniv, 1998, p. 35), resulting in a competitive ratio as given in Theorem 1.

\textbf{Theorem 1.} Algorithm 4 is \( \sqrt{\phi} \) competitive.

Worst-case analysis is done in the following. To proof Theorem 1 we assume \textit{max-search} is carried out once \((p = 1)\).

\subsection*{4.1.2 Worst-Case Analysis}

\textbf{Proof of Theorem 1 for Algorithm 4:} Assume \( q_t \in [q^*, M] \). Then \( ON \) sells once at a price \( q_t \geq q^* \). Then the maximum possible price \( OPT \) achieves is \( M \).
With this, from equation (2.7) the competitive ratio for max-search equals

\[ c_{\text{max}}(1) = \frac{OPT}{ON} = \frac{M}{q_t} \geq \frac{M}{\sqrt{(M \cdot m)}} = \sqrt{\frac{M}{m}} = \sqrt{\varphi}. \] (4.2)

Further assume \( q_t \in [m, q^*] \), i.e. no price \( q_t \geq q^* \) appears. Then \( ON \) must sell at the last possible price \( q_T \) which is \( m \) in the worst-case. Then the maximum possible price \( OPT \) achieves is \( q^* - \varepsilon \) and, thus

\[ c_{\text{max}}(1) = \frac{OPT}{ON} = \frac{q^* - \varepsilon}{m} > \frac{\sqrt{(M \cdot m)}}{m} = \sqrt{\frac{M}{m}} = \sqrt{\varphi}. \] (4.3)

The value \( \sqrt{\varphi} \) measures the competitive ratio for max-search under worst-case assumptions in terms of maximum and minimum prices. From this follows that the reservation price policy suggested by El-Yaniv (1998) is \( \sqrt{\varphi} \)-competitive.

### 4.2 Extension to Bi-directional Search of Mohr and Schmidt (2008a)

Mohr and Schmidt (2008a,b) extend the uni-directional reservation price algorithm for selling of El-Yaniv (1998) (cf. Section 4.1) to buying and selling, i.e. introduce a rule for min-search.

#### 4.2.1 The Guaranteeing Algorithm

The above results can be transferred to bi-directional search if we modify the reservation price policy. The optimal deterministic bi-directional algorithm is the following RPP (Mohr and Schmidt, 2008a,b):

**Algorithm 5.** Buy at the first price smaller than or equal to, and sell at the first price greater than or equal to reservation price \( q^* = \sqrt{M \cdot m} \).

Algorithm 5 is denoted by SQRT, and results in a competitive ratio as given in Theorem 2.

**Theorem 2.** Algorithm 5 is \((\frac{M}{m})^p\) competitive.

The deviation of the competitive ratio for bi-directional search, as given in Theorem 2, assuming \( p \geq 1 \) trades is presented in the following.
4.2.2 Worst-Case Analysis

When bi-directional search is carried out, the competitive ratio is measured in terms of the (overall) return achieved.

Assume that for each of the \( p \geq 1 \) trades algorithm SQRT has to consider a worst-case time series \( Q = \left( \sqrt{(M(i) \cdot m(i))}, m(i), M(i), \sqrt{(M(i) \cdot m(i))}, M(i) \right) \) for buying and selling. \( M(i) \) and \( m(i) \) are upper and lower bounds of prices, with \( i = 1, \ldots, p \).

In the worst-case the algorithm SQRT buys and sells \( i \) times at reservation price(s) \( q^*(i) = \sqrt{(M(i) \cdot m(i))} \). Resulting in a worst-case geometric return of (cf. equation (3.4) and (3.5))

\[
R_{SQRT}(p) = \prod_{i=1}^{p} \frac{\sqrt{(M(i) \cdot m(i))}}{\sqrt{(M(i) \cdot m(i))}}
\]

(4.4)

iff \( q^*(i) \) is constant for each \( i \)-th trade.

\( OPT \) buys \( i \) times at minimum prices \( m(i) \), and sells \( i \) times at the maximum prices \( M(i) \). Resulting in a geometric return of (cf. equation (3.4) and (3.5))

\[
R_{OPT}(p) = \prod_{i=1}^{p} \frac{M(i)}{m(i)}
\]

(4.5)

as for each \( i \)-th trade different upper bounds \( M(i) \) and lower bounds \( m(i) \) are assumed. If \( m(i) = m \) and \( M(i) = M \) are constants, the worst-case geometric return of \( OPT \) equals

\[
R_{OPT}(p) = \left( \frac{M}{m} \right)^p
\]

(4.6)

assuming \( p \geq 1 \) trades.

**Proof of Theorem 2 for Algorithm 5:** In order to buy and sell \( p \geq 1 \) times in a row, for each \( i \)-th trade first the \textit{min-search problem} has to be solved for buying, and second the \textit{max-search problem} has to be solved for selling. Using equations (4.4) and (4.5) from equations (2.17) and (2.18) for SQRT we get a worst-case competitive ratio

\[
\frac{c_{SQRT}^{\text{wc}}(p)}{c_{SQRT}^{\text{wc}}(p)} = \frac{OPT}{SQRT} = \frac{R_{OPT}(p)}{R_{SQRT}(p)} = \prod_{i=1}^{p} \frac{M(i)}{m(i)}
\]

(4.7)
assuming different upper bounds $M(i)$ and lower bounds $m(i)$ for each $i$-th trade.

From this follows iff the lower bounds are constants ($m(i) = m$), and the upper bounds are constants ($M(i) = M$)

$$c_{SQRT}^{wcr}(p) = \left( \frac{M}{m} \right)^p$$

assuming $p \geq 1$ trades.

Alternatively, to calculate the worst-case competitive ratio for $p \geq 1$ trades of SQRT the competitive ratios for $\text{min-search}$, and for $\text{max-search}$ achievable by SQRT can be used as shown in equation (2.17).

The ratio $c_{SQRT}^{wcr}(p)$ can be interpreted as the competitive ratio the algorithm SQRT achieves when buying and selling $p \geq 1$ times under worst-case assumptions. The worst-case competitive ratio grows exponential with $p$. Compared to $OPT$ the more trades are carried out the worse SQRT gets.

4.3 The Bi-directional Algorithms of Brock, Lakonishok and LeBaron (1992)

Brock et al. (1992) introduce the algorithms Moving Average Crossover (MA) and Trading Range Breakout (TRB), which are based on technical indicators. These algorithms are of major interest in the literature, and have been empirically analyzed by several researchers, cf. Bessembinder and Chan (1995); Hudson et al. (1996); Mills (1997); Ratner and Leal (1999); Parisi and Vasquez (2000); Gunasekarage and Power (2001); Kwon and Kish (2002); Chang et al. (2004); Bokhari et al. (2005); Marshall and Cahan (2005); Ming-Ming and Siok-Hwa (2006); Hatgioannides and Mesomeris (2007); Lento and Gradojevic (2007); Lagoarde-Segot and Lucey (2008); Tabak and Lima (2009), and the overview in Section 3.4. Unfortunately, these works do not consider competitive analysis.

In the following we present the competitive analysis of MA and TRB. In general, both heuristic conversion algorithms are reservation price ($RP$) algorithms. Reservation price(s) $q^*$ are calculated based on the offered price(s) $q_t$. Using $q^*$ intersection points specifying when to buy or sell are determined.

For each $i$-th trade we assume a worst-case time series of prices containing only minimum prices $m(i)$, and maximum prices $M(i)$. At best the considered algorithm buys at price $m(i)$, and sells at price $M(i)$ resulting $i$ times in an optimum return of $OPT = M(i)/m(i)$. In the worst-case the algorithms $ON \in \{MA,TRB\}$ buy at prices $M(i)$ and sell at prices $m(i)$ $i$ times resulting in the worst possible return of $ON = m(i)/M(i) = 1/OPT$ assuming $p \geq 1$ with $i = 1,\ldots,p$. For $ON \in$
\{MA, TRB\}, from equations (2.17) and (2.18), we get a worst-case competitive ratio
\[ c_{ON}^{wc}(p) = \prod_{i=1}^{p} \left( \frac{M(i)}{m(i)} \right)^2, \quad (4.9) \]
and in case \( m(i) = m \) and \( M(i) = M \) are constants
\[ c_{ON}^{wc}(p) = \left( \frac{M}{m} \right)^{2p}. \quad (4.10) \]

To prove the competitive ratio given in equation (4.10) we assume that \( ON \in \{MA, TRB\} \) is allowed to trade only once \((p = 1)\).

**Theorem 3.** The worst-case competitive ratio of the heuristic conversion algorithms MA and TRB equals \( (\frac{M}{m})^{2p} \).

The deviation of the competitive ratio for bi-directional search, as given in Theorem 3, assuming \( p = 1 \), is presented in the following.

### 4.3.1 Moving Average Crossover

Assume the worst-case time series \( Q = (m, \ldots, m, M, m, \ldots, m) \). Hence, the prices \( q_1, \ldots, q_{t^* - 1} = m, q_{t^*} = M, \) and \( q_{t^* + 1}, \ldots, q_T = m \). The MA algorithm suggested by Brock et al. (1992) is:

**Algorithm 6.** Buy on day \( t \) if \( MA(S)_t > uB(L)_t \) and \( MA(S)_{t-1} \leq uB(L)_{t-1} \), and sell on day \( t \) if \( MA(S)_t < lB(L)_t \) and \( MA(S)_{t-1} \geq lB(L)_{t-1} \).

Where \( MA(S)_t \) is a short moving average, \( MA(L)_t \) a long moving average \((S < L)\), and the value \( n \in \{L, S\} \) defines the number of previous data points (days) considered to calculate \( MA(n)_t = \sum_{i=t-n+1}^{t} \frac{q_i}{n} \). Prices \( q_t \) are lagged by bands, the upper band \( uB(L)_t = MA(L)_t \cdot (1+\delta) \), and the lower band \( lB(L)_t = MA(L)_t \cdot (1-\delta) \) with \( \delta \in [0.00, \infty] \).

### 4.3.2 Worst-Case Analysis

**Proof of Theorem 3 for Algorithm 6:** Assume \( S = 1, L \leq (t^* - 1), \) and \( \delta = 0.00 \). This corresponds to increasing prices generating a buy signal if the price crosses the long MA from below. Similarly, this corresponds to decreasing prices generating a sell signal if the price crosses the long MA from above. Then MA

1. buys on day \( t^* \) at price \( q_{t^*} = M \). Because
   \[ MA(1)_{t^*} = q_{t^*} = M > uB(t^*-1)_{t^*} = MA(t^*-1)_{t^*} = \frac{(t^*-2)m+M}{(t^*-1)} < M, \phantom{=} \]
   \[ MA(1)_{t^*-1} = q_{t^*-1} = m \leq uB(t^*-1)_{t^*-1} = MA(t^*-1)_{t^*-1} = \frac{(t^*-1)m}{(t^*-1)} = m. \]
2. sells on day $t^* + 1$ at price $q_{t^*+1} = m$. Because
\[ MA(t^*) = q_{t^*} = M \geq IB(t^* - 1)_{t^*} = MA(t^* - 1)_{t^*} = \frac{(t^*-2)m+M}{(t^*-1)} < M. \]
Taking these decisions into account $MA$ achieves a return of $m/M$. Comparing this to the optimum return achieved by $OPT$, the worst-case competitive ratio equals $c^w_{MA}(1) = OPT/MA = (M/m)^2$. ■

4.3.3 Trading Range Breakout

Assume the worst-case time series $Q = (m + \epsilon, \ldots, m + \epsilon, M, m, \ldots, m)$. Hence, the prices $q_1, \ldots, q_{t^*-1} = m + \epsilon$, $q_{t^*} = M$, and $q_{t^*+1}, \ldots, q_T = m$. The TRB algorithm suggested by Brock et al. (1992) is:

**Algorithm 7.** Buy on day $t$ if $q_t > uB(n)_t$, and $q_{t-1} \leq uB(n)_{t-1}$, and sell on day $t$ if $q_t < IB(n)_t$ and $q_{t-1} \geq IB(n)_{t-1}$.

Where lower band $IB(n)_t = q^\min_t(n) \cdot (1 - \delta)$ with $q^\min_t(n) = \min \{q_i | i = t - n, \ldots, t - 1\}$, and upper band $uB(n)_t = q^\max_t(n) \cdot (1 - \delta)$ with $q^\max_t(n) = \max \{q_i | i = t - n, \ldots, t - 1\}$ where $\delta \in [0.00, \infty]$, and $n < t$ is the number of previous data points (days) considered.

4.3.4 Worst-Case Analysis

**Proof of Theorem 3 for Algorithm 7:** Assume $n \leq (t^* - 2)$, and $\delta = 0.00$. This corresponds to increasing prices generating a buy signal if the price crosses the upper band from below. Similarly, this corresponds to decreasing prices generating a sell signal if the price crosses lower band from above. Then TRB

1. buys on day $t^*$ at price $q_{t^*} = M$. Because
\[ q_{t^*} = M > UB(t^* - 2)_{t^*} = q^\max_{t^*}(t^* - 2) = \max \{q_i | i = 2, \ldots, t^* - 1\} = m + \epsilon, \]
and
\[ q_{t^*-1} = m + \epsilon \leq UB(t^* - 2)_{t^*-1} = q^\max_{t^*-1}(t^* - 2) = \max \{q_i | i = 1, \ldots, t^* - 2\} = m + \epsilon. \]

2. sells on day $t^* + 1$ at price $q_{t^*+1} = m$. Because
\[ q_{t^*+1} = m < IB(t^* - 2)_{t^*+1} = q^\min_{t^*+1}(t^* - 2) = \min \{q_i | i = 3, \ldots, t^*\} = m + \epsilon, \]
and
\[ q_{t^*} = M \geq IB(t^* - 2)_{t^*} = q^\min_{t^*}(t^* - 2) = \min \{q_i | i = 2, \ldots, t^* - 1\} = m + \epsilon. \]
Taking these decisions into account $TRB$ achieves a return of $m/M$. Comparing this to the optimum return achieved by $OPT$, the worst-case competitive ratio equals $c^w_{TRB}(1) = OPT/TRB = (M/m)^2$. ■
References for Chapter 4


Chapter 5

Selected Preemptive Algorithms

Preemptive algorithms allow to determine a function for conversion. An asset can be converted 'little by little' sequentially in parts, each part at a different price. In the following one uni-directional and two bi-directional preemptive online conversion algorithms from the literature are presented in detail. This chapter is used as the theoretical basis for the implementation and the experimental analysis of the algorithms presented.

5.1 The Uni-directional Algorithm of El-Yaniv, Fiat, Karp and Turpin (1992)

El-Yaniv et al. (1992) apply online algorithms to currency conversion, using competitive analysis as performance measure. The authors focus on uni-directional preemptive conversion: $ON$ is given the task of converting an asset $D$ into asset $Y$ while it is forbidden to convert $Y$ already purchased back into $D$. The amount $s_t$ of $D$ to be converted into $Y$ on days $t = 1, \ldots, T$ must be determined such that the amount of $Y$ is maximized on day $T$, and $\sum_{t=1}^{T} s_t = 1$. El-Yaniv et al. (1992) distinguish two cases:

1. Continuous case: The price fluctuates during the investment horizon, and $ON$ may convert continuously, i.e. at any moment.

2. Discrete case: One price is announced on each trading day $t$ and remains fixed throughout $t$, i.e. $ON$ converts at discrete time steps.

For both cases the suggested algorithm is identical. Thus, as in El-Yaniv et al. (2001), we do not differ between the continuous case and the discrete case in the following. We assume that at any point of time $t$ there is a price $q_t$ offered to
To solve the max-search problem the following algorithm is suggested by El-Yaniv et al. (1992, 2001).

### 5.1.1 The Guaranteeing Algorithm

The suggested online conversion algorithm is based on the assumption that there exists a threat that at some stage during the time interval, namely on day $k \leq T$, the offered price will drop to a minimum level $m$, and will remain there until the last day $T$. A worst-case time series of prices $Q = (q_1, q_2, \ldots, q_k, m, m, \ldots, m)$, where $t = 1, \ldots, k \leq T$, is assumed. For a start, assume that the worst-case competitive ratio $c$ is known to ON.\(^{24}\) The proposed algorithm is commonly referred to as the threat-based strategy, and the basic rules are (El-Yaniv et al., 1992, 2001, p. 3; p. 109):

#### Algorithm 8.

**Rule (1).** Consider a conversion from asset $D$ into asset $Y$ only if the current price offered is the highest seen so far.

**Rule (2).** Whenever you convert asset $D$ into asset $Y$, convert just enough $D$ to ensure that a competitive ratio $c$ would be obtained if an adversary dropped the price to the minimum possible price, and kept it there throughout the game.\(^{25}\)

**Rule (3).** On the last trading day $T$, all remaining $D$ must be converted into $Y$, possibly at the minimum price.

As long as the first price $q_1 \leq c \cdot m$ Algorithm 8 does not convert any $D$ into $Y$ (except of course on the last day $T$). Thus, El-Yaniv et al. (2001, p. 111) assume $m \cdot c \leq q_1 < q_2, \ldots, < q_k \leq M$ where $c$ is the target competitive ratio. This follows from **Rule (3):** A competitive ratio of $c$ is always attainable when the maximum price is $c \cdot m$, even if the whole asset $D$ is converted at the minimum $m$ (El-Yaniv et al., 2001, Remark 5, p. 110)

\[
\frac{OPT}{ON} = \frac{c \cdot m}{m} = c.
\]

El-Yaniv et al. (1992, 2001) suggest four variants of the threat-based algorithm; each converts according to **Rules (1) to (3)** given in Algorithm 8, but the worst-case competitive ratios differ depending on the assumed a-priori knowledge of $ON$:

---

\(^{24}\)For clarity, we denote the worst-case competitive ratio by $c$ within this chapter.

\(^{25}\)The ‘minimum possible price’ is defined with respect to the information known to ON. Which is $m$ if $m$ is known and is $q_t/\varphi$ if only $\varphi = M/m$ is known, and $q_t$ is highest price seen so far.
The Uni-directional Algorithm of El-Yaniv, Fiat, Karp and Turpin (1992) 93

**Variant 1a:** 26 Upper and lower bounds of prices, $M$ and $m$, known: Threat($m, M$)

**Variant 1b:** 27 Upper and lower bounds of prices, $M$ and $m$, as well as first price $q_1$ known: Threat($m, M, q_1$)

**Variant 2:** 28 Upper and lower bounds of prices, $M$ and $m$, as well as the number of trading days $k \leq T$ known: Threat($m, M, k$)

**Variant 3:** 29 Maximum price fluctuation ratio $\varphi = \frac{M}{m}$ as well as the number of trading days $k \leq T$ known: Threat($\varphi, k$)

**Variant 4:** 30 Maximum price fluctuation ratio $\varphi = \frac{M}{m}$ known: Threat($\varphi$)

El-Yaniv et al. (1992) analyze Variants 1 to 4 under worst-case assumptions. Without loss of generality, an optimal offline adversary ($OPT$) is considered that increases the offered prices $q_t$ from $q_1 \geq m$ continuously up to the maximum possible price $q_k \leq M$ with $1 \leq k \leq T$ (El-Yaniv et al., 1992). Threat is that the price drops to $m$ for the ‘rest’ of the time interval, i.e. $q_{k+1}, \ldots, q_T = m$. Thus, the worst-case time series $Q$ with $m \leq q_1 < \ldots < q_k \leq M$ and $k \leq T$ must be considered. It is assumed that $Q$ is monotone increasing, since both $OPT$ and $ON$ convert $D$ into $Y$ only when $q_t$ reaches a new maximum. Prices that are the same or lower than previous prices will be ignored (El-Yaniv et al., 2001, p. 111).

At the start of each trading day $t$ a price $q_t$ is offered to $ON$. Following Rules (1) to (3) given in Algorithm 8 $ON$ uses the (pre-)calculated worst-case competitive ratio $c$ to determine the amount of asset $D$ ($s_t \in [0, 1]$) to be converted into $Y$ on day $t$. $ON$ converts just enough to ensure $c$, as Rule (3) requires. On the ‘first’ day the current price is the highest seen so far, and $ON$ converts some amount of $D$ iff $q_1 \geq c \cdot m$. Thus, there exists some $s_1 \geq 0$ such that $c$ is still attainable if an amount of $s_1$ of $D$ is converted into $Y$. The chosen amount $s_1$ is such that $c$ is so far guaranteed even if there will be a permanent drop to $m$ on the next day, and no further conversions will be conducted (except for one last on day $T$ converting all remaining $D$). Similar arguments can be used to justify the choice of the subsequent amounts $s_t$, and thus Rules (1) to (3) induce a $c$-competitive algorithm (El-Yaniv et al., 2001, p. 110).

The values $d_t$ and $y_t$ denote the remaining amount of asset $D$, and the accumulated amount of asset $Y$ after the $t$-th day. The threat-based algorithm

---

26 Variant 2 in El-Yaniv et al. (2001).
27 Not discussed in El-Yaniv et al. (2001).
28 Variant 1 in El-Yaniv et al. (2001).
29 Variant 3 in El-Yaniv et al. (2001).
30 Not discussed in El-Yaniv et al. (1992).
starts with $d_0 = 1$ of $D$ and $y_0 = 0$ of $Y$, and then converts the initial amount of $D$ ‘little by little’ into $Y$.  The worst-case competitive ratio $c$ differs for Variants 1 to 4.  In the following worst-case analysis is done and the competitive ratios $c$, denoted by $c^\infty(m, M)$ and $c^\infty(m, M, q_1)$ for Variant 1, $c(m, M, k)$ for Variant 2, $c(\varphi, k)$ for Variant 3, and $c^\infty(\varphi)$ for Variant 4, are derived.

5.1.2 Worst-Case Analysis of Variant 1: Threat($m, M$) and Threat($m, M, q_1$)

Since this is the variant where the number of trading days $k \leq T$ is not given the threat-based algorithm, denoted by Threat($m, M$) and Threat($m, M, q_1$), must consider an adversary that may choose an arbitrary number of days $T \to \infty$ in the worst-case (El-Yaniv et al., 2001, p. 121).  The worst-case competitive ratio $c \in \{c^\infty(m, M), c^\infty(m, M, q_1)\}$, is fixed a-priori and does not change thereafter (El-Yaniv et al., 1992, p. 6).

For each trading day $t = 1, \ldots, k \leq T$, the values of $D$ remaining $d_t$ and $Y$ accumulated $y_t$ must always satisfy that (cf. equation (2.7))

$$\frac{OPT}{ON} = \frac{q_t}{m \cdot d_t + y_t}$$

where $ON = m \cdot d_t + y_t$ represents the performance of the threat-based algorithm Variant 1 if $OPT$ drops the price to $m$ and $q_t$ is the performance of $OPT$ for this case.

In order to meet the ratio $c$ on each day $t$ the value $d_t$ must be determined such that (Dannoura and Sakurai, 1998, p. 29) (see also Iwama and Yonezawa (1999, p. 412))

$$d_t = 1 - \frac{1}{c} \cdot \ln \frac{q_t - m}{c \cdot m - m}.$$  

The optimal $c$ must satisfy $d_t = 0$ for $q_t = M$.  For $q_t = M$ from equation (5.3) we get (El-Yaniv et al., 1992, Case 1, p. 3)

$$d_t = 1 - \frac{1}{c} \cdot \ln \underbrace{\frac{M - m}{c \cdot m - m}}_{c} = 0.$$  

This guarantees that the whole amount of asset $D$ (remaining) is converted in case the highest possible price $M$ occurs on $t$, and thus $d_t = 0$ after the $t$-th conversion.
From equation (5.4) follows that the competitive ratio $c^\infty(m, M)$ is the unique solution of $c$ (El-Yaniv et al., 2001, Formula (29), p. 122)

$$c = \ln \frac{M - m}{m \cdot (c - 1)}$$  \hspace{1cm} (5.5)

$$= \ln \frac{M}{m - 1}$$  \hspace{1cm} (5.6)

$$= \ln \frac{\phi - 1}{c - 1}.$$  \hspace{1cm} (5.7)

El-Yaniv et al. (1992) only consider the case $m = 1$, then (El-Yaniv et al., 1992, Formula (3))

$$c = \ln \left(\frac{M}{c - 1}\right).$$  \hspace{1cm} (5.8)

Note that when estimating $c$ equation (5.5) must be transformed to

$$e^c \cdot (c - 1) = \frac{M}{m} - 1,$$  \hspace{1cm} (5.9)

and then solved for $c$.

El-Yaniv et al. (1992) differ between two cases, Case 1 assumes that the first price $q_1$ is unknown, and Case 2 assumes that $q_1$ is known to $ON$. In the later work El-Yaniv et al. (2001, p. 110) only consider Case 1 as given in El-Yaniv et al. (1992). In the worst-case the pessimistic assumption $q_1 = m$ must be made. In case $q_1$ is assumed to be known a-priori, the same worst-case ratio $c$ is reached as in the case where $q_1$ is assumed to be unknown a-priori, i.e. the knowledge of $q_1$ does not improve the worst-case competitive ratio $c$. But in case $q_1$ is assumed to be known a-priori the competitive ratio, denoted by $c^\infty(m, M, q_1)$, is the unique solution of $c$ ((El-Yaniv et al., 1992, p. 3, Case 2) and (Dannoura and Sakurai, 1998, p. 29))

$$c = \begin{cases} 
\ln \frac{M}{c - 1} & q_1 \in [m, cm] \\
1 + \frac{q_1 - m}{q_1} \cdot \ln \frac{M - m}{q_1 - m} & q_1 \in [cm, M]. 
\end{cases}$$  \hspace{1cm} (5.10)

Thus, equation (5.5) holds for the case where the initial price $q_1$ is assumed to be unknown to $ON$ or $m \leq q_1 \leq c^\infty(m, M) \cdot m$ (El-Yaniv et al., 1992, p. 3). Further, depending on the value of $q_1$ the amount of $D$ remaining $d_t$ equals (El-Yaniv et al., 1992, p. 4)

$$d_t = \begin{cases} 
1 - \frac{1}{c} \cdot \ln \frac{q_t - m}{c - m} & q_1 \in [m, cm] \\
q_t - q_1 & q_1 \in [cm, M]. 
\end{cases}$$  \hspace{1cm} (5.11)

In both cases (for $q_1$ known and unknown) the amount of accumulated $Y$ on day $t$ equals

$$y_t = y_{t-1} + s_t \cdot q_t \text{ with } y_t \geq 0.$$  \hspace{1cm} (5.12)
The amount \( s_t \in [0,1] \) to be converted on day \( t \) equals
\[
s_t = d_{t-1} - d_t \text{ with } d_0 = 1
\] (5.11)
and \( d_t \) is calculated as given in equation (5.3) for \( q_1 \) unknown, and as given in equation (5.9) for \( q_1 \) known.

When considering worst-cases we assume \( q_1 = m \). Thus, unless otherwise stated, the achievable worst-case ratio of Variety 1 always means the value of equation (5.5) within this work. In case of an empirical evaluation of Variety 1 the knowledge of \( q_1 \) is of interest, then the cases considered in equation (5.8) hold. An open question is whether or not the knowledge of \( q_1 \) improves the empirical-case competitive ratio of the threat-based algorithm Variety 1. This is discussed in Section 6.4.

### 5.1.3 Worst-Case Analysis of Variety 2: Threat\((m, M, k)\)

This is the variant where the number of trading days \( k \leq T \) is assumed to be known. From this follows, the worst-case competitive ratio \( c \), denoted by \( c(m, M, k) \), is strictly increasing with \( k \leq T \), and the pessimistic assumption \( k = T \) must be made when considering worst-cases (El-Yaniv et al., 2001, p. 118). The worst-case competitive ratio \( c \) must be determined such that there will be no \( D \) left after the last conversion, i.e. \( d_T = 0 \). Analogously to Variety 1 the amount to be converted on the \( t \)-th day, with \( t = 1, \ldots, k \leq T \) equals
\[
s_t = d_{t-1} - d_t \text{ with } d_0 = 1.
\] (5.12)

From \( d_T = 0 \) follows \( s_T = d_{T-1} \) with (El-Yaniv et al., 2001, p. 113)
\[
\sum_{t=1}^{T} s_t = 1.
\] (5.13)

The overall amount of \( Y \) after day \( T \) equals
\[
y_T = \sum_{t=1}^{T} s_t \cdot q_t.
\] (5.14)

The amount of already accumulated \( Y \) on day \( t \), \( y_t \geq 0 \), equals
\[
y_t = y_{t-1} + s_t \cdot q_t
\] (5.15)
with \( y_1 = y_0 + s_1 \cdot q_1 = s_1 \cdot q_1 \) for \( t = 1 \). Further, the amount of \( D \) remaining on day \( t \), \( d_t \leq 1 \), equals
\[
d_t = d_{t-1} - s_t
\] (5.16)
with \( d_t = d_0 - s_t = 1 - s_t \) for \( t = 1 \).

El-Yaniv et al. (1992, 2001) consider max-search as discussed in Section 2.2.1. Rules (1) to (3) of Algorithm 8 ensure that at time \( t \), ‘just enough’ of asset \( D \) is converted that \( ON \) achieves a competitive ratio \( c \). Thus (cf. equation (2.7))

\[
\frac{OPT}{ON} = \frac{q_t}{y_t + m \cdot d_t} \leq c. \tag{5.17}
\]

The denominator \( y_t + m \cdot d_t \) represents the overall amount of \( Y \) \( ON \) achieves if \( OPT \) would drop \( q_{t+1} \) to \( m \), and the nominator \( q_t \) is the amount of \( Y \) \( OPT \) achieves in this case. For the case \( m = 1 \), as suggested in El-Yaniv et al. (1992), equation (5.17) reduces to

\[
\frac{OPT}{ON} = \frac{q_t}{y_t + d_t} \leq c. \tag{5.18}
\]

Following Rule (3) \( ON \) must convert the minimum \( s_t \) that satisfies equation (5.17). Solving (5.17) as an equality constraint with respect to \( s_t \) we get

\[
\frac{q_t}{c} = y_{t-1} + s_t \cdot q_t + m \cdot (d_{t-1} - s_t) = y_{t-1} + m \cdot d_{t-1} + s_t \cdot (q_t - m) = \frac{q_t - c \cdot (y_{t-1} + m \cdot d_{t-1})}{c}. \tag{5.19}
\]

From equation (5.19) we get the amount to be converted on each trading day \( s_t \) (El-Yaniv et al., 2001, Formula 27)

\[
s_t = \frac{q_t - c \cdot (y_{t-1} + m \cdot d_{t-1})}{c \cdot (q_t - m)} \tag{5.20}
\]

and for the case \( m = 1 \), as suggested in El-Yaniv et al. (1992), from equation (5.20) we get (El-Yaniv et al., 1992, Formula 4)

\[
s_t = \frac{q_t - c \cdot (y_{t-1} + d_{t-1})}{c \cdot (q_t - 1)} \tag{5.21} = \frac{q_t - q_t - 1}{c \cdot (q_t - 1)}.
\]

It remains to determine the global competitive ratio \( c \) used in equation (5.20) that is attainable by \( ON \). For every day \( t \) let \( k' = k - t + 1 \) be the number
remaining days before the price drops to $m$. Let $q_1$ be the first price of this series. Let $c^{k'}(q_1)$ be a *local* (lower bound) competitive ratio which is achievable on a sequence of $k' \leq T$ remaining prices assuming $d_t = 1$ and $y_t = 0$. The overall achievable worst-case competitive ratio $c$, with respect to $M$ and $m$, in a $k$-day time interval can be determined by maximizing $c^{k'}(q_1, \ldots, q_k)$ over all choices of $k \leq T$ (El-Yaniv et al., 2001, Formula (13))

\[
    c = \sup c^{k'}(q_1, \ldots, q_k) = \sup c^{k'}(q_1, q_k)
\]

with

\[
    c^{k'}(q_1, q_k) = 1 + \frac{q_1 - m}{q_1} \cdot (k' - 1) \cdot \left[ 1 - \left( \frac{q_1 - m}{M - m} \right)^{\frac{1}{k' - 1}} \right].
\]

(5.23)

Because $c^{k'}(q_1, q_k)$ is maximized for $q_k = M$ $\sup c^{k'}(q_1, q_k)$ reduces to $c^{k'}(q_1)$. As a result, the local competitive ratio for each remaining day $k'$, denoted by $c^{k'}(q_1)$, can be given as (El-Yaniv et al., 2001, Formula 15)

\[
    c^{k'}(q_1) = 1 + \frac{q_1 - m}{q_1} \cdot (k' - 1) \cdot \left[ 1 - \left( \frac{q_1 - m}{M - m} \right)^{\frac{1}{k' - 1}} \right].
\]

(5.24)

When calculating $c^{k'}(q_1)$ it is assumed that each day is the ‘only’ day. When $c^{k'}(q_1)$ is calculated for each remaining day $k'$ the value $c^{k'}(q_1)$ is decreasing with increasing prices $q_1$ and is minimized when $q_1 = M$, i.e. $c^{k'}(M) = 1$. In other words, on each remaining day $k'$ the value of $c^{k'}(q_1)$ would be reached if the whole asset $D$ would be converted into $Y$ on day $k'$ and the price drops to $m$ on the next day (El-Yaniv et al., 2001, p. 120).

From equation (5.24) we get the worst-case competitive ratio for Variant 2 under the assumption that each price offered $q_1, \ldots, q_k$ ($k \leq T$) is the only (first) price offered, and the $q_t$ drops to $m$ on the next day. With $m = 1$ and $k' = T$ for a fixed value of $q_1$ the ratio $c(m, M, k)$ is the unique solution, $c$, of (El-Yaniv et al., 1992, Formula 2)

\[
    c = c^{k'}(q_1)
\]

\[
    = 1 + \frac{q_1 - 1}{q_1} \cdot (T - 1) \cdot \left[ 1 - \left( \frac{q_1 - 1}{M - 1} \right)^{\frac{1}{T - 1}} \right].
\]

(5.25)

As a function of $q_1$, $c(m, M, k)$ is the unique solution, $c$, of (El-Yaniv et al., 2001, Lemma 8, Formula 26)

\[
    c = T \cdot \left[ 1 - \left( \frac{m \cdot (c - 1)}{M - m} \right)^{\frac{1}{c}} \right].
\]

(5.26)
It remains to derive the overall worst-case ratio including all past trading days. Assume a sequence of $w$ price maxima. For $T \geq k \geq 2$ the best worst-case ratio $c$ can be achieved when converting at $i = 1, \ldots, w$ price maxima, i.e. $\sum_{i=1}^{w} s_i = 1$. The competitive ratio $c$ when investing in all $w$ maxima equals (El-Yaniv et al., 1992, Formula (1))

$$c = 1 + \frac{q_1 - m}{q_1} \cdot \sum_{i=2}^{w} \frac{q_i - q_{i-1}}{q_i - m}. \quad (5.27)$$

To determine the competitive ratio achievable over $w$ days equation (5.27) must be maximized over all choices of $w \leq T$ and $q_i$ such that $m \leq q_1$ and $q_k \leq M$. For a fixed value $q_1$, the maximum is achieved when $w = T$ and $q_T = M$, and all $w$ ratios $\frac{q_{w-i}}{q_{w-m}}$ in equation (5.27) are equal ($i = 2, \ldots, w$) (El-Yaniv et al., 1992, p. 4). This leads to

$$c = 1 + \frac{q_1 - 1}{q_1} \cdot (w - 1) \cdot \left[ 1 - \left( \frac{q_1 - 1}{M - 1} \right)^{\frac{1}{w-1}} \right] \quad (5.28)$$

which equals equation (5.24) for the case $m = 1$ and $w = k'$. The detailed derivation of equation (5.28) can also be found in Damaschke et al. (2009, Lemma 3, p. 636).

By maximizing equation (5.28) as a function of $q_1$ for $T \geq k \geq 2$, the overall worst-case ratio $c$ (El-Yaniv et al., 2001)$^{31}$

$$c = w \cdot \left[ 1 - \left( \frac{m \cdot (c - 1)}{M - m} \right)^{\frac{1}{c-1}} \right] \quad (5.29)$$

which equals equation (5.26) for $w = T$.

Let $c$ be a global (upper bound) competitive ratio assuming that $q_1$ is the highest price of the whole time series, i.e. $OPT$ converts the whole amount of asset $D$ into asset $Y$ at price $q_1$, and $ON$ converts the remaining amount of asset $D$ to asset $Y$. Then from equations (5.28) and (5.29) we get (El-Yaniv et al., 2001, 1992, Formula (1); Formula (28a))

$$c = \frac{q_t}{d_{t-1} \cdot q_t + y_{t-1}} \cdot \left[ 1 + \frac{q_t - m}{q_t} \cdot \sum_{i=2}^{k} \frac{q_i - q_{i-1}}{q_i - m} \right] \quad (5.30)$$

$$= \frac{q_t}{d_{t-1} \cdot q_t + y_{t-1}} \cdot \left( 1 + \frac{q_t - m}{q_t} \cdot (k' - 1) \cdot \left[ 1 - \left( \frac{q_1 - m}{M - m} \right)^{\frac{1}{k'-1}} \right] \right)$$

$$= \frac{q_t}{d_{t-1} \cdot q_t + y_{t-1}} \cdot c^{k'}(q_1).$$

The denominator $d_{t-1} \cdot q_t + y_{t-1}$ represents the amount of $Y$ accumulated by $ON$, and the nominator $q_t$ is the amount of $Y$ achieved by $OPT$.

$^{31}$Can also be found in Fiat and Woeginger (1998, p. 336).
Summing up, which worst-case competitive ratio \( c_{ON} \) could reach depends on the following cases:

1. \( q_1 \) is a global maximum and \( OPT \) will convert the whole of asset \( D \) at price \( q_1 = M \). Then from equation (5.30) the worst-case competitive ratio equals \( c(m, M, k) = c'_{k}(q_1) \) with \( q_1 = M \).

2. \( q_1 \) is not a global maximum and \( OPT \) will convert the whole of asset \( D \) at a future price. Then from equation (5.24) we get
\[
c(m, M, k) = \max \{ c'_{k}(q_1) | k' = 1, \ldots, k \leq T \} = c_T(q_1).
\]

Having calculated the achievable worst-case competitive ratio \( c \) the amount to be converted \( s_t \) is calculated according to equation (5.20). When experiments are carried out the empirical-case competitive ratio \( c^{ec} \) of Threat\((m, M, k)\) equals \( c \) as given in equation (5.30) for \( k' = 1 \) day remaining.

### 5.1.4 Worst-Case Analysis of Variant 3: Threat\((\varphi, k)\)

This is the variant where the price fluctuation ratio \( \varphi = \frac{M}{m} \) and the number of trading days \( k \leq T \) is assumed to be known. El-Yaniv et al. (2001, p. 122) observed that the minimum price offered on day \( t \) is at least \( \frac{q_t}{\varphi} \). Therefore, the worst-case competitive ratio \( c \) can be derived as in the analysis of Variant 2 (Threat\((m, M, k)\)). When specializing to the case \( m = \frac{q_t}{\varphi} \), we get (El-Yaniv et al., 2001, Formula 38)
\[
c = \varphi \cdot \left( 1 - \frac{(\varphi - 1)^k}{(\varphi^{k/(k-1)} - 1)^{k-1}} \right), \quad (5.31)
\]

In the worst-case the adversary will choose \( k \) to be \( T \). As the worst-case ratio \( c \), denoted by \( c(\varphi, k) \), is monotone increasing with \( k \leq T \), we get (El-Yaniv et al., 2001, p. 126, Theorem 6)
\[
c(\varphi, k) = \varphi \cdot \left( 1 - \frac{(\varphi - 1)^T}{(\varphi^{T/(T-1)} - 1)^{T-1}} \right). \quad (5.32)
\]

### 5.1.5 Worst-Case Analysis of Variant 4: Threat\((\varphi)\)

Analogously to Variant 1 (Threat\((m, M)\)) the number of trading days \( k \leq T \) is not given, and Threat\((\varphi)\) must consider an adversary that may choose an arbitrary number of days \( T \to \infty \) in the worst-case (El-Yaniv et al., 2001, p. 121). The worst-case competitive ratio \( c \), denoted by \( c^\infty(\varphi) \), is thus fixed a-priori and does not change thereafter (El-Yaniv et al., 1992, p. 6). Let \( c^\infty(\varphi) = \lim_{T \to \infty} c(\varphi, k) \),
then from equation (5.32) we get (El-Yaniv et al., 2001, p. 126)

\[
\lim_{T \to \infty} \frac{(\varphi - 1)^T}{(\varphi^{T/(T-1)} - 1)^{T-1}} = (\varphi - 1) \exp \left( -\frac{\varphi \ln \varphi}{\varphi - 1} \right).
\]

(5.33)

Therefore

\[
c^\infty(\varphi) = \varphi \cdot \left( 1 - (\varphi - 1) \exp \left( -\frac{\varphi \ln \varphi}{\varphi - 1} \right) \right)
\]

(5.34)

\[
= \varphi - \frac{\varphi - 1}{\varphi^{1/(\varphi - 1)}}.
\]

It remains to compute the amount to be converted \( s_t \) for the algorithms Threat(\( \varphi, k \)) and Threat(\( \varphi \)). For both El-Yaniv et al. (1992, 2001) observed that the minimum price offered on day \( t \) is at least \( \frac{q_t}{\varphi} \). From El-Yaniv et al. (2001, Formula (5)) we know

\[
\frac{q_t}{c} = y_t + d_t \cdot (\text{minimum possible price})
\]

(5.35)

By replacing the ‘minimum possible price’ by \( \frac{q_t}{\varphi} \) we get (El-Yaniv et al., 2001, Formula (30))

\[
\frac{q_t}{c} = y_t + d_t \cdot \frac{q_t}{\varphi}
\]

\[
\Rightarrow d_t = \varphi \cdot \left( \frac{1}{c} - \frac{y_t}{q_t} \right),
\]

(5.36)

and from equation (5.20) we get the amount to be converted

\[
s_t = \frac{q_t - c \cdot (y_{t-1} + d_{t-1} \cdot \frac{q_t}{\varphi})}{c \cdot (q_t - \frac{q_t}{\varphi})}
\]

(5.37)

where \( y_t = y_{t-1} + s_t \cdot q_t \). Note that \( c \) equals \( c(\varphi) \) for algorithm Threat(\( \varphi \)), and \( c(\varphi, k) \) for algorithm Threat(\( \varphi, k \)).

In the following we give some numerical examples for the above four variants of the threat-based algorithm.

### 5.1.6 Numerical Examples for Variant 1 to 4

To ensure that the competitive ratio is never smaller than one and that not more than the remaining amount of asset \( D \) is converted Cases (1) to (3) regarding the value of the first price \( q_1 \) are derived in the following. From these cases Conditions (1) to (3) are derived. Note that as long as there has been no conversion at all, each price \( q_t \) is considered as \( q_1 \).
Case (1): $m \leq q_1 \leq c \cdot m$

From Rule (3) given in Algorithm 8 follows that a competitive ratio $c$ is only achievable when the first price is at least $c \cdot m$ (as $c \geq 1$ and $m \in [m, M]$). Then $c$ holds even if the remaining amount of $D$ is converted at price $m$. From this follows:

1. As long as $q_t = c \cdot m$, no $D$ are converted: $d_0 = 1$ and $y_0 = 0$, and thus $s_1 = 0$ (except on day $T$, when $ON$ must convert all remaining $D$ into $Y$, possibly at $m$).

2. As long as $q_t < c \cdot m$, $s_1 < 0$
   (more than the initial amount of $D$ $d_0 = 1$ would be converted).

From this Condition (1) can be stated as follows:

$$s_1 = 0 \text{ iff } q_1 \leq c \cdot m \quad (\text{El-Yaniv et al., 2001, Remark 5, p. 110}).$$

In the following we give some numerical examples for Condition (1). Consider $T = 5$ possible prices $I = (3, 2, 1.5, 4, 5)$. Only the increasing prices $q_1 = 3$, $q_4 = 4$ and $q_5 = 5$ are considered, where $M = 5$ and $m = 1.5$.

**Variant 1** for $m \leq q_1 \leq c^\infty(m, M) \cdot m$. For both cases ($q_1$ assumed to be known and unknown) the worst-case competitive ratio to decide whether $q_1 > c^\infty(m, M) \cdot m$ or not is calculated using equation (5.5) in advance, i.e. equals $c^\infty(m, M) = 1.5136$.

If price $q_1$ is assumed to be known a-priori, and $q_1 \leq c^\infty(m, M) \cdot m$ Case 1 in El-Yaniv et al. (1992) holds. Thus, we do not need to differ between the case where $q_1$ is known or unknown, as given in equations (5.8) and (5.9). The already accumulated amount of asset $Y$, $y_t$, is calculated using equation (5.10), and $s_t$ using equation (5.11). As the number of days $k \leq T$ is unknown for Variant 1 there might be some amount of asset $D$ remaining which must be converted at the last price $q_T$, possibly at $m$. From equation (5.11) thus follows $s_T = d_{T-1}$, and the amount of asset $D$ remaining, $d_T$, is calculated using equation (5.3),

Following Condition (1), if the first price $q_1$ is smaller than or equal to ($\leq$) $c^\infty(m, M) \cdot m$ the amount to be converted $s_1 = 0$. Table 5.1 gives a numerical example for $c^\infty(m, M) \cdot m = 2.2704$. For Variant 1 the achievable worst-case competitive ratio $c^w$, denoted by $c^\infty(m, M)$, must equal

$$c^\infty(m, M) = \frac{q_1}{m \cdot d_1 + y_1} \text{ with } d_1 = 1 \text{ and } y_1 = 0 \quad (5.38)$$

$$\begin{align*}
&= \frac{q_1}{m} \text{ with } q_1 = c^\infty(m, M) \cdot m \\
&= \frac{c^\infty(m, M) \cdot m}{m} \\
&= 1.5136
\end{align*}$$
The Uni-directional Algorithm of El-Yaniv, Fiat, Karp and Turpin (1992)

Table 5.1: Numerical example for Variant 1 with $q_1 = c^\infty(m, M) \cdot m$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$q_t$</th>
<th>$s_t$</th>
<th>$d_t$</th>
<th>$y_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.2704</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.7777</td>
<td>0.2223</td>
<td>3.1108</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0.2223</td>
<td>0.0000</td>
<td>4.2223</td>
</tr>
</tbody>
</table>

where the value $q_t$ is the amount of asset $Y$ OPT achieves and $m \cdot d_t + y_t$ is the amount of $Y$ achieved by ON assuming that the price drops to $m$ on day $t + 1$.

As ON accumulated 4.2223 $Y$ on day $T$ the empirical-case competitive ratio $c^{ec}$ for Variant 1 on the considered input sequence $I$ equals

$$
c^{ec} = \frac{OPT}{ON} = \frac{q_T}{m \cdot d_T + y_T} \text{ with } d_T = 0 = \frac{M}{y_T} = \frac{5}{4.2223} = 1.1842.
$$

**Variant 2** for $m \leq q_1 \leq c^{k'}(q_1) \cdot m$. For Variant 2, using equation (5.24), the value $c^{k'}(q_1)$ is calculated for each day $t$. Following Condition (1), if the ‘first’ price $q_1 \leq c^{k'}(q_1) \cdot m$ then the amount to be converted $s_1 = 0$. For the input sequence $I$ considered the value $c^{k'}(q_1) \cdot m = 1.3818 \cdot 1.5 = 2.0727$. From equation (5.20) we get $s_1 = 0$ as long as $q_1 \leq 2.0727$. For $q_1 = 2.0727$ the overall worst-case competitive ratio $c^{wc}$, denoted by $c(m, M, k)$, is given by

$$
c(m, M, k) = \max \left\{ c^{k'}(q_1) | k' = 1, \ldots, 5 \right\} = c^5(2.0727) = 1.4023.
$$

To calculate $c(m, M, k) = 1.4023$ it is assumed that the price drops to $m$ on day 2 and remains there. Table 5.2 gives a numerical example.

As ON accumulated 4.2424 $Y$ on day $T$ the empirical-case competitive ratio
CHAPTER 5 Selected Preemptive Algorithms

\[ c^{ec} = \frac{OPT}{ON} = \frac{q_T}{y_T} \text{ with } q_T = M = \frac{5}{4.2424} = 1.1786. \]

**Variant 3** for \( m \leq q_1 \leq c(\varphi, k) \cdot m \). The worst-case competitive ratio to decide whether \( q_1 \leq c(\varphi, k) \cdot m \) or not is calculated using equation (5.32), and equals \( c(\varphi, k) = 1.8040 \) for the input sequence \( I = (3, 2, 1.5, 4, 5) \).

Analogously to **Variant 2**, the already accumulated amount of asset \( Y \), \( y_t \), is calculated using equation (5.10). The amount to be converted \( s_t \) is calculated using equation (5.37), with \( s_T = d_{T-1} \). The amount of asset \( D \) remaining, \( d_t \), is calculated using equation (5.36).

Following **Condition (1)**, if the first price \( q_1 \) is smaller than or equal to \( \leq \) \( c(\varphi, k) \cdot m \) the amount to be converted \( s_1 = 0 \). Table 5.3 gives a numerical example for \( q_1 = c(\varphi, k) \cdot m = 2.7060 \). For **Variant 3** the worst-case competitive ratio \( c^{wc} \),

<table>
<thead>
<tr>
<th>( t )</th>
<th>( q_t )</th>
<th>( k' )</th>
<th>( c^{k'}(q_1) )</th>
<th>( c )</th>
<th>( s_t )</th>
<th>( d_t )</th>
<th>( y_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0727</td>
<td>5</td>
<td>1.4023</td>
<td>1.4023</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1.1786</td>
<td>1.1786</td>
<td>0.7576</td>
<td>0.2424</td>
<td>3.0303</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1.0000</td>
<td>1.1786</td>
<td>0.2424</td>
<td>0.0000</td>
<td>4.2424</td>
</tr>
</tbody>
</table>

Table 5.2: Numerical example for **Variant 2** with \( q_1 = c^{k'}(q_1) \cdot m \)

<table>
<thead>
<tr>
<th>( t )</th>
<th>( q_t )</th>
<th>( s_t )</th>
<th>( d_t )</th>
<th>( y_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.7060</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.3633</td>
<td>0.6367</td>
<td>1.4533</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0.6367</td>
<td>0.0000</td>
<td>4.6367</td>
</tr>
</tbody>
</table>

Table 5.3: Numerical example for **Variant 3** with \( q_1 = c(\varphi, k) \cdot m \)
denoted by $c(\varphi, k)$, must equal

$$
c(\varphi, k) = \frac{q_1}{m \cdot d_1 + y_t} \quad \text{with } d_1 = 1 \text{ and } y_t = 0
$$

(5.42)

$$
c(\varphi, k) = \frac{q_1}{m} \quad \text{with } q_1 = c(\varphi, k) \cdot m
$$

$$
c(\varphi, k) = \frac{c(\varphi, k) \cdot m}{m} = 1.8040
$$

where the value $q_t$ is the amount of asset $Y$ OPT achieves and $m \cdot d_t + y_t$ is the amount of $Y$ achieved by ON assuming that the price drops to $m$ on day 2.

As ON accumulated 4.6367 $Y$ on day $T$ the empirical-case competitive ratio $c^{ec}$ for Variant 3 on the considered input sequence $I$ equals

$$
c^{ec} = \frac{OPT}{ON} = \frac{y_T}{m \cdot d_T + y_T} \quad \text{with } d_T = 0
$$

$$
= \frac{M}{y_T} = \frac{5}{4.6367} = 1.0784.
$$

**Variant 4** for $m \leq q_1 \leq c(\varphi) \cdot m$. The worst-case competitive ratio to decide whether $q_1 \leq c(\varphi) \cdot m$ or not is calculated using equation (5.34), and equals $c(\varphi) = 1.9405$ for the input sequence $I = (3, 2, 1.5, 4, 5)$.

Analogously to Variant 2, the already accumulated amount of asset $Y$, $y_t$, is calculated using equation (5.10). The amount to be converted $s_t$ is calculated using equation (5.37), with $s_T = d_{T-1}$. The amount of asset $D$ remaining, $d_t$, is calculated using equation (5.36).

Following **Condition 1**, if the first price $q_1$ is smaller than or equal to ($\leq$) $c(\varphi) \cdot m$ the amount to be converted $s_1 = 0$. Table 5.4 gives a numerical example for $q_1 = c(\varphi) \cdot m = 2.9108$. For Variant 4 the achievable worst-case competitive

<table>
<thead>
<tr>
<th>$t$</th>
<th>$q_t$</th>
<th>$s_t$</th>
<th>$d_t$</th>
<th>$y_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.9108</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.3076</td>
<td>0.6924</td>
<td>1.2304</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0.6924</td>
<td>0.0000</td>
<td>4.6924</td>
</tr>
</tbody>
</table>

Table 5.4: Numerical example for **Variant 4** with $q_1 = c(\varphi) \cdot m$
ratio \( c^{ec} \), denoted by \( c(\phi) \), must equal

\[
c(\phi) = \frac{q_1}{m \cdot d_1 + y_1} \quad \text{with} \quad d_1 = 1 \quad \text{and} \quad y_1 = 0 \tag{5.44}
\]

\[
= \frac{q_1}{m} \quad \text{with} \quad q_1 = c(\phi) \cdot m
\]

\[
= \frac{c(\phi) \cdot m}{m} = 1.9405
\]

where the value \( q_t \) is the amount of asset \( Y \) OPT achieves and \( m \cdot d_t + y_t \) is the amount of \( Y \) achieved by ON assuming that the price drops to \( m \) on day \( t + 1 \).

As ON accumulated 4.6924 \( Y \) on day \( T \) the empirical-case competitive ratio \( c^{ec} \) for Variant 4 on the considered input sequence \( I \) equals

\[
c^{ec} = \frac{OPT}{ON} \quad \text{with} \quad d_T = 0
\]

\[
= \frac{m \cdot d_T + y_T}{y_T} = \frac{M}{y_T}
\]

\[
= \frac{5}{4.6924} = 1.0656.
\]

The empirical-case competitive ratio \( c^{ec} = 1.0656 \) of Variant 4 is better (smaller) than the \( c^{ec} = 1.0784 \) of Variant 3 as a smaller amount \( s_4 = 0.3076 \) is converted at \( q_4 = 4 \).

**Case (2):** \( M > q_1 > c \cdot m \)

Analogously to **Case (1)**, as the number of days \( T \) is unknown for Variant 1 and Variant 4, there might be \( d_T > 0 \) of asset \( D \) remaining which must be converted at the last price \( q_T \), possibly at \( m \). Thus, the amount of asset \( D \) remaining

\[
d_T := \begin{cases} 
\geq 0, & \text{for Variant 1, Variant 4}, \\
= 0, & \text{for Variant 2, Variant 3},
\end{cases} \tag{5.46}
\]

and from equation (5.12) follows

\[
s_T = d_{T-1}. \tag{5.47}
\]

From this **Condition (2)** can be stated as follows:

\[0 < s_1 < 1 \iff M > q_1 > c \cdot m.\]

In the following we give some numerical examples for **Condition (2)**. Consider the same example of \( T = 5 \) possible prices \( I = (3, 2, 1.5, 4, 5) \) as for **Case 1**. Only the increasing prices \( q_1 = 3, q_4 = 4 \) and \( q_5 = 5 \) are considered, \( M = 5 \), and \( m = 1.5 \).
**Variant 1 for** $M > q_1 > c^\infty(m, M) \cdot m$ **and** $q_1$ **assumed to be unknown** a-priori. As $q_1$ is assumed to be unknown equation (5.5) is used to calculate $c^\infty(m, M) = 1.5136$ in advance. The amount $d_t$ on each day $t$ is calculated using equation (5.3), and $s_t$ using equation (5.11). From this follows that $y_t$ can be calculated using equation (5.10). Table 5.5 gives an example for **Variant 1** where $q_1 = 3 > c^\infty(m, M) \cdot m = 2.2704$. As $q_1 > c_\infty \cdot m$ the amount to be converted on the first day $s_1 = 0.4402 > 0$. For **Variant 1** with $M > q_1 > c_\infty \cdot m$ with $q_1$ and $k \leq T$ assumed to be unknown, the amount of $s_T = d_{T-1} = 0.2223$ of asset $D$ is converted at $q_T = 5$.

As $ON$ accumulated 3.7821 $Y$ on day $T$ the empirical-case competitive ratio $c^{ec}$ for **Variant 1a** on the considered input sequence $I$ equals

$$ c^{ec} = \frac{OPT}{ON} = \frac{q_T}{m \cdot d_T + y_T} \text{ with } d_T = 0 = \frac{M}{y_T} = \frac{5}{3.7821} = 1.3220. $$

**Variant 1 for** $M > q_1 > c^\infty(m, M) \cdot m$ **and** $q_1$ **assumed to be known** a-priori. The worst-case competitive ratio to decide whether $q_1 > c^\infty(m, M) \cdot m$ or not is calculated using equation (5.5), i.e. equals 1.5136. If the first price $q_1$ is assumed to be known a-priori, and $q_1 > c^\infty(m, M) \cdot m$ **Case 2** in El-Yaniv et al. (1992) holds. Then from equation (5.8) we get a worst-case competitive ratio $c^\infty(m, M, q_1) = 1.4236$ based on the value of $q_1$. Equation (5.9) is used to calculate $d_t$. Further, from equation (5.11) we get $s_t$ (with $s_T = d_{T-1}$) and $y_t$ is calculated using equation (5.10). Table 5.6 gives a numerical example. For **Variant 1** with $M > q_1 > c^\infty(m, M) \cdot m$ and $q_1$ assumed to be known, the a-priori knowledge of $q_1$ leads to a higher amount $y_T$ as less $Y$ are converted at the first price $q_1$: Without knowing $q_1$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$q_t$</th>
<th>$s_t$</th>
<th>$d_t$</th>
<th>$y_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>0.4402</td>
<td>0.5598</td>
<td>1.3206</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.3375</td>
<td>0.2223</td>
<td>2.6706</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0.2223</td>
<td>0.0000</td>
<td>3.7821</td>
</tr>
</tbody>
</table>

Table 5.5: Numerical example for **Variant 1a** with $M > q_1 > c^\infty(m, M) \cdot m$ and $q_1$ assumed to be unknown a-priori
and amount of \( s_1 = 0.4402 \) of asset \( D \) is converted (cf. Table 5.5), while knowing \( q_1 \) results in a smaller amount of \( s_1 = 0.4048 \) to be converted for \( q_1 = 3 \) (cf. Table 5.6). Thus, by the knowledge \( q_1 \), a higher amount of \( D \) remains to be converted at a better (higher) price. From this follows \( c^{ec}(m, M) \geq c^{ec}(m, M, q_1) \).

As \( ON \) accumulated 3.8315 \( Y \) on day \( T \) the empirical-case competitive ratio \( c^{ec} \) for Variant 1b on the considered input sequence \( I \) equals

\[
 c^{ec} = \frac{OPT}{ON} = \frac{q_T}{m \cdot d_T + y_T} \quad \text{with} \quad d_T = 0 = \frac{M}{y_T} = \frac{5}{3.8315} = 1.3050
\]

with \( c^{ec}(m, M) = 1.3220 \geq c^{ec}(m, M, q_1) = 1.3050 \).

**Variant 2 for** \( M > q_1 > c^k(q_1) \cdot m \). For Variant 2, using equation (5.24), the value \( c^k(q_1) \) is calculated for each trading day. Following Condition (2), if a ‘first’ price \( q_1 > c^k(q_1) \cdot m \) then the amount to be converted on this day \( s_1 > 0 \). Further, as \( T \) is known for Variant 2, the amount of asset \( D \) remaining on day \( T = 5 \), \( d_5 \), is null. The worst-case competitive ratio \( c^{wc} \), denoted by \( c(m, M, k) \), equals

\[
 c(m, M, k) = \max \left\{ c^k(q_t) | k = 1, \ldots, 5 \right\} = c^5(3) = 1.3818.
\]

It is assumed that in the worst-case the price drops to \( m \) on day 2 and remains there (cf. equation (5.30)). Table 5.7 gives a numerical example for \( q_1 = 3 > c^k(q'_1) \cdot m = 2.0727 \).

<table>
<thead>
<tr>
<th>( t )</th>
<th>( q_t )</th>
<th>( s_t )</th>
<th>( d_t )</th>
<th>( y_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>0.4048</td>
<td>0.5952</td>
<td>1.2145</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.3588</td>
<td>0.2363</td>
<td>2.6498</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0.2363</td>
<td>0.0000</td>
<td>3.8315</td>
</tr>
</tbody>
</table>

Table 5.6: Numerical example for Variant 1b with \( M > q_1 > c^\infty(m, M) \cdot m \) and \( q_1 \) assumed to be known a-priori
The Uni-directional Algorithm of El-Yaniv, Fiat, Karp and Turpin (1992)

<table>
<thead>
<tr>
<th>$t$</th>
<th>$q_t$</th>
<th>$k_t'$</th>
<th>$c_t' (q_t')$</th>
<th>$c_t$</th>
<th>$s_t$</th>
<th>$d_t$</th>
<th>$y_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>1.3818</td>
<td>1.3818</td>
<td>0.4474</td>
<td>0.5526</td>
<td>1.3422</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1.1786</td>
<td>1.3270</td>
<td>0.3373</td>
<td>0.2153</td>
<td>2.6914</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1.0000</td>
<td>1.3270</td>
<td>0.2153</td>
<td>0.0000</td>
<td>3.7679</td>
</tr>
</tbody>
</table>

Table 5.7: Numerical example for Variant 2 with $M > q_1 > c_t' (q_t) \cdot m$

As $ON$ accumulated 3.7679 $Y$ on day $T$ the empirical-case competitive ratio $c^{ec}$ for Variant 2 on the considered input sequence $I$ equals

$$c^{ec} = \frac{OPT}{ON} = \frac{q_T}{y_T} \text{ with } q_T = M = \frac{5}{3.7679} = 1.3270.$$  \hfill (5.51)

**Variant 3 for** $M > q_1 > c(\varphi, k) \cdot m$. The worst-case competitive ratio to decide whether $q_1 > c(\varphi, k) \cdot m$ or not is calculated using equation (5.32), and equals $c(\varphi, k) = 1.8040$ for the input sequence $I = (3, 2, 1.5, 4, 5)$.

Analogously to Variant 2, the already accumulated amount of asset $Y$, $y_t$, is calculated using equation (5.10). The amount to be converted $s_t$ is calculated using equation (5.37), with $s_T = d_{T-1}$. The amount of asset $D$ remaining, $d_t$, is calculated using equation (5.36).

Following Condition (2), if the first price $q_1 > c(\varphi, k) \cdot m$ then the amount to be converted $s_1 > 0$. Table 5.8 gives a numerical example.

<table>
<thead>
<tr>
<th>$t$</th>
<th>$q_t$</th>
<th>$s_t$</th>
<th>$d_t$</th>
<th>$y_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>0.3633</td>
<td>0.6367</td>
<td>1.0900</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.1298</td>
<td>0.5069</td>
<td>1.6090</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0.5069</td>
<td>0.0000</td>
<td>4.1436</td>
</tr>
</tbody>
</table>

Table 5.8: Numerical example for Variant 3 with $M > q_1 > c(\varphi, k) \cdot m$

As $ON$ accumulated 4.1436 $Y$ on day $T$ the empirical-case competitive ratio
\( c^{ec} \) for Variant 3 on the considered input sequence \( I \) equals

\[
c^{ec} = \frac{OPT}{ON} = \frac{q_T}{m \cdot d_T + y_T} \text{ with } d_T = 0 = \frac{M}{y_T} = \frac{5}{4.1436} = 1.2067.
\]

**Variant 4** for \( M > q_1 > c(\varphi) \cdot m \). The worst-case competitive ratio to decide whether \( q_1 > c(\varphi) \cdot m \) or not is calculated using equation (5.34), and equals \( c(\varphi) = 1.9405 \) for the input sequence \( I = (3, 2, 1.5, 4, 5) \).

Analogously to **Variant 2**, the already accumulated amount of asset \( Y \), \( y_t \), is calculated using equation (5.10). The amount to be converted \( s_t \) is calculated using equation (5.37), with \( s_T = d_{T-1} \). The amount of asset \( D \) remaining, \( d_t \), is calculated using equation (5.36).

Following **Condition (2)**, if the first price \( q_1 > c(\varphi) \cdot m \) then the amount to be converted \( s_1 > 0 \). Table 5.9 gives a numerical example.

<table>
<thead>
<tr>
<th>( t )</th>
<th>( q_t )</th>
<th>( s_t )</th>
<th>( d_t )</th>
<th>( y_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>0.3076</td>
<td>0.6924</td>
<td>0.9228</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.1099</td>
<td>0.5825</td>
<td>1.3622</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0.5825</td>
<td>0.0000</td>
<td>4.2749</td>
</tr>
</tbody>
</table>

Table 5.9: Numerical example for **Variant 4** with \( M > q_1 > c(\varphi) \cdot m \)

As \( ON \) accumulated 4.2749 \( Y \) on day \( T \) the empirical-case competitive ratio \( c^{ec} \) for **Variant 4** on the considered input sequence \( I \) equals

\[
c^{ec} = \frac{OPT}{ON} = \frac{M}{y_T} = \frac{5}{4.2749} = 1.1696.
\]

The empirical-case competitive ratio \( c^{ec} = 1.1696 \) of **Variant 4** is better (smaller) than the \( c^{ec} = 1.2067 \) of **Variant 3** as a smaller amount \( s_1 = 0.3076 \) is converted at \( q_1 = 3 \).
Case (3): $q_1 = M$

From Rule (2) follows that if the first price to be considered, $q_1$, equals $M$ the whole amount of asset $D$ is converted into $Y$ by $OPT$. Whether the whole amount of asset $D$ is converted or not depends on the a-priori knowledge of $ON$: In case the upper bound $M$ is assumed to be known the whole asset $D$ is converted at $q_1 = M$, i.e. $s_1 = 1$. In case only the price fluctuation ratio $\varphi = \frac{M}{m}$ is known the amount to be converted $s_1 < 1$.

Condition (3) differs for Variant 1,2 and Variant 3,4.

For Variant 1 and Variant 2 Condition (3) can be stated as follows:

$s_1 = 1$ if $q_1 = M$.

For Variant 3 and Variant 4 Condition (3) can be stated as follows:

$s_1 < 1$ if $q_1 = M$.

In the following we give some numerical examples for Condition (3). Assume the input sequence $I = (5, 2, 2.5, 4, 1.5)$, i.e. $q_1 = M = 5$ and $m = 1.5$.

**Variant 1 for $M = q_1$**. Table 5.10 gives an example for Variant 1 where $q_1 = 5$. We do not differ between the case where $q_1$ is known or unknown, as in both cases the whole amount of asset $D$ is converted on the first day at $M$. Equation (5.5) is used to calculate $c^\infty(m, M) = 1.5136$ in advance.

<table>
<thead>
<tr>
<th>$t$</th>
<th>$q_t$</th>
<th>$s_t$</th>
<th>$d_t$</th>
<th>$y_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>1.0000</td>
<td>0.0000</td>
<td>5.0000</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>1.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.10: Numerical example for Variant 1 with $M = q_1$

In case $q_1$ is assumed to be unknown a-priori the amount $d_t$ on each day $t$ is calculated using equation (5.3). As $q_1 = M$ the amount to be converted on the first day $s_1 = 1$.

In case $q_1$ is assumed to be known a-priori Case 2 in El-Yaniv et al. (1992) holds. Then from equation (5.8) we get a worst-case competitive ratio $c$, denoted by $c^\infty(m, M, q_1)$, based on the value of $q_1$, i.e. $c$ equals 1.0000. Further, from equation (5.11) we get $s_t$ (with $s_T = d_{T-1}$), and $y_t$ is calculated using equation (5.10).

As $ON$ accumulated 5.0000 $Y$ on day $T$ the empirical-case competitive ratio
\( c^{ec} \) for Variant 1 on the considered input sequence \( I \) equals

\[
\begin{align*}
\text{for } V & \text{ariant 1, } \frac{OPT}{ON} \\
& = \frac{q_T}{m \cdot d_T + y_T} \quad \text{with } d_T = 0 \text{ and } y_T = y_1 \\
& = \frac{M}{y_1} \\
& = \frac{5}{5} = 1.0000.
\end{align*}
\]

**Variant 2 for \( q_1 = M \).** As the number of trading days \( k \leq T \) is known, the whole amount of asset \( D \) is converted at \( q_1 \), i.e. \( s_1 = 1 \) and \( d_1 = 0 \). Thus, the accumulated amount of \( Y \) on the last day \( k \leq T \) equals

\[
y_T = y_T + m \cdot d_T
\]

\[
= y_T
\]

\[
= y_1.
\]

For Variant 2, using equation (5.24), the value \( c^{k'}(q_1) \) is calculated for each day \( t \). Following **Condition (3)** from equation (5.20) we get \( s_1 = 1 \). For \( q_1 = M = 5 \) the worst-case competitive ratio \( c \), denoted by \( c(m, M, k) \), equals

\[
c(m, M, k) = \max \left\{ c^{k'}(q_1)|k' = 1, \ldots, 5 \right\}
\]

\[
= c^5(5)
\]

\[
= 1.0000.
\]

It is assumed that the price drops to \( m \) on day 2 and remains there. Table 5.11 gives a numerical example.

<table>
<thead>
<tr>
<th>( t )</th>
<th>( q_1 )</th>
<th>( k' )</th>
<th>( c^{k'}(q_1) )</th>
<th>( c )</th>
<th>( s_t )</th>
<th>( d_t )</th>
<th>( y_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>5.0000</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.11: Numerical example for Variant 2 with \( M = q_1 \)

As \( ON \) accumulated 5.000 \( Y \) on day \( T \) the empirical-case competitive ratio \( c^{ec} \)
for **Variant 2** on the considered input sequence \( I \) equals

\[
c^{ec} = \frac{OPT}{ON} = \frac{q_T}{y_T} \quad \text{with} \quad q_T = M = \frac{5}{5} = 1.0000.
\]

The whole amount of asset \( D \) is converted into \( Y \) on the first day, i.e. the threat-based algorithm achieves the optimum amount of \( Y \).

**Variant 3** for \( q_1 = M \). The worst-case competitive ratio to decide whether \( q_1 > c(\varphi, k) \cdot m \) or not is calculated using equation (5.32), and equals \( c(\varphi, k) = 1.8040 \) for the input sequence \( I = (5, 2, 2.5, 4, 1.5) \).

Analogously to **Variant 2**, the already accumulated amount of asset \( Y \), \( y_t \), is calculated using equation (5.10). The amount to be converted \( s_t \) is calculated using equation (5.37), with \( s_T = d_{T-1} \). The amount of asset \( D \) remaining, \( d_t \), is calculated using equation (5.36). Table 5.12 gives a numerical example.

<table>
<thead>
<tr>
<th>( t )</th>
<th>( q_t )</th>
<th>( s_t )</th>
<th>( d_t )</th>
<th>( y_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>0.3633</td>
<td>0.6367</td>
<td>1.0900</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>1.5</td>
<td>0.6367</td>
<td>0.0000</td>
<td>2.7716</td>
</tr>
</tbody>
</table>

**Table 5.12**: Numerical example for **Variant 3** with \( M = q_1 \)

As \( ON \) accumulated 2.7716 \( Y \) on day \( T \) the empirical-case competitive ratio \( c^{ec} \) for **Variant 3** on the considered input sequence \( I \) equals

\[
c^{ec} = \frac{OPT}{ON} = \frac{q_T}{m \cdot d_T + y_T} \quad \text{with} \quad d_T = 0
\]

\[
= \frac{M}{y_T} = \frac{5}{2.7716} = 1.8040.
\]

For the input sequence considered the empirical-case ratio \( c^{ec} \) equals the worst-case ratio \( c^{wc} = c(\varphi, k) = 1.8040 \) as the amount of \( s_T = 0.6367 \) of asset \( D \) must be converted at the minimum price \( m = 1.5 \), i.e. the worst-case occurs.
**Variant 4 for** \( q_1 = M \). The worst-case competitive ratio to decide whether \( q_1 > c(\varphi, k) \cdot m \) or not is calculated using equation (5.32), and equals \( c(\varphi, k) = 1.9405 \) for the input sequence \( I = (5, 2, 2.5, 4, 1.5) \).

Analogously to **Variant 2**, the already accumulated amount of asset \( Y \), \( y_t \), is calculated using equation (5.10). The amount to be converted \( s_t \) is calculated using equation (5.37), with \( s_T = d_{T-1} \). The amount of asset \( D \) remaining, \( d_t \), is calculated using equation (5.36). Table 5.13 gives a numerical example.

<table>
<thead>
<tr>
<th>( t )</th>
<th>( q_t )</th>
<th>( s_t )</th>
<th>( d_t )</th>
<th>( y_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>0.3076</td>
<td>0.6924</td>
<td>1.5308</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>1.5</td>
<td>0.6924</td>
<td>0.0000</td>
<td>2.5766</td>
</tr>
</tbody>
</table>

Table 5.13: Numerical example for **Variant 4** with \( M = q_1 \)

As \( ON \) accumulated 2.5766 \( Y \) on day \( T \) the empirical-case competitive ratio \( c^{ec} \) for **Variant 4** on the considered input sequence \( I \) equals

\[
c^{ec} = \frac{OPT}{ON} = \frac{q_T}{m \cdot d_T + y_T} \quad \text{with} \quad d_T = 0
\]

\[
= \frac{M}{y_T} = \frac{5}{2.5766} = 1.9405.
\]

For the input sequence \( I \) considered the empirical-case ratio \( c^{ec} \) equals the worst-case ratio \( c^{wc} = c(\varphi, k) = 1.9405 \) as the amount of \( s_T = 0.6924 \) of asset \( D \) must be converted at the minimum price \( m = 1.5 \), i.e. the worst-case occurs.

For all variants of Algorithm 8, in the worst-case, the pessimistic assumption \( q_1 = m \) must be made. In case \( q_1 = M \) a competitive ratio of 1 is always achieved by the threat-based algorithm **Variant 2**. Thus, when considering worst-cases,

the threat-based algorithm is optimal for **Variant 2** (El-Yaniv et al., 1992, p. 4).

\( OPT \) can get an optimum amount of \( Y \) by converting the whole amount of \( D \) at price \( M \) on day \( k \leq T \). Then from equation (2.7) the competitive ratio \( c \) for
max-search of any threat-based algorithm equals
\[ c = \frac{OPT}{ON} = \frac{M}{y_k + m \cdot d_k}. \] (5.60)

Summing up, based on the assumption of a worst-case sequence of prices, Algorithm 8 does not convert at all iff \( q_1 \leq c \cdot m \) (cf. Condition (1)). Further, Conditions (2) and (3) ensure that for \( M \geq q_1 > c \cdot m \)

1. not more than the whole amount of \( D \) is converted by the threat-based algorithm, and

2. a worst-case competitive ratio
\[ c \in \{c^\infty(m, M), c^\infty(m, M, q_1), c(m, M, k), c(\varphi, k), c^\infty(\varphi)\} \] is achievable.

El-Yaniv et al. (1992) also suggested a threat-based algorithm for bi-directional search, which is presented in the following.

### 5.2 The Bi-directional Algorithm of El-Yaniv, Fiat, Karp and Turpin (1992)

El-Yaniv et al. (1992) consider bi-directional search under the assumption that the upper and lower bounds, \( M \) and \( m \), on possible prices are known. The uni-directional threat-based algorithm Variant 1 presented in Section 5.1 is extended to bi-directional search. El-Yaniv et al. (2001, p. 136) show that, to solve the bi-directional search problem, \( ON \) does not need to know the number of trading days \( k \leq T \).

#### 5.2.1 The Guaranteeing Algorithm

\( ON \) starts with \( d_0 = 1 \) of asset \( D \) (and \( y_0 = 0 \) of asset \( Y \)) and converts back and forth between asset \( D \) and \( Y \) according to the sequence of prices which is revealed online. It is assumed that prices \( q_t \in [m, M] \) but may rise or fall arbitrarily. The overall worst-case competitive ratio \( c^{wc} \) can be calculated either by the overall amount of asset \( D \) or asset \( Y \). Thus, at the latest on the last day \( T \) of the time horizon all remaining \( s_t \) must be converted either into \( D \) or \( Y \) (possibly at price \( q_T = m \)). The bi-directional threat-based algorithm converts according to the rules given in Algorithm 8. But in case bi-directional search is carried out, the algorithm divides the sequence of prices into upward and downward runs, representing price
trends, and repeats Algorithm 8 on each run. Asset \( D \) is converted into asset \( Y \) (max-search) if the price is on an upward run, i.e. the value of \( D \) is increasing. Asset \( Y \) is converted into asset \( D \) (min-search) if the price is on a downward run, i.e. the value of \( D \) is decreasing. Worst-case analysis is done in the following.

5.2.2 Worst-Case Analysis

El-Yaniv et al. (1992) assume overall \( w \) runs, i.e. \( \frac{w}{2} \) upward runs and \( \frac{w}{2} \) downward runs, with overall \( w \) price minima and price maxima (\( i = 1, \ldots, w \)). \( OPT \) converts the whole of asset \( D \) into \( Y \) (selling) at the end of each \( i \)-th upward run (at best at \( M \)), and converts the whole of asset \( Y \) into \( D \) (buying) at the end of each \( i \)-th downward run (at best at \( m \)).

Since this is the variant where the number of trading days \( k \leq T \) is not given (only \( m, M \) are known) \( ON \) must consider an adversary that may choose an arbitrary large number of days \( T \rightarrow \infty \) in the worst-case (El-Yaniv et al., 2001, p. 121).

Assume an upward run consists of \( q_1 \leq q_2 \leq \ldots, \leq q_t \) prices, i.e. on day \( t + 1 \) with \( q_{t+1} < q_t \) the first downward run begins (El-Yaniv et al., 1992, p. 7). During these \( t \) days \( ON \) converts \( D \) into \( Y \) according to Algorithm 8, achieving a competitive ratio equal to \( c^\infty(m, M) \) in the worst-case (cf. equation (5.5)). Thus, for each trading day \( t = 1, \ldots, t \) within the upward run, the amount of \( D \) remaining \( d_t \) and the accumulated amount of \( Y \), \( y_t \), must always satisfy

\[
\frac{OPT}{ON} = \frac{q_t}{m \cdot d_t + y_t} = c^\infty(m, M) \tag{5.61}
\]

where \( ON = m \cdot d_t + y_t \) represents the performance of the threat-based algorithm if an adversary drops the price to \( m \) and \( q_t \) is the performance of \( OPT \) for this case. Thus, after day \( t \leq T \) \( ON \) has \( d_t \) of \( D \) remaining, and accumulated \( y_t \) of \( Y \). From equation (5.61) follows (El-Yaniv et al., 1992, p. 7)

\[
m \cdot d_t + y_t = \frac{q_t}{c^\infty(m, M)} \tag{5.62}
\]

\[
\Rightarrow d_t = \left( \frac{q_t}{c^\infty(m, M)} - y_t \right) \frac{1}{m}.
\]

Assume a downward run begins on day \( t + 1 \) and consists of \( q_{t+1} \leq \ldots, \leq q_k \) prices with \( k \leq T \). Then the remaining amount \( d_k \) of \( D \) at the end of a previous upward run must be converted into \( Y \) on day \( t + 1 \), i.e. on the first day of the downward run. Since \( q_{t+1} \geq m \), in the worst-case \( ON \) has at least \( \frac{q_{t+1}}{c^\infty(m, M)} \geq \frac{m}{c^\infty(m, M)} \) of asset \( Y \) at the beginning the first downward run. Beginning on day \( t + 1 \) \( ON \) converts \( Y \) into
The Bi-directional Algorithm of El-Yaniv, Fiat, Karp and Turpin (1992) 117

$D$, and all remaining $Y$ on the last day $k$ of the downward run must be converted into $D$ on the first day $k + 1$ of the next upward run. Thus, two transactions are carried out on the first day of each downward run:

1. The conversion of all remaining $D$, given by $d_t$, into $Y$, and
2. the first fraction of asset $Y$ is converted back into $D$ with a competitive ratio $c^\infty(m, M)$ as the current price is the highest seen so far.

Similarly, on the first day of each upward run, two transactions are carried out:

1. The conversion of all remaining $Y$, given by $y_t$, into $D$, and
2. the first fraction of asset $D$ is converted back into $Y$ with a competitive ratio $c^\infty(m, M)$ as the current price is the highest seen so far.

From this follows, in each of the $w$ runs the ratio between $OPT$ and $ON$ increases at most by the factor $c^\infty(m, M)$. Thus $ON$ achieves an overall worst-case competitive ratio of (El-Yaniv et al., 1992, p. 7)

$$\frac{OPT}{ON} = c^\infty(m, M)^w$$

(5.63)

assuming $m$ and $M$ are constants.

The above bi-directional algorithm is not optimal: On any upward (downward) run $ON$ can take advantage of the knowledge that, to attain a competitive ratio of $c$ in the following run, $OPT$ must begin the run with a certain price. This knowledge might lead to smaller ratio than $c^\infty(m, M)^w$ (El-Yaniv et al., 1992, p. 7). Unfortunately, El-Yaniv et al. (1992) give no description or technique how this knowledge can be used.

The competitive ratio given in equation (5.63) is an upper bound, i.e. the ratio can be improved. Let $w$ be as described above, and assume $M$ and $m$ are known. El-Yaniv et al. (1992) show that for any (unknown) number of trading days $k \leq T$ it is possible to force a competitive ratio of $c^{w/2}$ and $c$ is defined as given in equation (5.26), i.e. equals

$$c = T \cdot \left[1 - \left(\frac{m \cdot (c - 1)}{M - 1}\right)^{\frac{1}{2}}\right].$$

(5.64)

Assume $OPT$ constructs a sequence of $k \leq T$ prices consisting of only $\frac{w}{2}$ upward runs, each followed by an immediate drop to $m$: The price increases from $m$, drops to $m$, and then repeats such fluctuations (Dannoura and Sakurai, 1998, Figure 2, p. 30). $ON$ converts asset $D$ into asset $Y$ during each of the $\frac{w}{2}$ upward runs, and converts $Y$ back into $D$ at price $m$, i.e. achieves the optimum. The terminal amount of asset $D$ ($Y$) achieved by $OPT$ will exceed the terminal amount achieved
by $ON$ by at least the ratio $c$ as given in equation (5.64). Thus, in each upward run followed by a drop to $m$, the competitive ratio can be made to increase by a factor of $c$ (El-Yaniv et al., 1992, Section 4.3). This yields to a factor of $c^{w/2}$ for the entire time interval of length $T$. As $ON$ must consider an arbitrary number of days in the worst-case. For $T \to \infty$ the (lower bound) competitive ratio $c$ approaches (El-Yaniv et al., 1992, p. 7)

$$\frac{OPT}{ON} = c^\infty (m, M)^{w/2}. \quad (5.65)$$

Dannoura and Sakurai (1998) claim that the above algorithm is not optimal but induces an optimal algorithm for bi-directional search under certain restrictions on the sequence of prices. The improvement of the lower bound competitive ratio, given in equation (5.65), of above bi-directional threat-based algorithm is presented in the following.

### 5.3 Improvement Idea of Dannoura and Sakurai (1998)

Dannoura and Sakurai (1998) improve the bi-directional threat-based algorithm suggested by El-Yaniv et al. (1992), and presented in Section 5.2. The basic idea is that a better lower bound can be achieved by assuming other restrictions on the sequence of prices than El-Yaniv et al. (1992).

The lower bound competitive ratio given in El-Yaniv et al. (1992) equals $c^\infty (m, M)^{w/2}$ as given in equation (5.65). Dannoura and Sakurai (1998) improve this lower bound ratio by assuming that initially the price increases from $m_1$ (possibly to $M$), but then suddenly drops to $m_2$, where $m_1$ and $m_2$ satisfy

$$\bar{c} \cdot m_2 = m_1 \quad (5.66)$$

and

$$\bar{c} \cdot \{1 + (\bar{c} - 1) \cdot e^{\bar{c}} \cdot m\}^2 = \frac{M}{m} \quad (5.67)$$

with $m \leq m_2 < m_1 \leq M$ and $\bar{c}$ denotes the improved lower bound competitive ratio. Then, the price decreases from $m_2$ to $m$ and rises suddenly to $m_1$, and increases again from $m_1$, etc. This pattern of increasing, dropping, decreasing, rising is then repeated (Dannoura and Sakurai, 1998, Figure 3, p. 30). The optimal bi-directional algorithm against this sequence of prices differs between two cases depending on the price trend (Dannoura and Sakurai, 1998, p. 30):
Case 1. The price is on an upward run, i.e. the value of asset $D$ is increasing. Asset $D$ is converted into $Y$ (max-search) with $q_t \in [m_1, M]$ according to Algorithm 8 presented in Section 5.1. All (remaining) $D$ are converted into $Y$ when $q_t$ drops to $m_2$.

Case 2. The price is on a downward run, i.e. the value of asset $Y$ is increasing. Asset $Y$ is converted into $D$ (min-search) with $q_t \in [m, m_2]$ according to Algorithm 8 presented in Section 5.1. All (remaining) $Y$ are converted into $D$ when $q_t$ rises to $m_1$.

Assuming $w$ price minima and price maxima, the best possible competitive ratio (the improved lower bound) then equals $\bar{c}^w$. Dannoura and Sakurai (1998) show that in case exactly one upward run with $w = 1$ is assumed, the relation $\bar{c} > c^\infty(m, M)^{(1/2)}$ holds, where $c^\infty(m, M)^{(1/2)}$ is the lower bound by El-Yaniv et al. (1992) given in equation (5.65).

Further, Dannoura and Sakurai (1998) observe a gap between the achievable competitive ratio and improved the lower bound $\bar{c}^w$. Thus, they suggest to improve Algorithm 8 of El-Yaniv et al. (1992) by assuming the above sequence of prices. The improved algorithm is presented in the following.

5.3.1 The Guaranteeing Algorithm

Remember that by using the original uni-directional threat-based algorithm of El-Yaniv et al. (1992) $ON$ faces the threat that during an upward run the price $q_t$ might suddenly drop to $m$. Thus, the amount of asset $D$ converted into $Y$ is such that a worst-case competitive ratio $c^w$, denoted by $c^\infty(m, M)$, (cf. equation 5.61) is achievable if $q_t$ indeed drops to $m$. Dannoura and Sakurai (1998) assume $w = 2$ subsequent upward runs, i.e. the price increases, followed by a sudden drop to $m$, then increases again, followed by a second drop to $m$. Each upward run leading to a competitive ratio of $c^\infty(m, M)$. From equation (5.63) the overall competitive ratio then equals $c^\infty(m, M)^w = c^\infty(m, M)^2$.

Dannoura and Sakurai (1998) claim that the overall competitive ratio ratio is not $c^\infty(m, M)^2$ but $c^\infty(m, M)$ in case of bi-directional search and $w = 2$.

Assuming the above $w = 2$ subsequent upward runs, and using Rule (1) to (3) as given in Algorithm 8 to solve the bi-directional search problem, $ON$ converts $Y$ into $D$ (min-search) at the best possible rate $m$ every time the rate drops, i.e. achieves the optimum. Thus, the worst-case assumption of El-Yaniv et al. (1992), i.e. the ‘threat’ of a sudden drop to $m$, holds only for the uni-directional case when converting $D$ into $Y$. 
In the bi-directional case a sudden drop to \(m\) leads to the best possible competitive ratio \(c^* = \frac{OPT}{ON} = \frac{m}{m} = 1\) for min-search. From this follows by using Algorithm 8 for bi-directional search \(ON\) faces too much of a "threat". Thus, Dannoura and Sakurai (1998) improve the original uni-directional algorithm by making the 'threat' smaller. Like Algorithm 8 of El-Yaniv et al. (1992) the improved uni-directional algorithm consists of three rules (Dannoura and Sakurai, 1998, p. 31) and is repeated for bi-directional search. For a start, assume that the worst-case competitive ratio \(c\), denoted by \(\hat{c}\), is known to \(ON\).

Algorithm 9.

Rule (1). Consider a conversion from asset \(D\) into asset \(Y\) only if the current price offered is the highest seen so far.

Rule (2). Whenever you convert asset \(D\) into asset \(Y\), convert "just enough" \(D\) to ensure that a competitive ratio \(\hat{c}\) would be obtained if an adversary dropped the price to the minimum possible price \(\hat{c} \cdot m\), and kept it there throughout the game.\(^{32}\)

Rule (3). On the last trading day \(T\), all remaining \(D\) must be converted into \(Y\), possibly at the minimum price.

Only the second rule is modified by Dannoura and Sakurai (1998): The lower bound on the exchange rates is assumed to be \(\hat{c} \cdot m\) instead of \(m\), i.e. the threat is 'smaller' as \(\hat{c} \geq 1\). In the following worst-case analysis of Algorithm 9 is done.

5.3.2 Worst-Case Analysis

Dannoura and Sakurai (1998) improve the threat-based algorithm Variant 1 of El-Yaniv et al. (1992, 2001) assuming \(m\) and \(M\) are known. Since this is the variant where the number of trading days \(k \leq T\) is not given \(ON\) must consider an adversary that may choose an arbitrary large number of days \(T \to \infty\) in the worst-case.

In order to meet the worst-case ratio \(\hat{c}\) on each day the values \(d_t\) and \(y_t\) must be determined such that the amount of asset \(D\) equals (Dannoura and Sakurai, 1998, p. 31)

\[
d_t = 1 - \frac{1}{\hat{c}} \cdot \ln \frac{q_t - \hat{c} \cdot m}{\hat{c}^2 \cdot m - \hat{c} \cdot m}.
\]

(5.68)

Since

\[
\hat{c} \cdot m \cdot d(q_t) + y(q_t) \geq \frac{q_t}{\hat{c}}
\]

(5.69)

is satisfied \(ON\) will get at least \(\frac{q_t}{\hat{c}}\) of asset \(Y\) (under \(q_t \in [\hat{c} \cdot m, M]\)).

\(^{32}\)The 'minimum possible price' equals \(\hat{c} \cdot m\) instead of \(m\) as assumed by El-Yaniv et al. (1992).
Dannoura and Sakurai (1998) assume that the behavior of Algorithm 9 is identical to Algorithm 8 of El-Yaniv et al. (1992). Thus, the worst-case competitive ratio $\hat{c}$ achieved by Algorithm 9 equals

$$\hat{c} = \ln \frac{M - \hat{c} \cdot m}{\hat{c}^2 \cdot m - \hat{c} \cdot m} \quad (5.70)$$

When estimating $\hat{c}$ equation (5.70) must be transformed to

$$\hat{c} \cdot (\hat{c} - 1) = \frac{M}{\hat{c} \cdot m} - 1. \quad (5.71)$$

Equation (5.69) holds for the improved uni-directional algorithm and $q_t \geq \hat{c} \cdot m$ (Dannoura and Sakurai, 1998, p. 32). But, in practice, the whole amount of $D$ remaining might be converted at price $m$, e.g. on the last day $T$ of the time interval. In this case, since $d_t, y_t \geq 0$

$$m \cdot d_t + y_t \geq \frac{\hat{c} \cdot m \cdot d_t + y_t}{\hat{c}} \geq \frac{q_t}{\hat{c}^2}. \quad (5.72)$$

Thus ON will achieve at least $\frac{q_t}{\hat{c}^2}$ of asset $Y$. Dannoura and Sakurai (1998) claim that thus the overall achievable competitive ratio (the lower bound) of Algorithm 9 equals $\hat{c}^2$.

From this follows, equation (5.70) holds for the case where the initial price $q_1$ is assumed to be unknown to ON or $q_1 \leq \hat{c}^2 \cdot m$ (Dannoura and Sakurai, 1998, p. 32). This is of main interest when determining the competitive ratio under worst-case assumptions as the pessimistic assumption $q_1 = \hat{c} \cdot m$ must be made.

Analogously to the threat-based algorithm Variant 1, in case the first price $q_1 > m$ is assumed to be known a-priori, the competitive ratio, denoted by $\check{c}$, is the unique solution of (Dannoura and Sakurai, 1998, p. 31)

$$\check{c} = \begin{cases} \ln \frac{M}{\check{c} - 1} & q_1 \in [m, \check{c} \cdot m] \\ 1 + \frac{q_1 - \check{c} \cdot m}{q_1 \cdot \ln \frac{M - \check{c} \cdot m}{q_1 - \check{c} \cdot m}} & q_1 \in [\check{c} \cdot m, M]. \end{cases} \quad (5.73)$$

Further, depending on the value of $q_1$ the amount of $D$ remaining $d_t$ equals (Dannoura and Sakurai, 1998, p. 31)

$$d_t = \begin{cases} 1 - \frac{1}{\check{c} \cdot \ln \frac{q_1 - \check{c} \cdot m}{\check{c} \cdot m - \check{c} \cdot m}} & q_1 \in [m, \check{c} \cdot m] \\ \frac{q_1 - \check{c} \cdot m}{q_1 \cdot \ln \frac{q_1 - \check{c} \cdot m}{q_1 - \check{c} \cdot m}} - \frac{1}{\check{c} \cdot \ln \frac{q_1 - \check{c} \cdot m}{q_1 - \check{c} \cdot m}} & q_1 \in [\check{c} \cdot m, M]. \end{cases} \quad (5.74)$$

Then the competitive ratio $\check{c}$ is a function of $q_1$. When considering worst-cases we make no assumptions about $q_1$. Only for the empirical evaluation of Algorithm
9 the value of $q_1$ is of interest. Thus, unless otherwise stated, the ratio $\tilde{c}$ always means the value of equation (5.70).

In both cases (for $q_1$ known and unknown) the amount of accumulated $Y$ on day $t$, $y_t$, equals

$$y_t = y_{t-1} + s_t \cdot q_t \text{ with } y_t \geq 0,$$

and the amount $s_t \in [0, 1]$ to be converted on day $t$ equals

$$s_t = d_{t-1} - d_t \text{ with } d_0 = 1.$$

The amount of $D$ remaining, $d_t$, is calculated as given in equation (5.68) for $q_1$ unknown, and as given in equation (5.74) for $q_1$ known. The suggested uni-directional algorithm is not optimal, nevertheless it achieves a better performance than the original uni-directional algorithm of El-Yaniv et al. (1992) (cf. Dannoura and Sakurai, 1998, p. 32).

The improved bi-directional algorithm of Dannoura and Sakurai (1998) repeats the proposed uni-directional Algorithm 9 in a similar manner to the original method of El-Yaniv et al. (1992). Thus, the overall achievable competitive ratio (the improved upper bound) is calculated as for their bi-directional algorithm, and equation (5.63) holds. Assuming $\frac{w}{2}$ upward runs and $\frac{w}{2}$ downward runs, $ON$ achieves an overall competitive ratio of (Dannoura and Sakurai, 1998, p. 33)

$$\frac{OPT}{ON} = \tilde{c}^w$$

as the overall $w$ minima and maxima of prices are assumed.

Summing up, Dannoura and Sakurai (1998) improve the upper and lower bound for bi-directional run search given in the previous work by El-Yaniv et al. (1992). The improved algorithm is not yet optimal, thus the challenge of designing an optimal algorithm for bi-directional search remains (Dannoura and Sakurai, 1998, p. 33).

In Chapter 6 the above described threat-based algorithms are evaluated empirically assuming $p \geq 1$ trades. We compare worst-case results to empirical-case results.

References for Chapter 5


Chapter 6

Results

In this chapter selected results are given. All results are presented in the form of research papers. Each paper is provided in its originally published or submitted version. Thus, a preface links the paper to the previous chapters of this work. We consider a set-up where the price fluctuates on a day to day basis, and decisions when and how much to convert have to be made online – without any knowledge of the future prices.

6.1 Results of Mohr and Schmidt (2008)

Preface

The following two research papers investigate the performance of the uni-directional non-preemptive reservation price (RP) algorithm introduced by El-Yaniv (1998). The RP algorithm is presented in detail in Section 4.1: Algorithm 4, p. 81. To enable bi-directional search, this uni-directional RP algorithm for selling is extended to buying and selling: Mohr and Schmidt (2008a,b) introduce a rule for min-search. The resulting Algorithm 5, p. 83, and denoted by SQRT, achieves a worst-case competitive ratio as given in Theorem 2.

For the empirical-case analysis transaction costs are assumed and backtesting of algorithm SQRT is done on the German Dax-30 index for the investment horizon 01-01-2007 to 12-31-2007. Each of the 30 assets of the index can be chosen by the investigated algorithms \( ON \in \{SQRT, BH, Rand\} \) and \( OPT \). In order to trade multiple times the investment horizon is divided into time intervals of different length \( T \in \{7, 14, 28, 91, 182, 364\} \) days. The following questions are to be answered:

1. Does algorithm SQRT show a superior behavior to a classic buy-and-hold algorithm (BH)?
2. Does algorithm SQRT show a superior behavior to a randomized algorithm (Rand)?

3. How do estimates on \( m \) and \( M \) influence the performance of SQRT?

4. Which empirical-case competitive ratio \( c^e \) and which worst-case competitive ratio \( c^w \) achieves SQRT?

To answer these questions two different variants of algorithm SQRT are assumed. The first variant, denoted by ‘Historic’, uses estimates from the past to calculate a reservation price \( q^* = \sqrt{M \cdot m} \): In case of a time interval of length \( T \) days the upper and lower bounds of prices \( q_t, M \) and \( m \), are calculated by the \( T \) prices preceding the actual day \( t \). The second variant, denoted by ‘Clairvoyant’, uses precise estimates to calculate \( q^* = \sqrt{M \cdot m} \): In case of a time interval length of \( T \) days the actually observed values of \( m \) and \( M \) within each \( T \) are used. It is obvious that the better the estimates of \( m \) and \( M \) the better the performance of algorithm SQRT.

Results show that the shorter the time intervals, the better are estimates by historical \( m \) and \( M \). Summing up, Mohr and Schmidt (2008a,b) analyze multiple bi-directional conversion while trading multiple assets from an empirical-case and a worst-case point of view.

6.1.1 Mohr and Schmidt (2008a)

Digital Object Identifier (DOI): 10.1007/978-3-540-87477-5_32.
© Springer-Verlag Berlin Heidelberg 2008, published online: October 25, 2008.\(^{33}\)

\(^{33}\)The copyright permission can be found in the Appendix, cf. Section A.1 and the original publication is available at www.springerlink.com.
Empirical Analysis of an Online Algorithm for Multiple Trading Problems

Esther Mohr\textsuperscript{1} and Günter Schmidt\textsuperscript{1,2,⋆}

\textsuperscript{1} Saarland University, P.O. Box 151150, D-66041 Saarbrücken, Germany
\textsuperscript{2} University of Liechtenstein, Fürst-Franz-Josef-Strasse, 9490 Vaduz, Liechtenstein
em@itm.uni-sb.de, gs@itm.uni-sb.de

Abstract. If we trade in financial markets we are interested in buying at low and selling at high prices. We suggest an active trading algorithm which tries to solve this type of problem. The algorithm is based on reservation prices. The effectiveness of the algorithm is analyzed from a worst case and an average case point of view. We want to give an answer to the questions if the suggested active trading algorithm shows a superior behaviour to buy-and-hold policies. We also calculate the average competitive performance of our algorithm using simulation on historical data.

Keywords: online algorithms, average case analysis, stock trading, trading rules, performance analysis, competitive analysis, trading problem, empirical analysis.

1 Introduction

Many major stock markets are electronic market places where trading is carried out automatically. Trading policies which have the potential to operate without human interaction are of great importance in electronic stock markets. Very often such policies are based on data from technical analysis [8, 6, 7]. Many researchers have also studied trading policies from the perspective of artificial intelligence, software agents and neural networks [1, 5, 9].

In order to carry out trading policies automatically they have to be converted into trading algorithms. Before a trading algorithm is applied one might be interested in its performance. The performance analysis of trading algorithms can basically be carried by three different approaches. One is Bayesian analysis where a given probability distribution for asset prices is a basic assumption. Another one is assuming uncertainty about asset prices and analyzing the trading algorithm under worst case outcomes; this approach is called competitive analysis. The third one is a heuristic approach where trading algorithms are designed and the analysis is done on historic data by simulation runs. In this paper we apply the second and the third approach in combination. We consider a multiple trade problem and analyze an appropriate trading algorithm from a worst case

⋆ Corresponding author.
point of view. Moreover we evaluate its average case performance empirically and compare it to other trading algorithms.

The reminder of this paper is organized as follows. In the next section the problem is formulated and a worst case competitive analysis of the proposed trading algorithm is performed. In Section 3 different trading policies for the multiple trade problem are introduced. Section 4 presents detailed experimental findings from our simulation runs. We finish with some conclusions in the last section.

2 Problem Formulation

If we trade in financial markets we are interested in buying at low prices and selling at high prices. Let us consider the single trade and the multiple trade problem. In a single trade problem we search for the minimum price \( m \) and the maximum price \( M \) in a time series of prices for a single asset. At best we buy at price \( m \) and sell later at price \( M \). In a multiple trade problem we trade assets sequentially in a row, e.g. we buy some asset \( u \) today and sell it later in the future. After selling asset \( u \) we buy some other asset \( v \) and sell it later again; after selling \( v \) we can buy \( w \) which we sell again, etc. If we buy and sell (trade) assets \( k \) times we call the problem \( k \)-trade problem with \( k \geq 1 \).

As we do not know future prices the decisions to be taken are subject to uncertainty. How to handle uncertainty for trading problems is discussed in [3]. In [2] and [4] online algorithms are applied to a search problem. Here a trader owns some asset at time \( t = 0 \) and obtains a price quotation \( m \leq \text{price}(t) \leq M \) at points of time \( t = 1, 2, \ldots, T \). The trader must decide at every time \( t \) whether or not to accept this price for selling. Once some price \( \text{price}(t) \) is accepted trading is closed and the trader's payoff is calculated. The horizon \( T \) and the possible minimum and maximum prices \( m \) and \( M \) are known to the trader. If the trader did not accept a price at the first \( T - 1 \) points of time he must be prepared to accept some minimum price \( m \) at time \( T \). The problem is solved by an online algorithm.

An algorithm \( ON \) computes online if for each \( j = 1, \ldots, n - 1 \), it computes an output for \( j \) before the input for \( j + 1 \) is given. An algorithm computes offline if it computes a feasible output given the entire input sequence \( j = 1, \ldots, n - 1 \). We denote an optimal offline algorithm by \( OPT \). An online algorithm \( ON \) is \( c \)-competitive if for any input \( I \)

\[ ON(I) > 1/c \times OPT(I). \tag{1} \]

The competitive ratio is a worst-case performance measure. In other words, any \( c \)-competitive online algorithm is guaranteed a value of at least the fraction \( 1/c \) of the optimal offline value \( OPT(I) \), no matter how unfortunate or uncertain the future will be. When we have a maximization problem \( c \geq 1 \), i.e. the smaller \( c \) the more effective is \( ON \). For the search problem the policy (trading rule) [2]...
accept the first price greater or equal to reservation price \( p^* = \sqrt{M \cdot m} \)

has a competitive ratio \( c_s = \sqrt{M/m} \) where \( M \) and \( m \) are upper and lower bounds of prices \( p(t) \) with \( p(t) \) from \([m, M]\). \( c_s \) measures the worst case in terms of maximum and minimum price.

This result can be transferred to \( k \)-trade problems if we modify the policy to buy the asset at the first price smaller or equal and sell the asset at the first price greater or equal to reservation price \( p^* = \sqrt{M \cdot m} \).

In the single trade problem we have to carry out the search twice. In the worst case we get a competitive ratio of \( c_s \) for buying and the same competitive ratio of \( c_s \) for selling resulting in an overall competitive ratio for the single trade problem of \( c_t = c_s \). In general we get for the \( k \)-trade problem a competitive ratio of \( c_t(k) = \prod_{i=1}^{k} (M(i)/m(i)) \). If \( m \) and \( M \) are constant for all trades \( c_t(k) = (M/m)^k \). The ratio \( c_t \) can be interpreted as the rate of return we can achieve by buying and selling assets.

The bound is tight for arbitrary \( k \). Let us assume for each of \( k \) trades we have to consider the time series \((M, (M \cdot m)^{1/2}, m, m, (M \cdot m)^{1/2}, M)\). \( OPT \) always buys at price \( m \) and sells at price \( M \) resulting in a return rate of \( M/m \); \( ON \) buys at price \((M \cdot m)^{1/2}\) and sells at price \((M \cdot m)^{1/2}\) resulting in a return rate of \( 1 \), i.e. \( OPT/ON = M/m = c \). If we have \( k \) trades \( OPT \) will have a return of \((M/m)^k\) and \( ON \) of \( 1^k \), i.e. \( OPT(k)/ON(k) = (M/m)^k = c(k) \).

In the following we apply the above modified reservation price policy to multiple trade problems.

### 3 Multiple Trade Problem

In a multiple trade problem we have to choose points of time for selling current assets and buying new assets over a known time horizon. The horizon consists of several trading periods \( i \) of different types \( p \); each trading period consists of a constant number of \( h \) days. We differ between \( p = 1, 2, \ldots, 6 \) types of periods with length \( h \) from \{7, 14, 28, 91, 182, 364\} days e.g. period type \( p = 6 \) has length \( h = 364 \) days; periods of type \( p \) are numbered with \( i = 1, \ldots, n(p) \). There is a fixed length \( h \) for each period type \( p \), e.g. period length \( h = 7 \) corresponds to period type \( p = 1 \), period length \( h = 14 \) corresponds to period type \( p = 2 \), etc. For a time horizon of one year, for period type \( p = 1 \) we get \( n(1) = 52 \) periods of length \( h = 7 \), for type \( p = 2 \) we get \( n(2) = 26 \) periods of length \( h = 14 \), etc.

We may choose between three trading policies. Two elementary ones are Buy-and-Hold \((B + H)\), a passive policy, and Market Timing \((MT)\), an active policy. The third one is a random \((Rand)\) policy. As a benchmark we use an optimal offline algorithm called Market \((MA)\). We assume that for each period \( i \) there is an estimate of the maximum price \( M(i) \) and the minimum price \( m(i) \). Within each period \( i = 1, \ldots, n(p) \) we have to buy and sell an asset at least once. The annualized return rate \( R(x) \), with \( x \) from \{MT, Rand, B + H, MA\} is the
performance measure used. At any point of time of the horizon the policy either holds an asset or an overnight deposit.

In order to describe the different policies we define a holding period with respect to $MT$. A holding period is the number of days $h$ between the purchase of asset $j$ and the purchase of another asset $j'$ ($j' \neq j$) by $MT$. Holding periods are determined by either reservation prices $RP_j(t)$ which give a trading signal or when the last day $T$ of the period is reached.

**MARKET TIMING ($MT$)**

$MT$ calculates reservation prices $RP_j(t)$ for each day $t$ for each asset $j$. At each day $t$, $MT$ must decide whether to sell asset $j$ or to hold it another day considering the reservation prices. Each period $i$, the first offered price $p_j(t)$ of asset $j$ with $p_j(t) \geq RP_j(t)$ is accepted by $MT$ and asset $j$ is sold. The asset $j^*$, which is bought by $MT$ is called $MT$-asset. $MT$ chooses the $MT$-asset $j^*$ if $RP_{j^*}(t) - p_{j^*}(t) = \max\{RP_j(t) - p_j(t) | j = 1, \ldots, m\}$ and $p_{j^*}(t) < RP_{j^*}(t)$. If there was no trading signal in a period related to reservation prices then trading is done on the last day $T$ of a period. In this case $MT$ must sell asset $j$ and invest in asset $j'$ at day $T$. The holding period of $MT$ showing buying ($Buy$) and selling ($Sell$) points and intervals with overnight deposit ($OD$) is shown in Fig. 1.

**RANDOM (Rand)**

Rand will buy and sell at randomly chosen prices $p_j(t)$ within the holding period of $MT$ (cf. Fig. 1).

**BUY AND HOLD ($B + H$)**

$B + H$ will buy at the first day $t$ of the period and sell at the last day $T$ of the period.

**MARKET ($MA$)**

To evaluate the performance of these three policies empirically we use as a benchmark the optimal offline policy. It is assumed that $MA$ knows all prices $p_j(t)$ of a period including also these which were not presented to $MT$ if there were any. In each period $i MA$ will buy at the minimum price $p_{\text{min}} > m(i)$ and sell...
The performance of the investment policies is evaluated empirically. Clearly, all policies cannot beat the benchmark policy $MA$.

4 Experimental Results

We want to investigate the performance of the trading policies discussed in Section 3 using experimental analysis. Tests are run for all $p = 1, 2, \ldots, 6$ period types with the number of periods $n(p)$ from $\{52, 26, 13, 4, 2, 1\}$ and period length $h$ from $\{7, 14, 28, 91, 182, 364\}$ days. The following assumptions apply for all tested policies:

1. There is an initial portfolio value greater zero.
2. Buying and selling prices $p_j(t)$ of an asset $j$ are the closing prices of day $t$.
3. At each point of time all money is invested either in assets or in 3% overnight deposit.
4. Transaction costs are 0.0048% of the market value but between 0.60 and 18.00 Euro.
5. When selling and buying is on different days the money is invested in overnight deposit.
6. At each point of time $t$ there is at most one asset in the portfolio.
7. Each period $i$ at least one buying and one selling transaction must be executed. At the latest on the last day of each period asset $j$ has to be bought and on the last day it has to be sold.
8. In period $i = 1$ all policies buy the same asset $j$ on the same day $t$ at the same price $p_j(t)$; the asset chosen is the one $MT$ will chose ($MT_{asset}$).
9. In periods $i = 2, \ldots, n(p) - 1$ trades are carried out according to the different policies.
10. In the last period $i = n(p)$ the asset has to be sold at the last day of that period. No further transactions are carried out from there on.
11. If the reservation price is calculated over $h$ days, the period length is (also) $h$ days.

We simulate all policies using historical XETRA DAX data from the interval 2007.01.01 until 2007.12.31. This interval we divide into $n(p)$ periods where $n(p)$ is from $\{52, 26, 13, 4, 2, 1\}$ and $p$ is from $\{7, 14, 28, 91, 182, 364\}$. With this arrangement we get 52 periods of length 7 days, 26 periods of length 14 days, etc. We carried out simulation runs in order to find out...
if $MT$ shows a superior behaviour to buy-and-hold policies
(2) the influence of $m$ and $M$ on the performance of $MT$
(3) the average competitive ratio for policies for $MA$ and $MT$.

Two types of buy-and-hold policies are used for simulation; one holds the
$MT$ asset within each period ($MT_{B+H}$) and the other holds the index over all
periods ($Index_{B+H}$) of a simulation run. Thus, $MT_{B+H}$ is synchronized with
the $MT$ policy, i.e., $MT_{B+H}$ buys on the first day of each period the same asset
which $MT$ buys first in this period (possibly not on the first day) and sells this
asset on the last day (note that this asset may differ from the one $MT$ is selling
on the last day) of the period. Using this setting we compare both policies related
to the same period. $Index_{B+H}$ is a common policy applied by ETF investment
funds and it is also often used as a benchmark although it is not synchronized
with the $MT$ policy. In addition to these policies also the random policy Rand
is simulated. Rand buys the same asset which $MT$ buys on a randomly chosen
day within a holding period.

We first concentrate on question (1) if $MT$ shows a superior behaviour to the
policies $MT_{B+H}$ and $Index_{B+H}$. For calculating the reservation prices we use
estimates from the past, i.e. in case of a period length of $h$ days $m$ and $M$ are
taken from the prices of these $h$ days which are preceding the actual day $t^*$ of
the reservation price calculation, i.e. $m = \min \{p(t)|t = t^* - 1, t^* - 2, \ldots, t^* - h\}$
and $M = \max \{p(t)|t = t^* - 1, t^* - 2, \ldots, t^* - h\}$. In Table 1 the trading results
are displayed considering also transaction costs. The return rates are calculated
covering a time horizon of one year. For the three active policies ($MA$, $MT$, $Rand$) the transaction costs are the same because all follow the holding period
of $MT$; in all these cases there is a flat minimum transaction fee.

<table>
<thead>
<tr>
<th>Policy</th>
<th>1 Week</th>
<th>2 Weeks</th>
<th>4 Weeks</th>
<th>3 Months</th>
<th>6 Months</th>
<th>12 Months</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MA$</td>
<td>418.18%</td>
<td>138.40%</td>
<td>201.61%</td>
<td>47.93%</td>
<td>72.95%</td>
<td>61.95%</td>
</tr>
<tr>
<td>$MT$</td>
<td>41.08%</td>
<td>1.37%</td>
<td>54.86%</td>
<td>6.08%</td>
<td>32.39%</td>
<td>31.35%</td>
</tr>
<tr>
<td>$MT_{B+H}$</td>
<td>9.70%</td>
<td>0.50%</td>
<td>17.18%</td>
<td>15.80%</td>
<td>45.30%</td>
<td>35.29%</td>
</tr>
<tr>
<td>$Index_{B+H}$</td>
<td>20.78%</td>
<td>20.78%</td>
<td>20.78%</td>
<td>20.78%</td>
<td>20.78%</td>
<td>20.78%</td>
</tr>
<tr>
<td>Rand</td>
<td>-23.59%</td>
<td>-21.23%</td>
<td>17.18%</td>
<td>-18.23%</td>
<td>6.20%</td>
<td>15.42%</td>
</tr>
</tbody>
</table>

$MT$ dominates $MT_{B+H}$ and $Index_{B+H}$ in two cases (1 and 4 weeks). $MT_{B+H}$ dominates $MT$ and $Index_{B+H}$ in two cases (6 and 12 months). $Index_{B+H}$ dominates $MT$ and $MT_{B+H}$ in two cases (2 weeks and 3 months). $MT$ generates the best overall annual return rate when applied to 4 weeks. $MT_{B+H}$ generates the worst overall annual return rate when applied to 2 weeks. $MT_{B+H}$ policy improves its performance in comparison to $Index_{B+H}$ and $MT$ policy proportional to the length of the periods. We might conclude the longer the period the
better the relative performance of $MT_{B+H}$. $MT$ outperforms $MT_{B+H}$ in four of six cases and it outperforms $MT_{B+H}$ in three of six cases; $MT$ and $MT_{B+H}$ have the same relative performance. If the period length is not greater than 4 weeks $MT$ outperforms $MT_{B+H}$ in all cases. If the period length is greater than 4 weeks $MT_{B+H}$ outperforms $MT$ in all cases. Index$_{B+H}$ outperforms $MT_{B+H}$ in three of six cases. If we consider the average performance we have 27.86% for $MT$, 20.78% for Index$_{B+H}$, and 20.63% for $MT_{B+H}$. $MT$ is not always the best but it is on average the best. From this we conclude that $MT$ shows on average a superior behaviour to buy-and-hold policies under the assumption that $m$ and $M$ are calculated by historical data.

In general we would assume that the better the estimates of $m$ and $M$ the better the performance of $MT$. Results in Table 1 show, that the longer the periods the worse the relative performance of $MT$. This might be due to the fact that for longer periods historical $m$ and $M$ are worse estimates in comparison to those for shorter periods. In order to analyze the influence of estimates of $m$ and $M$ we run all simulations also with the observed $m$ and $M$ of the actual periods, i.e. we have optimal estimates. Results for optimal estimates are shown in Table 2 and have to be considered in comparison to the results for historic estimates shown in Table 1.

Now we can answer question (2) discussing the influence of $m$ and $M$ on the performance of $MT$. The results are displayed in Table 2. It turns out that in all cases the return rate of policy $MT$ improves significantly when estimates of $m$ and $M$ are improved. For all period lengths now $MT$ is always better than $MT_{B+H}$ and Index$_{B+H}$. From this we conclude that the estimates of $m$ and $M$ are obviously of major importance for the performance of the $MT$ policy. Now we concentrate on question (3) discussing the average competitive ratio for policies $MA$ and $MT$. We now compare the experimental competitive ratio $c_{ec}$ to the analytical competitive ratio $c_{wc}$. To do this we have to calculate $OPT$ and $ON$ for the experimental case and the worst case. We base our discussion on the return rate as the performance measure. We assume that we have precise forecasts for $m$ and $M$.

A detailed example for the evaluation of the competitive ratio is presented in Table 3 considering a period length of 12 months. In this period six trades were executed using reservation prices based on the clairvoyant test set. The analytical results are based on the values of $m$ and $M$ for each holding period.

### Table 2. Annualized returns for optimal historic estimates

<table>
<thead>
<tr>
<th>Policy</th>
<th>Annualized Returns Including Transaction Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 Week 2 Weeks 4 Weeks 3 Months 6 Months 12 Months</td>
</tr>
<tr>
<td>n(7) = 52</td>
<td>n(14) = 26 n(28) = 13 n(91) = 4 n(182) = 2 n(364) = 1</td>
</tr>
<tr>
<td>$MA$</td>
<td>418.18% 315.81% 280.94% 183.43% 86.07% 70.94%</td>
</tr>
<tr>
<td>$MT$</td>
<td>102.60% 87.90% 76.10% 81.38% 55.11% 54.75%</td>
</tr>
<tr>
<td>$MT_{B+H}$</td>
<td>9.70% -4.40% 22.31% 19.79% 45.30% 35.29%</td>
</tr>
<tr>
<td>Index$_{B+H}$</td>
<td>20.78% 20.78% 20.78% 20.78% 20.78% 20.78%</td>
</tr>
<tr>
<td>Rand</td>
<td>-23.59% -101.3% -10.67% 47.37% 46.08% 15.42%</td>
</tr>
</tbody>
</table>
Table 3. Periodic results for period length one year

<table>
<thead>
<tr>
<th>Clairvoyant Data</th>
<th>Analytical Results</th>
<th>Experimental Results</th>
</tr>
</thead>
<tbody>
<tr>
<td># Trades</td>
<td>Holding m</td>
<td>$c_{wc} = \frac{M/m}{MA/MT}$</td>
</tr>
<tr>
<td>Period</td>
<td>$MA$ buy at $Sell$</td>
<td>MA/MT Return</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$1^{st}$ trade</td>
<td>Week 1-14 37.91</td>
<td>43.23 1.1403</td>
</tr>
<tr>
<td></td>
<td>$MA$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$MT$</td>
<td></td>
</tr>
<tr>
<td>$2^{nd}$ trade</td>
<td>Week 14-24 34.25</td>
<td>38.15 1.1139</td>
</tr>
<tr>
<td></td>
<td>$MA$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$MT$</td>
<td></td>
</tr>
<tr>
<td>$3^{rd}$ trade</td>
<td>Week 24-25 13.54</td>
<td>13.69 1.0111</td>
</tr>
<tr>
<td></td>
<td>$MA$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$MT$</td>
<td></td>
</tr>
<tr>
<td>$4^{th}$ trade</td>
<td>Week 25-30 33.57</td>
<td>35.73 1.0643</td>
</tr>
<tr>
<td></td>
<td>$MA$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$MT$</td>
<td></td>
</tr>
<tr>
<td>$5^{th}$ trade</td>
<td>Week 30-46 51.23</td>
<td>58.86 1.1489</td>
</tr>
<tr>
<td></td>
<td>$MA$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$MT$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$1^{st}$ trade</td>
<td>Week 1-14 37.91</td>
</tr>
<tr>
<td></td>
<td>$MA$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$MT$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$2^{nd}$ trade</td>
<td>Week 14-24 34.25</td>
</tr>
<tr>
<td></td>
<td>$MA$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$MT$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$3^{rd}$ trade</td>
<td>Week 24-25 13.54</td>
</tr>
<tr>
<td></td>
<td>$MA$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$MT$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$4^{th}$ trade</td>
<td>Week 25-30 33.57</td>
</tr>
<tr>
<td></td>
<td>$MA$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$MT$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$5^{th}$ trade</td>
<td>Week 30-46 51.23</td>
</tr>
<tr>
<td></td>
<td>$MA$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$MT$</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Competitive ratio and annualized return rates

<table>
<thead>
<tr>
<th>Clairvoyant Data</th>
<th>Analytical Results</th>
<th>Experimental Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period Length # Trades</td>
<td>$OPT/ON$</td>
<td>MA</td>
</tr>
<tr>
<td>12 Months 6</td>
<td>1.7108</td>
<td>71.08%</td>
</tr>
<tr>
<td>6 Months 7</td>
<td>1.8624</td>
<td>86.24%</td>
</tr>
<tr>
<td>3 Months 18</td>
<td>2.8387</td>
<td>183.87%</td>
</tr>
<tr>
<td>4 Weeks 38</td>
<td>3.8185</td>
<td>281.85%</td>
</tr>
<tr>
<td>2 Weeks 48</td>
<td>4.1695</td>
<td>316.95%</td>
</tr>
<tr>
<td>1 Week 52</td>
<td>4.1711</td>
<td>317.11%</td>
</tr>
</tbody>
</table>

The analytical results are based on the consideration that $MA$ achieves the best possible return and $MT$ achieves a return of zero. E.g. for the first trade $MA$ achieves a return rate of 14.03% and $MT$ achieves a return rate of 0% i.e. $MT$ achieves absolutely 14.03% less than $MA$ and relatively a multiple of 1.1403. The experimental results are also based on the consideration that $MA$ achieves the best possible return and $MT$ now achieves the return rate generated during the experiment. E.g. for the first trade $MA$ achieves a return rate of 1.1403
or 14.03% and $MT$ achieves a return rate of 1.1322 or 13.22%. We compared the analytical results with the experimental results based on annualized return rates for the period lengths 1, 2, 4 weeks, 3, 6, and 12 months. The overall competitive ratio is based on period adjusted annual return rates. The results for all period lengths are presented in Table 4. Transaction costs are not taken into account in order not to bias results. As the policies are always invested there is no overnight deposit. E.g. For the period of 12 months the analytical worst case ratio $OPT/ON$ is 1.7108 and the average experimental ratio $MA/MT$ is 1.2950. The values of the competitive ratios for the other period lengths are also given in Table 4. The return of $MT$ reached in the experiments reaches at least 27.33%, at most 77.22% and on average 45.67% of the return of $MA$.

5 Conclusions

In order to answer the three questions from section 4 twelve simulation runs were performed. $MT$ outperforms buy-and-hold in all cases even when transaction costs are incorporated in the clairvoyant test set. Tests on historical estimates of $m$ and $M$ show that $MT$ outperforms buy-and-hold in one third of the cases and also on average. We conclude that when the period length is small enough $MT$ outperforms $B + H$.

It is obvious that the better the estimates of $m$ and $M$ the better the performance of $MT$. Results show that the shorter the periods, the better are estimates by historical $m$ and $M$. As a result, the performance of $MT$ gets worse the longer the periods become.

In real life it is very difficult to get close to the (analytical) worst cases. It turned out that the shorter the periods are the less $MT$ achieves in comparison to $MA$. A $MT$ trading policy which is applied to short periods leads to small intervals for estimating historical $m$ and $M$. In these cases there is a tendency to buy too late (early) in increasing (decreasing) markets and to sell too late (early) in decreasing (increasing) markets due to unknown overall trend directions, e.g. weekly volatility leads to wrong selling decisions during an upward trend.

The paper leaves also some open questions for future research. One is that of better forecasts of future upper and lower bounds of asset prices to improve the performance of $MT$. The suitable period length for estimating $m$ and $M$ is an important factor to provide a good trading signal, e.g. if the period length is $h$ days estimates for historical $m$ and $M$ were also be calculated over $h$ days. Simulations with other period lengths for estimating $m$ and $M$ could be of interest. Moreover, the data set of one year is very small. Future research should consider intervals of 5, 10, and 15 years.

References

302 E. Mohr and G. Schmidt


6.1.2 Mohr and Schmidt (2008b)

Digital Object Identifier (DOI): 10.1007/978-3-642-00142-0_6.
Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), University of Augsburg, September 3-5, 2008, pp. 33-38 © Springer-Verlag Berlin Heidelberg 2008, published online: August 4, 2008.\textsuperscript{34}

\textsuperscript{34}The copyright permission can be found in the Appendix, cf. Section A.1 and the original publication is available at www.springerlink.com.
Trading in Financial Markets with Online Algorithms

Esther Mohr¹ and Günter Schmidt¹²∗

¹ Saarland University, P.O. Box 151150, D-66041 Saarbrücken, Germany, em@itm.uni-sb.de
² Hochschule Liechtenstein, Fürst-Franz-Josef-Strasse, FL-9490 Vaduz, Liechtenstein, gs@itm.uni-sb.de

Summary. If we trade in financial markets we are interested in buying at low and selling at high prices. We suggest an active reservation price based trading algorithm which tries to solve this type of problem. The effectiveness of the algorithm is analyzed from a worst case point of view. We want to give an answer to the question if the suggested algorithm shows a superior behaviour to buy-and-hold policies using simulation on historical data.

1 Introduction

Many major stock markets are electronic market places where trading is carried out automatically. Trading policies which have the potential to operate without human interaction are often based on data from technical analysis [5, 3, 4]. Many researchers studied trading policies from the perspective of artificial intelligence, software agents or neural networks [1, 6]. In order to carry out trading policies automatically they have to be converted into trading algorithms. Before a trading algorithm is applied one might be interested in its performance. The performance of trading algorithms can basically be analyzed by three different approaches. One is Bayesian analysis, another is assuming uncertainty about asset prices and analyzing the trading algorithm under worst case outcomes. This approach is called competitive analysis [2]. The third is a heuristic approach where trading algorithms are analyzed by simulation runs based on historical data. We apply the second and the third approach in combination.

* corresponding author
The reminder paper is organized as follows. In the next section different trading policies for a multiple trade problem are introduced. Section 3 presents detailed experimental findings from our simulation runs. In the last section we finish with some conclusions.

2 Multiple Trade Problem

In a multiple trade problem we have to choose points of time for selling current assets and buying new assets over a known time horizon. The horizon consists of several trading periods $i$ of different types $p$ with a constant number of $h$ days. We differ between $p = 1, 2, \ldots, 6$ types of periods numbered with $i = 1, \ldots, n(p)$ and length $h$ from $\{7, 14, 28, 91, 182, 364\}$ days, e.g. period type $p = 6$ has length $h = 364$ days. There is a fixed length $h$ for each period type $p$, e.g. period length $h = 7$ corresponds to period type $p = 1$, period length $h = 14$ corresponds to period type $p = 2$, etc.

We differ between three trading policies. Two elementary ones are Buy-and-Hold ($B + H$), a passive policy, and Market Timing ($MT$), an active policy. The third one is a Random ($Rand$) policy. To evaluate the policies’ performance empirically we use an optimal algorithm called Market ($MA$) as a benchmark. We assume that for each period $i$ there is an estimate of the maximum price $M(i)$ and the minimum price $m(i)$. Within each period $i = 1, \ldots, n(p)$ we have to buy and sell an asset at least once. The annualized return rate $R(x)$, with $x$ from $\{MT, Rand, B + H, MA\}$ is the performance measure used. At any point of time a policy either holds an asset or overnight deposit.

In order to describe the different policies we define a holding period with respect to $MT$. A holding period is the number of days $h$ between the purchase of asset $j$ and the purchase of another asset $j'$ ($j' \neq j$) by $MT$. Holding periods are determined either by reservation prices $RP_j(t)$ which give a trading signal or by the last day $T$ of a period.

**MARKET TIMING ($MT$).** Calculates $RP_j(t)$ for each day $t$ for each asset $j$ based on $M(i)$ and $m(i)$. The asset $j^*$ $MT$ buys within a period is called $MT$ asset. An asset $j^*$ is chosen by $MT$ if $RP_{j^*}(t) - p_{j^*}(t) = \max \{RP_j(t) - p_j(t) | j = 1, \ldots, m\}$ and $p_{j^*}(t) < RP_{j^*}(t)$. Considering $RP_{j^*}(t)$ $MT$ must decide each day $t$ whether to sell $MT$ asset $j^*$ or to hold it another day: the first offered asset price $p_{j^*}(t)$ with $p_{j^*}(t) \geq RP_{j^*}(t)$ is accepted by $MT$ and asset $j^*$ is sold. If there was no signal by $RP_{j^*}(t)$ within a period trading must be executed at the last day $T$ of the period, e.g. $MT$ must sell asset $j^*$ and invest asset $j'$ ($j' \neq j^*$).
RANDOM (Rand). Buys and sells at randomly chosen prices $p_j^*(t)$ within the holding period.

BUY AND HOLD ($B + H$). Buys $j^*$ at the first day $t$ and sells at the last day $T$ of each period.

MARKET (MA). Knows all prices $p_j^*(t)$ of a period in advance. Each holding period MA will buy the MT asset at the minimum possible price $p_{\text{min}} \geq m(i)$ and sell at the maximum possible price $p_{\text{max}} \leq M(i)$.

The performance of the investment policies is evaluated empirically.

3 Experimental Results

Simulations of the trading policies discussed in Section 2 are run for all six period types with number $n(p)$ from $\{52, 26, 13, 4, 2, 1\}$ and length $h$. Clearly the benchmark policy MA cannot be beaten. Simulations are run on Xetra DAX data for the interval 2007/01/01 to 2007/12/31 in order to find out

(1) if $MT$ shows a superior behaviour to buy-and-hold policies (2) the influence of $m$ and $M$ on the performance of $MT$

Two types of $B + H$ are simulated. ($MT_{B + H}$) holds the MT asset within each period and (Index$_{B + H}$) the index over the whole time horizon. $MT_{B + H}$ is synchronized with $MT$, i.e. buys the MT asset on the first day and sells it on the last day of each period. Index$_{B + H}$ is a common policy and often used as a benchmark. In addition the random policy Rand buys and sells the MT asset on randomly chosen days within a holding period.

We first concentrate on question (1) if $MT$ shows a superior behaviour to $MT_{B + H}$ and Index$_{B + H}$. Simulation runs with two different reservation prices are carried out, called $A$ and $R$. For calculating both reservation prices estimates from the past are used, i.e. in case of a period length of $h$ days $m$ and $M$ are taken from these $h$ days which are preceding the actual day $t^*$ of the reservation price calculation, i.e. $m = \min \{p(t)|t = t^* - 1, t^* - 2, \ldots, t^* - h\}$ and $M = \max \{p(t)|t = t^* - 1, t^* - 2, \ldots, t^* - h\}$. Table 1 displays trading results under transaction costs. For MA, $MT$ and Rand) transaction costs are the same; all follow the holding period of $MT$. The $MT$ policy for both reservation prices, $R$ and $A$, dominates $MT_{B + H}$ and Index$_{B + H}$ in two cases (1 and 4 weeks). $MT_{B + H}$ dominates $MT$ and Index$_{B + H}$ in two cases (6 and 12 months). Index$_{B + H}$ dominates $MT$ and $MT_{B + H}$
Table 1. Annualized return rates for different period lengths

<table>
<thead>
<tr>
<th>Historic R</th>
<th>Annualized Returns Including Transaction Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy</td>
<td>1 Week</td>
</tr>
<tr>
<td>n(7) = 52</td>
<td>n(14) = 26</td>
</tr>
<tr>
<td><strong>MA</strong></td>
<td>418.18%</td>
</tr>
<tr>
<td><strong>MT</strong></td>
<td>41.08%</td>
</tr>
<tr>
<td><strong>MT_{B+H}</strong></td>
<td>9.70%</td>
</tr>
<tr>
<td>Index_{B+H}</td>
<td>20.78%</td>
</tr>
<tr>
<td><strong>Rand</strong></td>
<td>-23.59%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Historic A</th>
<th>Annualized Returns Including Transaction Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy</td>
<td>1 Week</td>
</tr>
<tr>
<td>n(7) = 52</td>
<td>n(14) = 26</td>
</tr>
<tr>
<td><strong>MA</strong></td>
<td>437.14%</td>
</tr>
<tr>
<td><strong>MT</strong></td>
<td>31.52%</td>
</tr>
<tr>
<td><strong>MT_{B+H}</strong></td>
<td>7.45%</td>
</tr>
<tr>
<td>Index_{B+H}</td>
<td>20.78%</td>
</tr>
<tr>
<td><strong>Rand</strong></td>
<td>-1.49%</td>
</tr>
</tbody>
</table>

MT generates the best overall annual return rate when applied to 4 weeks. In case R, \( MT_{B+H} \) generates the worst overall annual return rate when applied to 2 weeks, in case A when applied to 1 week. \( MT_{B+H} \) improves its performance in comparison to Index_{B+H} and MT proportional to period length \( h \). The longer the period the better the relative performance of \( MT_{B+H} \). MT outperforms Index_{B+H} in two-thirds and \( MT_{B+H} \) in one-thirds of the cases. If period length \( h \leq 4 \) MT outperforms \( MT_{B+H} \) in all cases and if \( h > 4 \) \( MT_{B+H} \) outperforms MT in all cases. Index_{B+H} outperforms \( MT_{B+H} \) in half the cases. If we consider the average performance we have 27.86% for MT, 20.78% for Index_{B+H}, and 20.63% for \( MT_{B+H} \) in case R and 30.63% for MT, 20.78% for Index_{B+H}, and 22.09% for \( MT_{B+H} \) in case A. MT is best on average. On average MT shows a superior behaviour to B+H policies under the assumption that \( m \) and \( M \) are based on historical data.

In general we assume that the better the estimates of \( m \) and \( M \) the better the performance of MT. Results in Table 1 show that the longer the periods the worse the relative performance of MT. This might be due to the fact that for longer periods historical \( m \) and \( M \) are worse estimates in comparison to those for shorter periods. To analyze the influence of estimates of \( m \) and \( M \) simulations are run with the observed \( m \) and \( M \) of the actual periods, i.e. we have optimal estimates. Results shown in Table 2 have to be considered in comparison to the results for historic estimates in Table 1. Now we can answer question (2) discussing the influence of \( m \) and \( M \) on the performance of MT. In all cases the returns of policy MT improve significantly when estimates
Table 2. Annualized returns for optimal historic estimates

<table>
<thead>
<tr>
<th>Clairvoyant</th>
<th>Annualized Returns Including Transaction Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Policy 1 Week 2 Weeks 4 Weeks 3 Months 6 Months 12 Months</td>
</tr>
<tr>
<td></td>
<td>n(7) = 52 n(14) = 26 n(28) = 13 n(91) = 4 n(182) = 2 n(364) = 1</td>
</tr>
<tr>
<td>MA</td>
<td>418.18% 315.81% 280.94% 183.43% 86.07% 70.94%</td>
</tr>
<tr>
<td>MT</td>
<td>102.60% 87.90% 76.10% 81.38% 55.11% 54.75%</td>
</tr>
<tr>
<td>MT_{B+H}</td>
<td>9.70% -4.40% 22.31% 19.79% 45.30% 35.29%</td>
</tr>
<tr>
<td>Index_{B+H}</td>
<td>20.78% 20.78% 20.78% 20.78% 20.78% 20.78%</td>
</tr>
<tr>
<td>Rand</td>
<td>-23.59% -101.3% -10.67% 47.37% 46.08% 15.42%</td>
</tr>
</tbody>
</table>

4 Conclusions

To answer the questions from section 3 24 simulation runs were performed. In the clairvoyant test set MT outperforms B + H in all cases even under transaction costs. Tests on historical estimates of m and M show that MT outperforms B + H in one-thirds of the cases and also on average. We conclude that if the period length is small enough MT outperforms B + H. It is obvious that the better the estimates of m and M the better the performance of MT. Results show that the shorter the periods, the better the estimates by historical data. As a result, the performance of MT gets worse the longer the periods become. It turned out that the shorter the periods the less achieves MT in comparison to MA. A MT trading policy which is applied to short periods leads to small intervals for estimating historical m and M. In these cases there is a tendency to buy too late (early) in increasing (decreasing) markets and to sell too late (early) in decreasing (increasing) markets due to unknown overall trend directions, e.g. weekly volatility leads to wrong selling decisions during an upward trend.

The paper leaves some open questions for future research. One is that of better forecasts of future upper and lower bounds of asset prices to improve the performance of MT. The suitable period length for estimating m and M is an important factor to provide a good trading
signal. Simulations with other period lengths for estimating $m$ and $M$ could be of interest. Moreover, the data set of one year is very small. Future research should consider intervals of 5, 10, and 15 years.

References

6.2 Results of Schmidt, Mohr and Kersh (2010)

Digital Object Identifier (DOI): 10.1016/j.endm.2010.05.066
Electronic Notes in Discrete Mathematics, Vol. 36, pp. 519-526, 2010
© 2010 Elsevier B.V.

Preface

The following research paper investigates the performance of different online conversion algorithms. The bi-directional non-preemptive reservation price (RP) algorithm of Mohr and Schmidt (2008a,b) (Algorithm 5, p. 83) is compared to the preemptive threat-based algorithm of El-Yaniv et al. (1992, 2001) (Algorithm 8, p. 92). Algorithm 5 is presented in detail in Section 4.1, and Algorithm 8 in Section 5.1.

Algorithm 5, denoted by SQRT, achieves a worst-case competitive ratio as given in Theorem 2. Schmidt et al. (2010) consider Variant 2 of Algorithm 8, denoted by Threat($m, M, k$), i.e. the a-priori knowledge of $m, M$ and the number of trading days $k \leq T$ is assumed. The worst-case competitive ratio of Algorithm 8 is strictly increasing with $k$, and calculated as given in equation (5.24).

For the empirical-case analysis transaction costs are not considered, and the backtesting of the algorithms is done on the German Dax-30 index for the investment horizon 01-01-1998 to 12-31-2007; stylized facts are given in Example 2, p. 62. Only the index itself can be traded by the investigated algorithms $ON \in \{\text{SQRT}, \text{Threat} (m, M, k), \text{CR}, \text{BH}\}$ and $\text{OPT}$. The investment horizon is divided into several time intervals of different length $T$. Within each $T$ uni-directional search, solving either the min-search problem for buying or the max-search problem for selling, might be carried out. As suggested in the work of Borodin et al. (2004), two consecutive time intervals of equal length $T$ built trading intervals of length $2 \cdot T$, with $T \in \{260, 130, 65, 20, 10\}$. In order to trade multiple times for example $2 \cdot T = 260$ days equal $T = 130$ days for buying, and $T = 130$ days for selling, etc. The following questions are to be answered:

1. How does the empirical performance of the algorithms compare?

2. How do the empirical-case competitive ratios $c^{ec}$ found in the experiments compare?

3. How do the worst-case competitive ratios $c^{wc}$ which could have been possible from the experimental data compare?

---

35The copyright permission can be found in the Appendix, cf. Section A.2 and the original publication is available at www.elsevier.com/locate/endm.
4. What are the performance ratios \([\text{Threat}(m, M, k)/\text{SQRT}]\) in the empirical-case and in the worst-case?

5. Can the answers to Questions 1 and 2 be confirmed by a statistical \(t\)-test?

Algorithm SQRT uses precise estimates to calculate a reservation price \(q^* = \sqrt{M \cdot m}\): In case of a time interval of length \(T\) days the actually observed values of \(m\) and \(M\) within each \(T\) are used. Analogously, Threat\((m, M, k)\) uses precise estimates of \(M\), \(m\) and \(k\) to calculate the amount to be converted \(s_t\) using equation (5.20). The constant rebalancing algorithm \((CR)\) converts the same amount \(s_t = 1/T\) of the index on each day \(t\). The empirical-case performance is evaluated by a \(t\)-test, as given in Algorithm 2, p. 67.

Results show that Threat\((m, M, k)\) clearly outperforms BH and CR. To reduce the number of conversions SQRT is a good alternative to Threat\((m, M, k)\) as it also outperforms BH. The results found in the experiments could be confirmed by the \(t\)-test. Summing up, Schmidt et al. (2010) analyze uni-directional conversion while converting a single asset from an empirical-case and a worst-case point of view.
Experimental Analysis of an Online Trading Algorithm

Günter Schmidt a,b,1, Esther Mohr a,2, Mike Kersch a,3

a Saarland University
P.O. Box 151150
D-66041 Saarbrücken, Germany
Pho +49-681-302-4559
Fax +49-681-302-4565

b University of Liechtenstein
Fürst-Franz-Josef-Strasse
9490 Vaduz, Liechtenstein

Abstract
Trading decisions in financial markets can be supported by the use of online algorithms. We evaluate the empirical performance of a threat-based online algorithm and compare it to a reservation price algorithm, an average price algorithm and to buy-and-hold. The algorithms are analyzed from a worst case and an empirical case point of view. The effectiveness of the algorithms is analyzed with historical DAX-30 prices for the years 1998 to 2007. The performance of the threat-based algorithm found in the simulation runs dominates all other investigated algorithms. We also compare its performance to results from worst case analysis and conduct a t-test.

Keywords: Investment analysis, Decision support systems, Decision analysis, Heuristics, OR in banking, Simulation, Risk management, Uncertainty modeling
1 Introduction

The performance analysis of trading algorithms can basically be carried out by three different approaches. One is Bayesian analysis where a given probability distribution of asset prices is a basic assumption. Another one is competitive analysis where uncertainty about asset prices is assumed. Algorithms are analyzed under worst case outcomes. The third one is a heuristic approach where analysis is done on historic data by simulation runs. We apply the second and the third approach considering single and multiple trade problems.

2 Problem Formulation

In a single trade problem we search for the minimum price $m$ and the maximum price $M$ once. In a multiple trade problem we trade more than once. If we buy and sell assets $k$ times we call the problem $k$-trade problem with $k \geq 1$. As we do not know future asset prices decisions to be taken are subject to uncertainty. Trading is represented by search. To solve the financial search problem a trader observes prices $q(t)$ with $m \leq q(t) \leq M$ at points of time $t = 1, 2, \ldots, T$. For each $q(t)$ he must decide which fraction of his current asset $s(t)$ he wants to sell at time $t$. At the last price $q(T)$ the trader must sell all the remaining fractions of the asset he holds. It is assumed that the time interval $[1, T]$ and the possible minimum and maximum prices $m$ and $M$ are known.

The problem to determine $s(t)$ is solved by online algorithms. An algorithm $ON$ computes online if for each $j = 1, \ldots, T - 1$, it computes an output for $j$ before the input for $j + 1$ is given. An algorithm $OPT$ computes offline if it computes a feasible output given the entire input sequence $j = 1, \ldots, T - 1$. An online algorithm $ON$ is $c$-competitive if for any input $I$

\[(1) \quad ON(I) \geq \frac{1}{c} \cdot OPT(I).\]

If the competitive ratio is related to a performance guarantee it must be a worst case measure. Thus any $c$-competitive online algorithm guarantees a value of at least the fraction $1/c$ of the optimal offline value $OPT(I)$ no matter how unfortunate or uncertain the future will be. As we have a maximization problem $c \geq 1$ the smaller $c$ the more effective is $ON$. We analyze the competitive ratio of two online algorithms based on a reservation price policy ($s(t) \in \{0, 1\}$) and a threat-based policy ($0 \leq s(t) \leq 1$).

1 Email: gs@itm.uni-sb.de
2 Email: em@itm.uni-sb.de
3 Email: mk@itm.uni-sb.de
Reservation Price Policy. For the search problem the selling rule introduced by [2] “sell at the first price greater or equal to reservation price \(q^* = \sqrt{M \cdot m}\) has a worst case competitive ratio \(c_s = \sqrt{M \cdot m}\) where \(M\) and \(m\) are upper and lower bounds of prices \(q(t) \in [m, M]\). This result can be transferred to a single trade problem if we modify the rule to “buy at the first price smaller or equal and sell at the first price greater or equal to \(q^* = \sqrt{M \cdot m}\)”. In the single trade problem we have to carry out search twice. In the worst case we get \(c_s\) for buying and the same \(c_s\) for selling resulting in an overall competitive ratio for single trading \(c_t = c_s \cdot c_s = M \cdot m\). For the \(k\)-trade problem we get a worst case competitive ratio of

\[
(2) \quad c_t(k) = \prod_{i=1}^{k} \left( \frac{M(i)}{m(i)} \right)
\]

If \(m\) and \(M\) are constant for all trades \(c_t(k) = (\frac{M}{m})^k\). The ratio \(c_t(k)\) can be interpreted as the geometric return we can achieve by buying and selling sequentially as stated in [5].

Threat-based Policy. To solve the search problem the following procedure is suggested by [3]: (i) Choose a competitive ratio \(c\) and select a trading policy which can guarantee \(c\). (ii) Consider trading asset \(d\) for asset \(y\) only when the current exchange rate \(q(t)\) is the highest seen so far. (iii) Whenever you trade asset \(d\) for asset \(y\) convert just enough to ensure that the given \(c\) would be obtained if an adversary dropped the next rate \(q(t + 1)\) to the minimum possible rate \(m\) and kept it there until the end of the time horizon \(T\), i.e. that this threat exists. Let \(k \leq T\) be the remaining exchange rates in the time series. Let \(q'(1)\) be the first exchange rate of this time series. Let \(c^k(q'(1))\) be a competitive ratio which is achievable on a sequence of \(k\) exchange rates \(q'(1), \ldots, q'(k)\). The achievable competitive ratio \(c^k(q'(1))\) for \(k\) remaining trading days is

\[
(3) \quad c^k(q'(1)) = 1 + \frac{q'(1) - m}{q'(1)} \cdot (k - 1) \cdot \left(1 - \left[\frac{q'(1) - m}{M - m}\right]^\frac{1}{k-1}\right)
\]

\(c = \sup \ c^k(q(1), q(2), \ldots, q(k)|k \leq T)\) is the optimal competitive ratio for the search problem [3]. For each trade we conduct the threat-based algorithm twice. The competitive ratio for trading of the threat-based algorithm can be calculated in the same way as it is done for the reservation price algorithm.
3 Experiments

We use daily closing prices of the DAX-30 index for the time interval 01-01-1998 to 12-31-2007 and divide the time horizon into several trading periods $i$ of different length $K$. Each $i$ consists of two sub-periods $T = \left\lceil \frac{K}{2} \right\rceil$ for buying (buying period $b$) and $T = \left\lfloor \frac{K}{2} \right\rfloor$ for selling (selling period $s$). We differ between trading periods with length 260, 130, 65, 20, 10 days, i.e. for $K = 260$ days $T = 130$ days for buying (selling) etc. We investigate the following trading algorithms:

**Optimal Trading.** Optimal Trading ($OPT$) is an offline algorithm which achieves the best possible return in each $i$. We assume that $OPT$ knows all prices of $i$. $OPT$ buys at the minimum realized price $p_{\text{min}} \geq m(b)$ and sells at the maximum realized price $p_{\text{max}} \leq M(s)$ in each sub-period.

**Threat-based Trading.** Every time an exchange is carried out the threat-based algorithm ($Threat$) calculates the achievable competitive ratio and buys (sells) the corresponding quantities such that the achievable $c$ is realized in each sub-period.

**Reservation Price Trading.** For every sub-period the reservation price algorithm ($Square$) calculates reservation prices $RP(t)$ for each day $t$. $Square$ buys (sells) the index at the first price $q(t) \leq (\geq)RP(t)$. If there was no such price buying (selling) has to be done on the last day $T$ of a period.

**Average Price Trading.** The average price algorithm ($Constant$) buys (sells) with the constant fraction $\frac{1}{T}$ in each sub-period.

**Buy and Hold.** Buy and Hold ($BH$) buys on the first day of the buying period and sells on the last day of the selling period.

The following assumptions apply for all algorithms: (1) there is an initial cash value greater zero; (2) transaction costs are not considered; (3) minimum price $m$, maximum price $M$, and the length $T$ of each sub-period are known; (4) interest rate on cash is zero; (5) within each $b$ all cash must be exchanged in the index and within each $s$ all index must be exchanged back into cash; (6) the performance measure is the average trading period return ($AR$). $AR$ tells us which performance we could expect within $i$. Let $d_i$ and $D_i$ be the amount of cash at the beginning and at the end of period $i$. Let $r_i = \frac{D_i}{d_i}$ be the return in $i$. Let $n$ be the number of trading periods considered. Then,

\begin{equation}
AR(n) = \left( \prod_{i=1}^{n} r_i \right)^{\frac{1}{n}}.
\end{equation}
We also calculate the worst case competitive ratio and the empirical case competitive ratio. The competitive ratio is calculated by solving equation (1) to \( c \) where \( ON \in \{ \text{Threat}, \text{Square}, \text{Constant}, \text{BH} \} \). Let \( c_w \) be the worst case competitive ratio and let \( c_e \) be the empirical case competitive ratio. For the worst case competitive ratio \( ON(I) \) is the worst case return which could have been achieved taking the data of the problem instance into account; for the empirical case competitive ratio \( ON(I) \) is the empirical case return which actually was achieved by \( ON \) and is calculated according to equation (4). We only consider \( c_w \) for algorithms \text{Threat} and \text{Square}. For \text{Threat} the empirical ratio can be achieved also in the worst case. Thus, \( c_w \) of \text{Threat} is the same as its \( c_e \). For \text{Square} we must calculate \( c_w \). Let \( m(b) \) and \( M(b) \) be the bounds for \( b \) and let \( m(s) \) and \( M(s) \) be the bounds for \( s \). Then, for trading the worst case competitive ratio is \( c_w = \sqrt{(M(b) \cdot M(s))/(m(b) \cdot m(s))} \). To find out how \text{Threat} and \text{Square} behave relative to each other in the empirical and in the worst case we calculate empirical case ratio by \( AR_{\text{Threat}}(n)/AR_{\text{Square}}(n) \). For the worst case we want to know the worst case return ratio of \text{Threat} and \text{Square}, i.e. \( c(\text{Square})/c(\text{Threat}) = \text{Threat}(I)/\text{Square}(I) \) where \( \text{Threat}(I) \) and \( \text{Square}(I) \) relate to worst case performances.

4 Experimental Results

We carried out simulation runs in order to find out how the following measures compare: (1) the empirical performance of the algorithms; (2) the \( c_e \) found in the experiments; (3) the \( c_w \) which could have been possible from the experimental data; (4) the performance ratios \( \text{Threat}/\text{Square} \) in the empirical case and in the worst case. Clearly, all online algorithms cannot beat the benchmark algorithm \textit{OPT}.

Question 1: How does the empirical performance of the algorithms compare? Answering this question we calculated the experimental performance of the online algorithms \text{Threat}, \text{Square}, \text{BH}, and \text{Constant} and compared it to \textit{OPT} (cf. equation (4)). Results are presented in Table 1. \text{Threat} dominates all other online algorithms. \text{Square} dominates \text{BH} and \text{Constant}. \text{Constant} is dominated by all other algorithms except for 65 days. We can conclude that in our experiments it is better to have more periods \( i \) than longer ones.

Question 2: How do the \( c_e \) found in the experiments compare? Clearly, the answers to Question 1 regarding the performance comparison of the algorithms are also true for Question 2 because the numerator in \( c \geq \textit{OPT}(I)/ON(I) \) is constant for all algorithms in each \( i \). The shorter the trading period length the better is the \( c_e \) of the algorithms, i.e. the algorithms loose performance
Table 1
Average Period Return in the interval 1998-2007

<table>
<thead>
<tr>
<th>Period Length</th>
<th>10 days</th>
<th>20 days</th>
<th>65 days</th>
<th>130 days</th>
<th>260 days</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT</td>
<td>1.0308</td>
<td>1.0562</td>
<td>1.1320</td>
<td>1.2110</td>
<td>1.2923</td>
</tr>
<tr>
<td>Threat</td>
<td>1.0236</td>
<td>1.0376</td>
<td>1.0807</td>
<td>1.0981</td>
<td>1.1636</td>
</tr>
<tr>
<td>Square</td>
<td>1.0218</td>
<td>1.0302</td>
<td>1.0602</td>
<td>1.0528</td>
<td>1.1220</td>
</tr>
<tr>
<td>BH</td>
<td>1.0024</td>
<td>1.0050</td>
<td>1.0137</td>
<td>1.0242</td>
<td>1.0568</td>
</tr>
<tr>
<td>Constant</td>
<td>1.0005</td>
<td>1.0028</td>
<td>1.0154</td>
<td>1.0099</td>
<td>0.9930</td>
</tr>
</tbody>
</table>

compared to OPT the longer the periods are.

**Question 3:** How do the $c_w$ which could have been possible from the experimental data compare? Answering this question we calculated the $c_w$ for Threat and Square which are possible from the data set. The results are shown in Table 2. Using the worst case criteria Threat clearly outperforms Square, i.e. if we like to minimize worst case returns we choose Threat. Moreover the performance of Square gets worse compared to Threat the longer the periods are.

Table 2
Worst case competitive ratio for the interval 1998-2007

<table>
<thead>
<tr>
<th>Period Length</th>
<th>10 days</th>
<th>20 days</th>
<th>65 days</th>
<th>130 days</th>
<th>260 days</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT/Threat</td>
<td>1.0070</td>
<td>1.0179</td>
<td>1.0475</td>
<td>1.1028</td>
<td>1.1106</td>
</tr>
<tr>
<td>OPT/Square</td>
<td>1.0302</td>
<td>1.0529</td>
<td>1.1109</td>
<td>1.1962</td>
<td>1.2913</td>
</tr>
</tbody>
</table>

**Question 4:** What are the performance ratios Threat/Square in the empirical case and in the worst case? Comparing Threat and Square by their $c_w$ we know that Threat outperforms Square (cf. Table 2). Answering **Question 4** we want to know how the ratios of the worst case and of the empirical case differ, i.e. where the out-performance is greater. The answer is given in Table 3. Using the $AR$ as performance measure the ratio is between 2.3% and 16.3% in the worst case and only between 0.18% and 4.31% in the experiments. So we conclude that trading with Square is a good alternative to Threat in practical applications especially if we want to reduce the number
of transactions.

Table 3
Empirical case versus worst case ratio for the interval 1998-2007

<table>
<thead>
<tr>
<th>Period Length</th>
<th>10 days</th>
<th>20 days</th>
<th>65 days</th>
<th>130 days</th>
<th>260 days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empirical Case</td>
<td>1.0018</td>
<td>1.0072</td>
<td>1.0193</td>
<td>1.0431</td>
<td>1.0370</td>
</tr>
<tr>
<td>Worst Case</td>
<td>1.0230</td>
<td>1.0343</td>
<td>1.0605</td>
<td>1.0847</td>
<td>1.1627</td>
</tr>
</tbody>
</table>

**Question 5:** Can the answers to Questions 1 and 2 be confirmed by a statistical \( t \)-test? The null hypothesis \( H_0 \) is that the AR of one algorithm \( A_1 \leq A_2 \). Before running a \( t \)-test we have to check if the \( r_i \) of the compared two algorithms (\( t \)-test samples) are normally distributed (Jarque-Bera test) and have equal variances or not. If data is normally distributed, the Bartlett test is used to test the variances; if not the Levene test [1]. The \( r_i \) are used to run the \( t \)-test. Depending on the results for the variances different kinds of \( t \)-tests are used. We use a significance level of 5%. We run five \( t \)-tests for each pair of algorithms, one for each period length. For six pairs of algorithms 30 \( t \)-tests were conducted. The answers to the above questions are summarized in Table 4: the ‘no’ entries in column ‘\( t \)-test’ mean that the null hypothesis cannot be rejected; the ‘(yes)’ entry means that the null hypothesis could not be rejected for two period lengths. The results found in the experiments could be confirmed clearly in three cases and weakly in one case. This is also true for the corresponding competitive ratio. Where the results from the experiments cannot be confirmed by a \( t \)-test the returns generated by the two algorithms are too close to produce significance.

5 Conclusions

\textit{Threat} clearly outperforms \textit{BH} and \textit{Constant}. If transaction costs have to be considered \textit{Threat} still outperforms \textit{Constant} because it never generates more transactions. If we want to reduce transaction costs \textit{Square} is a good alternative to \textit{Threat}, i.e. it also outperforms \textit{BH}. The worst AR is achieved by \textit{Constant}. \textit{BH} looses performance relative to \textit{Threat} and \textit{Square} the shorter the periods are. For the worst case ratio AR values are increasing the longer the periods are. The worst case performance is the greater the difference in \( m \) and \( M \), which gets greater with longer periods. It would be interesting to analyze the performance of \textit{Threat} compared to...

\[525]
Table 4
Summary of simulation and t-test results

<table>
<thead>
<tr>
<th>10 Year Interval 1998-2007</th>
<th>Average Period Return</th>
<th>Simulation</th>
<th>t-test</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Threat dominates Square</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>(2) Threat dominates BH</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>(3) Threat dominates Constant</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>(4) Square dominates BH</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>(5) Square dominates Constant</td>
<td>yes</td>
<td>(yes)</td>
<td></td>
</tr>
<tr>
<td>(6) BH dominates Constant</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

Square and BH in further experiments taking transaction costs into account. Another open question is to conduct experiments with forecasts for m and M. The suitable period length for estimating m and M is an important factor to provide good online algorithms. It would be of further interest to assume that we do not have information about m and M. One approach is to observe a certain number k of the T prices within a time horizon with k < v ≤ T and then trade to the next best price q(v) > max (< min) \{ q(j) | j = 1, \ldots, k \} (cf. the secretary’s problem [4]).

References


6.3 Results of Ahmad, Mohr and Schmidt (2010)

Manuscript submitted for publication, November 20, 2010

Preface

Inspired by the survey of Graham et al. (1979) the following paper provides a classification scheme for online conversion problems.

A considerable amount of literature is devoted to online conversion algorithms, an overview is given in Section 2.4. In addressing the conversion problem, various aspects are covered and different settings are assumed. In addition, the terminology used is not coherent and standardized. The great variety of online conversion algorithms, and the non-adherence to standards might lead to misconception on part of the reader. As each online conversion algorithm assumes different problem settings, assumptions and nomenclature it is difficult to evaluate the suggested algorithms on existing methods, or to compare them on a mutual basis. We provide a novel scheme to classify online conversion algorithms based on the problem setting they are using. Similarly, we define a standard nomenclature for the terms used in the literature in relation to online algorithms for conversion problems.

Our aim is to remove the discrepancies currently existing in the literature, and to introduce a standard classification scheme. Further, we provide a comprehensive review of the literature addressing online conversion problems. We restrict the literature review to competitive search algorithms in the context of conversion in financial markets, i.e. the search for best prices in order to buy and sell assets (min-search and max-search). Different classes of online conversion algorithms are discussed, and their competitive ratios are derived. We conclude indicating some problems for future research and give a selective bibliography.
A Classification Scheme for Online Conversion Problems

Iftikhar Ahmad\textsuperscript{a,∗}, Esther Mohr\textsuperscript{a}, Günter Schmidt\textsuperscript{a,b}

\textsuperscript{a}Saarland University, P.O. Box 151150, D-66041 Saarbrücken, Germany
Phone +49-681-302-4559, Fax +49-681-302-4565
\textsuperscript{b}University of Liechtenstein, Fürst-Franz-Josef-Strasse, 9490 Vaduz, Liechtenstein

Abstract
A considerable amount of literature is devoted to online conversion problems which signifies its growing importance. We provide a standard nomenclature and a unique classification scheme for online conversion problems (maximum and minimum search). Based on the suggested scheme, we classify the existing work and provide a short review of the literature. Different classes of online conversion algorithms are discussed, and their competitive ratios are shown as well. We also provide an insight into future work, and potential new areas of research.

Keywords: Classification Scheme, Online Conversion Problem, Online Algorithms, Competitive Analysis, Trading Algorithms

1. Introduction
An online conversion problem deals with the scenario of converting an asset \( D \) into another asset \( Y \) with the objective to get the maximum amount of \( Y \) after time \( T \). The process can be repeated in both directions, i.e. converting asset \( D \) into asset \( Y \), and \( Y \) back to asset \( D \). In a typical problem setting, on each day \( t \), the player is offered a price \( q_t \) to convert \( D \) to \( Y \), the player may accept the price \( q_t \) or may decide to wait for a better price. The game ends when the player converts whole of the asset \( D \) to \( Y \).

Based on the context of decision making, algorithms can broadly be classified in two categories, a) those which make a decision based on the complete

\footnotesize{\textsuperscript{∗}Principal corresponding author

Email addresses: ia@itm.uni-sb.de (Iftikhar Ahmad), em@itm.uni-sb.de (Esther Mohr), gs@itm.uni-sb.de (Günter Schmidt)}
knowledge about future input data, resulting in an optimum solution, and are referred to as optimum offline algorithms and, b) those which make a decision with no or partial knowledge about future input data, very often not resulting in an optimum solution, and are referred to as online algorithms. It is nevertheless desired to evaluate its effectiveness against the performance of other algorithms for the same problem. The technique used to evaluate online algorithms is called competitive analysis. It compares the performance of an online algorithm to that of an optimum offline algorithm. Let ‘ON’ be an online algorithm for some maximization problem ‘P’ and ‘I’ be set of all inputs. Let ON(I) be the return of algorithm ‘ON’ on input instance \( I \in \mathcal{I} \). Let ‘OPT’ be the optimum offline algorithm for the same problem ‘P’, and OPT(I) its return for the input on the same instance \( I \in \mathcal{I} \). An online algorithm ‘ON’ is called c-competitive if

\[
ON(I) \geq \frac{1}{c} \cdot OPT(I).
\]

**Problem Setting**

Consider a player who wants to convert an asset \( D \) into another asset \( Y \). Assume that the player starts with \( d_0 = 1 \) and \( y_0 = 0 \). At each time \( t = 1, 2, ..., T \) the player is offered a price \( q_t \), and must immediately decide whether to accept the offered price \( q_t \) or not. If the player decides to accept the price, he can convert a portion or the whole amount of asset \( D \) at the offered price \( q_t \). The game ends when the player has converted \( D \) completely into \( Y \). If there is still some amount of asset \( D \) remaining on the last day \( T \), it must be converted at the last offered price \( q_T \) which might be the worst(lowest) offered price.

Based on the design pattern of conversion algorithms, we can broadly classify them into two classes, a) online conversion algorithms – developed to give a performance guarantee under worst-case conditions, and referred to as guaranteeing conversion algorithms. The worst-case performance guarantee is usually evaluated using competitive analysis [15], and b) heuristic conversion algorithms – which are developed to achieve a preferably high average-case performance. Very often heuristic conversion algorithms are based on data from technical analysis [37]. The assumptions of heuristic conversion algorithms are found similar to guaranteeing conversion algorithms. Both classes work without any knowledge of future input. Guaranteeing conversion algorithms as well as heuristic conversion algorithms are referred to as
online conversion algorithms. Both classes can be evaluated using competitive analysis.

Motivation

A great deal of literature is devoted to the study of online and heuristic algorithms for conversion problems. In addressing the problem, various aspects are covered, and different settings are assumed. For instance, some algorithms are designed based on assumptions that expected lower and upper bounds of offered prices, \( m \) and \( M \), are known to the online algorithm \([11, 13, 20, 38]\). Whereas others consider assumptions in which the knowledge of the fluctuation ratio \( \phi = M/m \), and the length of the time interval \( T \) is assumed \([13, 18]\). Other variants also exist, and each depends on different assumptions \([22]\). In addition, the terminology used is not coherent and standardized. The great variety of online conversion algorithms, and the non-adherence to standards might lead to misconception on part of the reader. As each online conversion algorithm assumes different problem settings, assumptions, and nomenclature it is difficult to evaluate the suggested algorithms on existing methods, or to compare them on a mutual basis. We provide a novel scheme to classify online conversion algorithms based on the problem setting they are using. Similarly, we define a standard nomenclature for the terms used in the literature in relation to online algorithms for conversion problems. Our aim is to remove the discrepancies currently existing in the literature, and to introduce a standard classification scheme. Further, we provide a comprehensive review of the literature addressing online conversion problems. We restrict the literature review to competitive search algorithms in the context of conversion in financial markets, i.e. the search for best prices in order to buy and sell assets. Further applications like algorithmic trading, and online auctions are not considered. (cf. \([4, 8, 23]\)). We conclude presenting open questions and potential future research directions.

2. Classification Scheme

Our proposed classification scheme is based on three pillars, a) the nomenclature – a standardized set of definitions, b) the classification factors – parameters that affect the class of problems, for example the knowledge about the future prices, and c) the tree – the resultant structure that will classify existing (and future) work.
2.1. Nomenclature

We provide a standard nomenclature to define the terms used in relation to online conversion problems. The objective of the nomenclature is to adhere to a standard set of definitions, and to avoid ambiguity.

i. **Transaction**: A transaction is either selling or buying of an asset.

ii. **Trade**: A trade consists of two transactions, one is buying and one is selling. The number of trades is $p$, with $i = 1, \ldots, p$.

iii. **Investment Horizon**: The total time duration in which all transactions must be carried out. The investment horizon can be divided into one or more time intervals for conversion.

iv. **Uni-directional search (uni)**: Searching for maximum (max-search) or minimum (min-search) price(s) to carry out either a selling or a buying transaction within one time interval.

v. **Bi-directional search (bi)**: Searching for maximum (max-search) and minimum (min-search) price(s) to carry out both a buying and a selling transaction within one time interval, i.e. bi-directional search is synonym for trading.

vi. **Non-Preemptive conversion (non-pmtn)**: Search for one single price within the time interval to convert the asset.

vii. **Preemptive conversion (pmtn)**: Search for more than one price within the time interval to convert the asset. Typically the number of prices considered for conversion is determined by the algorithm. Except in one special case where the player desires to convert at a specific number $u$ of prices. This is referred to as $u$-preemption ($u - pmtn$); the player must specify $u$.

viii. **Offered Price ($q_t$)**: A price from a sequence of prices presented to the player to carry out a transaction. Offered prices are denoted by $Q = q_1, q_2, \ldots, q_T$, where $q_t$ is the price offered at time $t$ within the time interval.

ix. **Predicted Upper Bound ($M$)**: Represents the upper bound on possible prices during the time interval.
110  x. Predicted Lower Bound ($m$): Represents the lower bound on possible
111  prices during the time interval.

112  xi. Fluctuation Ratio ($\phi$): The predicted maximum fluctuation of prices
113  that can possibly be observed during the time interval, calculated by
114  $M/m$.

115  xii. Duration ($T$): The length of the time interval, where $t = 1, ..., T$.

116  xiii. Threat Duration ($k$): The number of trading days after which the offered
117  price might drop to some minimum level, for instance $m$, and stays there
118  until the last day $T$, where $k \leq T$.

119  xiv. Price Function ($g(q_t)$): Models a price $q_t$ based on some predefined
120  function; for instance the current price $q_t$ is a function of the previous
121  price $q_{t-1}$, i.e. $q_t = g(q_{t-1})$.

122  xv. Amount Converted ($s_t$): Specifies which fraction of the amount available
123  (e.g. wealth) is to be converted at price $q_t$ on day $t$, with $0 \leq s_t \leq 1$.

124  xvi. Return Function ($f(q_t)$): The return $r_t$ for accepting a price $q_t$ is not
125  exactly the price itself but a function of the price. Such as accepted
126  price minus the accumulated sampling costs for observing a time series
127  of prices during the time interval $T$.

128  xvii. Risk Tolerance ($a$): An acceptable level of risk (risk tolerance) the player
129  is willing to take for some higher reward.

2.2. Classification Factors

The factors used to classify the conversion problems are discussed as
follows:

$\alpha$. Nature of search

$\alpha_1$. Uni-directional: In uni-directional search, the player converts an
asset $D$ into another asset $Y$, but conversion back from $Y$ to $D$ is
forbidden. There is no restriction on the number of transactions.

$\alpha_2$. Bi-directional: In bi-directional search, the player converts an as-
set $D$ back and forth, i.e. converts $D$ into $Y$, and $Y$ back to $D$
etc. There is no restriction on the number of transactions.
β. Amount converted per transaction

β1. Non-preemptive conversion: Search for one single price in the time interval to convert the asset. Typically, the whole amount available is converted in one single transaction, i.e. $s_t \in \{0, 1\}$.

β2. Preemptive conversion: Search for more than one price in the time interval to convert the asset. Typically, only a fraction of the whole amount available is converted in one transaction, i.e. $s_t \in [0, 1]$.

γ. Given information

Parameters assumed to be known a priori, such as

γ1. predicted upper bound $M$,

γ2. predicted lower bound $m$,

γ3. fluctuation ratio $\phi = M/m$,

γ4. duration $T$,

γ5. threat duration $k \leq T$,

γ6. price function $g(q_t)$,

γ7. return function $f(q_t)$,

γ8. risk tolerance $a \in [1, OPT/O\!N]$.

2.3. The Tree

Based on the classification factors, we can divide a conversion problem into one of four main categories, as shown in Fig: 1. i) Uni-directional Non-preemptive, ii) Uni-directional Preemptive, iii) Bi-directional Non-preemptive, and iv) Bi-directional Preemptive. One observation from the tree structure is that a solution for a problem at the higher level (closer to the root) is also a solution for the problem setting at the lower level in the same path. For instance a solution for the problem setting of uni-directional preemptive conversion with only $M$ and $m$ known is also a solution for the lower level in the same path, where further knowledge is assumed; for example the duration $T$. This however does not guarantee the same performance, i.e. the solution for a higher level may not necessarily be as good as the one where more a priori knowledge is assumed. It must be noted that for the sake of clarity, we do not show all the possible nodes in the tree (Fig:1). Likewise, a scenario where
the player has no knowledge about the future, is not represented as separate node in the tree and can be represented at the same level as non-preemptive ($\beta_1$) or preemptive ($\beta_2$). We limit our review only to those nodes relevant to the problems addressed in the literature.

![Classification tree based on the classification factors](image)

**Figure 1:** Classification tree based on the classification factors

### 3. Uni-directional Search

The main focus of conversion problems remains on uni-directional search. We classify the uni-directional search problem in two main categories based on the amount converted per transaction. We relate our discussion w.l.o.g. to max-search.
3.1. Uni-directional Non-preemptive Conversion

In the uni-directional non-preemptive scenario, the player is allowed to convert an asset $D$ into an asset $Y$ in one single transaction, based on a pre-calculated reservation price ($RP$). The literature concerning the uni-directional non-preemptive scenario is either based on one single $RP$, denoted by $q^*$, or on a time varying $RP$, denoted by $q_t^*$. In both cases, each price $q_t$ offered at day $t$ is checked against the pre-calculated $RP$: If the offered price $q_t$ is greater than or equal to $RP$ the price $q_t$ is accepted, and search is closed. Otherwise the search continues until the desired price is offered or the last price $q_T$ occurs which the player must accept. At this point, asset $D$ must be converted at price $q_T$, which might be $m$.

Problems from the literature addressing the uni-directional non-preemptive scenario are discussed in the following.

3.1.1. Problem: uni\-|\-non-pmtn\|\(M,m\)

El-Yaniv [12] provided an elegant algorithm for uni-directional non-preemptive conversion with $m$ and $M$ known. The algorithm is called ‘Reservation Price Policy’ (RPP).

Algorithm 1. Accept the first price greater than or equal to $q^* = \sqrt{M \cdot m}$.

**Theorem 1.** Algorithm 1 is $\sqrt{M/m}$ competitive.

**Proof.** Let the reservation price ($RP$) be $q^*$. Two cases exist: i) the computed $RP$ is too low, or ii) the computed $RP$ is too high. A clever adversary with complete knowledge of the future, and the $RP$, can use this information to exploit the algorithm making the player perform worse, as shown in the following.

*Case 1:* If $q^*$ is too low, then the adversary provides an input sequence in such format that $M \geq q_{\text{max}} \geq q^*$, and thus the player may suffer from the so called ‘too early error’: The player could have achieved $M$ but gets $q^*$ in the worst-case. The competitive ratio achieved thus will be $c_1 = M/q^*$.

*Case 2:* If $q^*$ is too high, then the adversary provides an input sequence in such format that $m \leq q_{\text{max}} \leq q^*$, and thus the player may suffer from the ‘too late error’: The player could have achieved $q^*$, and gets $m$ in the worst-case. The competitive ratio achieved thus will be $c_2 = q^*/m$.
The player must choose a $q^*$ while balancing the two errors, i.e. to ensure that
\[
\begin{align*}
  c_1 &= c_2 \\
  M/q^* &= q^*/m \\
  q^* &= \sqrt{M \cdot m}
\end{align*}
\]
Thus, we get an overall competitive ratio of $\sqrt{M/m}$. ■

3.1.2. Problem: $\text{uni|non-pmtn}|M,T$
Damaschke et al. [10] considered a problem setting in which the upper bound $M$, and the duration $T$ is known. The model assumes that the prices offered $q_t \in [M/T, M]$, i.e. the minimum possible price $q_{\text{min}} = M/T$, and the maximum possible price $q_{\text{max}} = M$, with $t = 1 \ldots T$.

Algorithm 2. Accept the first price greater than or equal to $q^* = M/\sqrt{T}$.

Theorem 2. Algorithm 2 is $\sqrt{T}$ competitive.

Proof. Let the reservation price (RP) be $q^*$, and $q_{\text{max}} \leq M$ the highest price selected by the adversary. At any time $t \leq T$ the player accepts an offered price if $q_t \geq q^*$. If no such price occurs, the player must accept the minimum value $q_{\text{min}} = M/T$. Two cases exist: i) the computed RP is too low, or ii) the computed RP is too high. A clever adversary with complete knowledge of the future, and the RP, can use this information to exploit the algorithm making the player perform worse, as shown in the following.

Case 1: If $q^*$ is too high, the adversary will choose $q_{\text{max}} < q^*$. As no offered price $q_t$ will satisfy the condition $q_t \geq q^*$ during $T$, the player must accept $q_{\text{min}} = M/T$ on day $T$ in the worst-case. Thus, the competitive ratio in this case equals
\[
\begin{align*}
  c_1 &= \frac{\text{OPT}}{\text{ON}} \\
  &= \frac{q_{\text{max}}}{(M/T)} \\
  &< \frac{q^*}{(M/T)}.
\end{align*}
\]

Case 2: If $q^*$ is too low, the adversary will offer $q^*$ as the first price $q_1$. The player will accept $q_1$, and the game ends. Afterwards, the adversary
increases the prices up to $q_{max} = M$. Thus, the competitive ratio in this case equals

\[ c_2 = \frac{OPT}{ON} = \frac{M}{q^*}. \]  

The player must choose a $q^*$ while balancing the competitive ratios $c_1$ and $c_2$, resulting in

\[ c_1 = c_2 = \frac{q_{max}}{(M/T)} = \frac{M}{q^*} \]

\[ q^* = \frac{M}{\sqrt{T}}. \]

Thus, we get an overall competitive ratio of $\sqrt{T}$. □

3.1.3. Problem: uni\|non-pmntn\|M,m,f(q_t)

Xu et al. [38] presented a uni-directional non-preemptive RP algorithm based on the assumption that the lower and upper bounds, $m$ and $M$, as well as the return function $f(q_t)$ are known to the player. The model extends the algorithm by El-Yaniv [12] (cf. Problem: uni\|non-pmntn\|M,m) by introducing sampling costs for observing prices $q_t$. It is assumed that the achievable return $r_t$ when accepting a price $q_t$ on day $t$ is not exactly the price itself, but a function of the price (accepted price minus accumulated sampling cost). In contrast to El-Yaniv [12] the considered RP is not constant but varies with time, and thus is denoted by $q^*_t$. After the player accepts one specific price $q'$, the game ends. It is assumed that a larger price results in a larger return $r'$ for $q'$. Further, the achieved return $r'$ is higher when accepting the price $q'$ earlier, as less sampling costs occur. These basic assumptions are summarized as follows:

i. The values $m$, $M$ and $f_i(q')$ are known to the player, and the price $q_t \in [m, M]$ with $0 < m < M$.

ii. The return function $f_i(q')$ with $t = 1, 2, \ldots, T$ is continuous, and increasing in $q'$.

iii. For any accepted price $q' \in [m, M]$ the return for accepting $q'$ is the higher the earlier $q'$ is accepted: $f_1(q') \geq f_2(q') \geq \cdots \geq f_T(q') > 0$.  

10
Algorithm 3. On day \( t \), accept price \( q_t \) if \( q_t \geq q^*_t \) resulting in a return \( f_t(q_t) \).

If no price was accepted until the last day \( T \), the last price \( q_T \) must be accepted (possibly \( q_T = m \) resulting in \( f_T(m) \)).

Xu et al. [38] focus on the case where \( f_{t+1}(M) > f_t(m) \) for \( t \in [1, T-1] \), because if \( f_{t+1}(M) \leq f_t(m) \) the game ends on or before day \( t \) as the player achieves a return of \( f_j(q_j) \geq f_t(m) \) when accepting \( q_j \) at day \( j \in [1, t] \).

Calculating Reservation Price \( q^*_t \)

From assumption (i.) follows that for \( T = 1 \) the unique price \( q_1 = q' \) with the same return is accepted. Thus, the case where \( T \geq 2 \) is of main interest.

For each (unknown) duration \( L \in [1, T] \) let

\[
Z_L = \min \left\{ \max \left\{ \frac{f_{t+1}(M)}{f_t(m)}, \sqrt{\frac{f_2(M)}{f_t(m)}} \right\}, t = 1, \ldots, L-1 \right\}, \sqrt{\frac{f_2(M)}{f_L(m)}} \right\}
\]

with \( Z_L \geq 1 \) since \( f_{t+1}(M) > f_t(m) \), and \( f_2(M) > f_L(m) \). Let

\[
L' = \max \left\{ L | L = \arg \max_{2 \leq L \leq T} Z_L \right\}.
\]

This means that \( Z_{L'} \geq Z_L \) for every \( L \in [2, T] \). By definition of \( Z_{L'} \) there exists a natural number \( x \), such that

\[
Z_{L'}^x = \frac{f_{x+1}(M)}{f_x(m)} \quad \text{for} \quad x \leq L' - 1,
\]

or

\[
Z_{L'}'' = \sqrt{\frac{f_2(M)}{f_x(m)}} \quad \text{for} \quad x \leq L',
\]

with

\[
Z_{L'} = \min \{ Z_{L'}^x, Z_{L'}'' \}.
\]

Let the reservation price be \( q^*_t \). From eq (8) \( q^*_t \) is derived by the following cases:

Case 1: \( Z_{L'} = Z_{L'}^x \). For \( t \in [1, x] \) let \( q^*_t \) either be the solution of

\[
Z_{L'} f_t(q^*_t) = f_{t+1}(M),
\]

or

\[
q^*_t = m \quad \text{if no solution exists.}
\]
Case 2: $Z_L' = Z''_L$. Let $t^* = \max\{t|f_{t+1}(M) \geq \sqrt{f_2(M) \cdot f_x(m)}\}$.

Case 2.1: For $\min\{t^*, x-1\} < t \leq x$,

$$q_t^* = m. \quad (10)$$

Case 2.2: For $1 \leq t < \min\{t^*, x-1\}$ let $q_t^*$ be either the solution of

$$Z_L' f_t(q_t^*) = f_{t+1}(M), \quad (11)$$

or

$$q_t^* = m \text{ if no solution exists.}$$

Theorem 3. Algorithm 3 is $Z_L'$ competitive.

The proof for the competitive ratio $Z_L'$, discussing several cases and worst-case time series, is not given here due to its length. The reader is referred to Xu et al. [38], Section 4.2.

For the problem considering different return functions, an extension of the current work can possibly be to design randomized algorithms to achieve a better competitive ratio.

3.1.4. Problem: uni|non-pmtn|M,m,T,f(q_t)

In the previous section, we did not consider the knowledge of duration $T$.

Based on this additional knowledge, Xu et al. [38] proposed a second $RP$ algorithm which is presented in the following. Assumptions as well as the proposed algorithm are identical to Algorithm 3. Only the calculation of the $RP$ $q_t^*$ differs.

Algorithm 4. On day $t$, accept price $q_t$ if $q_t \geq q_t^*$ resulting in a return of $f_t(q_t)$.

Calculating Reservation Price $q_t^*$

For each (known) duration $T$, let

$$Z = \min \left\{ \left\{ \max \left( \frac{f_{t+1}(M)}{f_t(m)}, \sqrt[2]{\frac{f_2(M)}{f_t(m)}} \right) \right\}, t = 1, \ldots, T - 1 \right\}, \sqrt[2]{\frac{f_2(M)}{f_T(m)}} \quad (12)$$
with $T \geq 1$ as $f_{t+1}(M) > f_t(m)$ and $f_2(M) > f_t(m)$. By definition of $Z$
there exists a natural number $y$, such that

$$Z' = \frac{f_{y+1}(M)}{f_y(m)} \text{ for } y \leq T - 1,$$

or

$$Z'' = \sqrt{\frac{f_2(M)}{f_y(m)}} \text{ for } y \leq T,$$

with

$$Z = \min\{Z', Z''\}.$$  

From eq(13) the RP $q^*_t$ is derived by the following cases:

**Case 1:** $Z = Z'$. For $t \in [1, y]$ let $q^*_t$ either be the solution of

$$Z f_t(q^*_t) = f_{t+1}(M)$$

or

$$q^*_t = m \text{ if no solution exists.}$$

**Case 2:** $Z = Z''$. Let $t^* = \max\{t | f_{t+1}(M) \geq \sqrt{f_2(M) \cdot f_y(m)}\}$.

**Case 2.1:** For $\min\{t^*, y-1\} < t \leq y$,

$$q^*_t = m.$$  

**Case 2.2:** For $1 \leq t < \min\{t^*, y-1\}$ let $q^*_t$ be either the solution of

$$Z f_t(q^*_t) = f_{t+1}(M),$$

or

$$q^*_t = m \text{ if no solution exists.}$$  

**Theorem 4.** Algorithm 4 is $Z$ competitive.

The proof for the competitive ratio $Z$, discussing several cases and worst-case
time series, is not given here due to its length. The reader is referred to Xu et al. [38], Section 3.2.
3.2. Uni-directional Preemptive Conversion

In uni-directional preemptive conversion, asset $D$ can be converted in parts with the possibility to convert at different points of time during the time interval, i.e. $s_t \in [0, 1]$. The only restriction is that during the time interval the player must convert asset $D$ into the asset $Y$ completely, i.e. $\sum_{t=1}^T s_t = 1$.

A great deal of literature addresses the problem of uni-directional preemptive search. El-Yaniv et al. [13, 14] introduced a genre of algorithms based on the assumption that there exists a threat that at some stage during the time interval, namely on day $k \leq T$, the offered price will drop to a minimum level $m$, and will remain there until the last day $T$. The algorithm proposed is commonly referred to as the threat-based strategy [14, p. 109].

Algorithm 5. The basic rules of the threat-based algorithm are:

Rule 1. Consider a conversion from asset $D$ into asset $Y$ only if the price offered is the highest seen so far.

Rule 2. Whenever you convert asset $D$ into asset $Y$, convert just enough $D$ to ensure that a competitive ratio $c$ would be obtained if an adversary dropped the price to the minimum possible price $m$, and kept it there afterwards.

Rule 3. On the last trading day $T$, all remaining $D$ must be converted into $Y$, possibly at price $m$.

El-Yaniv et al. [13, 14] discussed four variants of the above algorithm, each assuming a different knowledge about the future. Dannoura and Sakurai [11] improved the algorithm by improving the lower bound given in El-Yaniv et al. [13, 14]. It is shown that the threat is $c \cdot m \geq m$ (where $c \geq 1$ is the competitive ratio), and not $m$ as assumed by El-Yaniv et al. [13, 14].

Further variants of the threat-based algorithm can be found in the literature. Chen et al. [9] considered a price function $g(q_t)$. Each ‘next’ price $q_{t+1}$ depends on the current price $q_t$ in a geometric manner: $q_t/B \leq q_{t+1} \leq A \cdot q_t$, where $A$ and $B$ are constants. It is assumed that $T$, $A$ and $B$ are known a priori to the player.

Hu et al. [18] suggested two algorithms assuming the fluctuation ratio $\phi = M/m$, and $T$ is known. The first algorithm (static mixed strategy) is deemed to be overly pessimistic since it fixes the competitive ratio based on the
assumption of a worst-case input sequence of prices, and does not change it thereafter. Thus, they offered a second algorithm (dynamic mixed strategy) which converts based on the number of remaining days \( T' = T - t + 1 \), and the fluctuation ratio \( \phi \). Thus, the competitive ratio is improved by recalculating the achievable competitive ratio.

Damaschke et al. [10] assumed prior knowledge of \( m, M(t) \) and \( T \). The original threat-based algorithm by El-Yaniv et al. [13, 14] is improved by assuming that the upper bound is a decreasing function of time, i.e. \( M(t) = M/t \), and the lower bound \( m \) is constant.

Lorenz et al. [27] studied the max- (min-) search problem, and provided solution based on \( u \)-preemption and reservation prices. It is assumed that a player wants to convert at a specific number of prices \( u \). The problem setting assumed that \( m \) and \( M \) are known.

The above algorithms are described in detail in the following text.

3.2.1. Problem: \( \text{uni|pmtn|M,m,k} \)

El-Yaniv et al. [13, 14] presented a threat-based strategy that works on rules 1 to 3 as described in Algorithm 5. With known \( m, M \) and \( k \leq T \) the algorithm achieves a pre-calculated competitive ratio \( c \). Let \( d_t \) be the amount of asset \( D \) remaining after day \( t \), and \( y_t \) be the amount of asset \( Y \) accumulated after day \( t \). In order to achieve the competitive ratio \( c \), the amount to be invested at time \( t \), denoted by \( s_t \), must be determined such that \( c \) holds in case the price drops to \( m \), i.e. the worst-case occurs.

**Lemma 1.** If \( A \) is a \( c \)-competitive threat-based algorithm then for every \( t \geq 1 \)

\[
\begin{align*}
s_t &= \frac{q_t - c \cdot (y_{t-1} + d_{t-1} \cdot m)}{c \cdot (q_t - m)} \quad (17) \\
n_t &= y_{t-1} + m \cdot d_{t-1} \cdot (t - 1) + s_t \cdot (q_t - m). \quad (18)
\end{align*}
\]

**Proof.** The threat-based algorithm ensures that at time \( t \), enough \( D \) is converted to achieve the pre-specified competitive ratio \( c \). Thus

\[
\begin{align*}
\frac{OPT}{ON} &= \frac{q_t}{y_t + m \cdot d_t} \\
&= \frac{q_t}{(y_{t-1} + s_t \cdot q_t) + m \cdot (d_{t-1} - s_t)} \\
&\leq c.
\end{align*}
\]
The denominator $y_t + m \cdot d_t$ represents the return of ON if an adversary drops the price to $m$ and the nominator $q_t$ is the return of OPT for this case, as $q_t$ is the maximum and OPT will invest all $D$ at price $q_t$. According to rule 3 ON must spend the minimum $s_t$ that satisfies eq (19). Solving eq (19) as an equality constraint with respect to $s_t$ results in eq (17). Thus, for $t = 1$ we get

$$s_1 = \frac{1}{c} \cdot \frac{q_1 - c \cdot m}{q_1 - m} \tag{20}$$

as $d_0 = 1$ and $y_0 = 0$. Using eq (18) we get

$$s_t = \frac{1}{c} \cdot \frac{q_t - q_{t-1}}{q_t - m} \tag{21}$$

\[\]

**Definition 1.** A threat-based algorithm $A_c$ is $c$-proper iff

1. $\sum_{t=1}^{T} s_t \leq 1$,
2. $\frac{OPT(Q)}{A_c(Q)} \leq c$,

where $Q$ is the sequence of prices offered to the online player (algorithm).

**Lemma 2.** Let $Q$ be the sequence of offered prices. If algorithm $A_c$ is $c$-proper with respect to $Q$, then for any $c' \geq c$, algorithm $A_c'$ is $c'$-proper.

**Proof.** We assume that $Q = q_1, q_2, \ldots, q_k, m, m, \ldots, m$ with $m < q_1 < q_2 < \ldots < q_k$ and $t = 1, \ldots, T$. At any given time $t$, the amount converted $s_t$ by $A_c$ is smaller than or equal to the amount converted $s'_t$ by $A_c'$. Using eq (20), on day $t = 1$

$$s_1 - s'_1 = \frac{q_1}{(q_1 - m)} \left( \frac{1}{c} - \frac{1}{c'} \right) \geq 0, \tag{22}$$

and for $t > 1$

$$s_t - s'_t = \frac{q_t - q_{t-1}}{(q_t - m)} \left( \frac{1}{c} - \frac{1}{c'} \right) \geq 0. \tag{23}$$

As $\sum_{t=1}^{T} s'_t \leq \sum_{t=1}^{T} s_t$, and as $A_c$ is $c$-proper $\sum_{t=1}^{T} s_t \leq 1$. Hence, $\sum_{t=1}^{T} s'_t \leq 1$. As the competitive ratio $c'$ is achievable $A_c'$ selects transactions that ensure a competitive ratio $c'$, even if the prices drop to $m$. Hence, $A_c'$ is $c'$-proper. ■
3.2.2. Problem: uni|pmtn|M,m

El-Yaniv et al. [13, 14] addressed the scenario where the player knows only the lower and upper bound, \( m \) and \( M \), of the offered prices and presented a threat-based strategy. The basic rules of the strategy remain the same as discussed in Algorithm 5. As the player is oblivious about the time interval \( T \), it is assumed that the adversary selects \( T \to \infty \). Let \( A_{c}^{\infty} \) be the algorithm, then as per Lemma 2, the algorithm \( A_{c}^{\infty} \) is \( c^{\infty} \)-proper for any input sequence \( Q \), and hence \( c^{\infty} \) is an attainable competitive ratio. We now calculate \( c^{\infty} \), using \( c \cdot m \) as lower bound.

Let \( X = \frac{m \cdot (c-1)}{M-m} \), then

\[
\lim_{T \to \infty} T(1 - X^{1/T}) = \lim_{T \to \infty} cT(m, M) = \lim_{T \to \infty} \frac{X^{1/n} \cdot \ln X/T^2}{-1/T^2} \quad \text{[Using L'Hopital's Rule]}
\]

\[
= \lim_{T \to \infty} X^{1/n} \cdot \ln X = -\ln X.
\]

Thus \( c^{\infty}(m, M) \) is the unique solution \( c \), and

\[
c = \ln \frac{M}{m} - \frac{1}{c - 1},
\]

It can be seen that \( c^{\infty} = O(\ln \phi) \), where \( \phi = M/m \).

Dannoura and Sakurai [11] improved the lower bound presented by El-Yaniv et al.[13, 14], and suggested a more competitive algorithm. They claimed that a player using the algorithm of [13, 14] assumes a much greater threat than actually faced by the player. The threat assumed by [13, 14] is that the price might drop to \( m \), and will remain there for the rest of the time interval. Dannoura and Sakurai observed that the proposed algorithm suggested by El-Yaniv et al. does not convert unless the price is as large as \( c \cdot m \), i.e. the threat is at most \( c \cdot m \), and shall not go beyond this point. Thus \( c^{\infty}(m, M) \) is unique solution of \( c \), and

\[
c = \ln \frac{M}{c \cdot m} - \frac{1}{c - 1}.
\]

3.2.3. Problem: uni|pmtn|M,m,q1

El-Yaniv et al. [13, 14] and Dannoura and Sakurai [11] addressed the scenario where the player knows the lower and upper bound, \( m \) and \( M \), of
the offered prices, as well as the first price $q_1$, and presented a threat-based strategy. The basic rules of the strategy remain the same as discussed in Algorithm 5. Although we know $q_1$, the same $c$ is reached as in the case we would not know it (cf. Problem: uni|pmtn|M,m). So the knowledge of $q_1$ does not improve the competitive ratio, and eq (25) holds.

For calculating the competitive ratio $c$, an arbitrary number of trading days $T \to \infty$ is considered. Thus $c^\infty(m,M,q_1)$ is the unique solution of $c$.

$$c = \begin{cases} 
\ln \frac{\frac{M}{m} - 1}{c-1} & q_1 \in [m,c_m] \\
1 + \frac{q_1 - m}{q_1} \ln \frac{M-m}{q_1-m} & q_1 \in [c_m,M].
\end{cases} \quad (27)$$

3.2.4. Problem: uni|pmtn|\phi

El-Yaniv et al. [13, 14] addressed the scenario where the player knows only the price fluctuation ratio, $\phi = M/m$, of the offered prices, and presented a threat-based strategy. The basic rules of the strategy remain the same as discussed in Algorithm 5. As the player does not know $T$, the player assumes the adversary to choose $T \to \infty$. El-Yaniv et al. [13, 14] computed the optimal achievable competitive ratio to be $c^\infty(\phi)$, and is calculated as follows. Let $c^\infty(\phi) = \lim_{T \to \infty} c_T(\phi)$, then

$$\lim_{T \to \infty} \frac{(\phi - 1)^T}{(\phi^{T/(T-1)} - 1)^{T-1}} = (\phi - 1) e^{\frac{-\phi \ln \phi}{\phi - 1}}. \quad (28)$$

Therefore

$$c^\infty(\phi) = \phi \left(1 - (\phi - 1) e^{-\frac{\phi \ln \phi}{\phi - 1}}\right) \quad (29)$$

3.2.5. Problem: uni|pmtn|\phi,k

In this scenario, the online player along with the duration $k (k \leq T)$ knows only the fluctuation ratio $\phi = M/m$, but the real bounds on $M$ and $m$ are not known. The basic rules of the strategy remain the same as discussed in Algorithm 5. El-Yaniv et al. [13, 14] discussed the scenario, and observed that minimum price offered on day $t$ is at least $q_t/\phi$. Using eq (17) and (18), and replacing the minimum possible price in these equations by $q_t/\phi$ from eq
(18), we get
\[ y_t + d_t(q_t/\phi) = q_t/c \]
\[ \Rightarrow d_t = \phi \left( \frac{1}{c} - \frac{y_t}{q_t} \right). \]  
(30)

From eq (17), we get
\[ s_t = \frac{q_t - c(y_{t-1} + d_{t-1} \cdot q_t/\phi)}{c(q_t - q_t/\phi)} \] 
(31)

On day \( t = 1 \), we know that \( y_0 = 0 \), and \( d_0 = 1 \). Thus
\[ s_1 = \frac{\phi - c}{c(\phi - 1)}. \]

Similarly, for \( t > 1 \), we have
\[ s_t = \frac{y_{t-1}\phi}{\phi - 1} \left( \frac{1}{q_{t-1}} - \frac{1}{q_t} \right) \] 
(33)

**Theorem 5.** Competitive ratio of threat-based algorithm with \( \phi \) and \( k \) known is:
\[ c(\phi, k) = \phi \left( 1 - (\phi - 1)^k / \left( \phi^{k/(k-1)} - 1 \right) \right)^{k-1} \] 
(32)

For proof of Theorem 5, the reader is referred to El-Yaniv et al. [14] Section 4.4.

3.2.6. Problem: uni|pmtn|\( M(t), m, T \)

Damaschke et al. [10] assumed that the player knows the lower and upper bounds of the offered prices, \( m \) and \( M(t) \), as well as the duration \( T \). Their model is based on the assumption that the upper bound is not constant but varies with time (\( M(t) = M/t \)). Damaschke et al. presented a threat-based strategy, the basic principle remains the same as described in Algorithm 5.

Let \( s_t \) be the amount converted at time \( t \), then
\[ s_t = \begin{cases} \frac{1}{c} \left( \frac{q_t - c m}{q_t - m} \right) & t = 1 \\ \frac{1}{c} \left( \frac{q_t - q_{t-1}}{q_t - m} \right) & t \in [2, T] \end{cases} \] 
(33)
Theorem 6. The competitive ratio $c$ achieved is

$$c = \max_{k=2 \ldots T} \left\{ c | c = k \left(1 - \left(\frac{c - 1}{M(k)/m} - 1\right)^{1/k}\right) \right\}$$  (34)

where $q_t$ is price offered to the player at time $t$, and is modeled as $m \leq q_t \leq M(t)$, where $M(t)$ is decreasing function of time and $m$ is constant.

3.2.7. Problem: uni|pmtn|φ, T

In this scenario, the online player, along with the knowledge of duration $T$ knows only the fluctuation ratio $\phi = M/m$ but the real bound on $M$ and $m$ are not known. Hu et al. [18] presented two algorithms to achieve optimal competitive ratio under worst case assumptions, namely the Static Mixed Strategy and the Dynamic Mixed Strategy.

Static Mixed Strategy: The static mixed strategy allocates the amount to be converted based on the worst-case input sequence of prices.

Algorithm 6. Determine the amount to be converted at time $t$ by the following rules

$$s_t = \begin{cases} \frac{1+\phi}{(T-1)\phi+2} & t = 1 \\ \frac{\phi}{(T-1)\phi+2} & t \in [2, T-1] \\ \frac{1}{(T-1)\phi+2} & t = T \end{cases}$$  (35)

Theorem 7. The competitive ratio $c$ achieved by Algorithm 6 is

$$c = 1 + \frac{\phi}{2} (T - 1)$$  (36)

For the proof of Theorem 7, the reader is referred to Hu et al. [18] Theorem 1.

Dynamic Mixed Strategy: The worst-case scenario does not occur that frequently as assumed by the static mixed strategy. The dynamic mixed strategy addresses this issue, and allocates $s_t$ based on the remaining number of days $T'$ in the time interval.
Algorithm 7. Determine the amount to be converted at time \( t \) by the following rules

\[
s_t = \begin{cases} 
    \left( \frac{1+\phi}{(T-1)\phi+2} \right) W'_t & t = 1 \\
    \frac{\phi}{(T-1)\phi+2} W'_t & t \in [2, T - 1] \\
    \frac{1}{(T-1)\phi+2} W'_t & t = T
  \end{cases}
\]  

(37)

where \( W'_t \) denotes the remaining amount of wealth at day \( t \).

Theorem 8. The competitive ratio \( c \) achieved by Algorithm 7 based on the remaining number of days \( T' \) is

\[
c = 1 + \frac{(T' - 1)\phi}{2}.
\]

(38)

For the proof of Theorem 8, the reader is referred to Hu et al. [18].

The dynamic mixed strategy is more competitive than the static mixed strategy but the competitiveness does not exist when the the duration \( T \) is extended to infinity, therefore designing a strategy which works independent of the duration \( T \) is an open question. In addition, investigating bi-directional strategy, and incorporating transaction cost also requires further research.

3.2.8. Problem: uni\|pmtn\|T,g(q_t)

Chen et al. [9] presented an algorithm for uni-directional search. The model assumes prior knowledge of the duration \( T \), and the price function \( g(q_t) \). The constants \( A \) and \( B \) (\( A, B \geq 1 \)) determine the prices offered on a day \( t \), and \( q_t \) is modeled as \( q_{t-1}/B \leq q_t \leq A \cdot q_{t-1} \). The algorithm and the amount invested \( s_t \) on day \( t \) is described as follows:

Algorithm 8. Determine the amount to be converted at time \( t \) by the following rules

\[
s_t = \begin{cases} 
    \frac{A(B-1)}{(A-1)(B-1)} t = 1 \\
    \frac{AB-(T-1)(A+B)+(T-2)}{(A-1)(B-1)} t \in [2, T - 1] \\
    \frac{AB-(T-1)(A+B)+(T-2)}{AB} t = T
  \end{cases}
\]  

(39)

Theorem 9. The competitive ratio \( c \) achieved by Algorithm 8 is

\[
c = \frac{TAB - (T - 1) (A + B) + (T - 2)}{AB - 1}
\]

(40)
For proof of Theorem 9, the reader is referred to Chen et al. [9] Theorem 3.4.

The problem requires further investigation where there is a continuous flow of wealth/cash instead of one time fixed cash. Similarly replacing the constants \(A\) and \(B\) with some known probability distribution can also be investigated.

3.2.9. Problem: uni\(\)\textit{pmtn}\(\)\(M,m,a\)

The threat-based algorithm presented by El-Yaniv et al, [13, 14] (and its variants) attempts to safe guard against a clever adversary who might drop the offered prices at some point during the time interval to the lowest level \(m\), and keep it there for the rest of the time interval. The threat-based strategy is thus risk-averse, i.e. it mitigates the amount of risk involved, and provides a solution that ensures an optimal competitive ratio under worst case assumption. Al-Binali [1] introduced the concept of risk management, and presented a risk-reward framework. The main idea is to allow the player to manage his risk for some kind of reward, and to allow the player to develop a trading algorithm based on risk tolerance and forecast. A forecast is the prospected value of the price that might be reached in the time interval. The forecast can either be on the maximum value in the future (‘above forecast’ \(M_1\)) or on the minum value in the future (‘below forecast’). Iwama and Yonezawa [20] presented an extension of the threat-based algorithms using generalized forecasts and incorporating a risk tolerance level of the player. In general, the risk-reward threat-based algorithms are based on the scenario where a single above forecast is assumed. They also discussed scenarios where ‘double above forecast’ and ‘single above and below forecast’ are assumed. They are natural extensions of the more generalized single above forecast.

The algorithm runs in two phases, phase 1 assumes that the forecast will not come true and thus enough wealth is converted to ensure a competitive ratio \(a \cdot c_0\). Phase 2 starts when the forecast becomes true, at this stage a new competitive ratio \(c_1\) is computed, and the wealth is converted at offered prices to achieve \(c_1\). The formal algorithm is outlined as follows. Assume the starting price \(q_0\) is greater than \(c \cdot m\) \((q_0 \geq c \cdot m\), and \(M_1\) is the forecasted upper bound.

\textbf{Algorithm 9.} \(q_t \in [q_0, M_1]\) : \textit{Convert just enough to ensure a competitive ratio of } \(a \cdot c_0\) \textit{is achieved.}
\[ c_0 = \ln \left[ \frac{M - m}{c_0 m - m} \right], \quad (41) \]
\[ d_1(q_t) = 1 - \left( \frac{1}{ac_0} \right) \ln \left[ \frac{q_t - m}{ac_0 m - m} \right], \quad (42) \]
\[ y_1(q_t) = \frac{1}{ac_0} \left[ m \cdot \ln \frac{q_t - m}{ac_0 m - m} + q_t - ac_0 m \right]. \quad (43) \]

\( q_t \in [M_1, M] \): compute the new competitive ratio \( c_1 \) (better than \( c_0 \)), and convert just enough to achieve this ratio. Let \( d_2(x) \) and \( y_2(x) \) be the amounts of dollars and yen in this phase. Then
\[ d_2(q_t) = d - \left( \frac{1}{c_1} \right) \ln \left[ \frac{q_t - m}{M_1 - m} \right], \quad (44) \]
\[ y_2(q_t) = y + \frac{1}{c_1} \left[ m \cdot \ln \frac{q_t - m}{M_1 - m} + q_t - M_1 \right]. \quad (45) \]

In eq (44), and (45), \( d \) is dollars and \( y \) is the amount of yens at hand, given by
\[ d = d_1(M_1) - \left( \frac{M_1}{M_1 - m} \right) \left( \frac{1}{c_1} - \frac{1}{ac_0} \right), \quad (46) \]
and
\[ y = y_1(M_1) - \left( \frac{M_1}{M_1 - m} \right) \left( \frac{1}{c_1} - \frac{1}{ac_0} \right). \quad (47) \]

The optimal strategy enforces the condition that all dollars must be converted, such that \( d_2(M) = 0 \) or
\[ 1 - \frac{1}{ac_0} \ln \frac{M_1 - m}{ac_0 m - m} - \frac{M_1}{M_1 - m} \left( \frac{1}{c_1} - \frac{1}{ac_0} \right) - \frac{1}{c_1} \ln \frac{M - m}{M_1 - m} = 0 \quad (48) \]

By solving eq (48), we get the competitive ratio \( c_1 \)
\[ c_1 = \frac{M_1 - m}{(M_1 - m) \left( 1 - \frac{1}{ac_0} \ln \frac{M_1 - m}{ac_0 m - m} \right) + \frac{M_1}{ac_0} \left( \frac{M_1}{M_1 - m} + \ln \frac{M - m}{M_1 - m} \right)}. \quad (49) \]

The work is based on the simple assumption that a forecast can either be true or false. However in practice a forecast has an associated probability \( \rho \) to become true, so the reward can be represented as function of \( \rho \) when the forecast becomes true.
3.2.10. Problem: uni|u-pmtn|M,m

Lorenz et al. [27] designed a strategy for u−pmtn with m and M known.

Two different strategies are proposed one each for buying and selling.

Algorithm 10. 1. Max-search (selling) Problem: At the start of the game compute reservation prices $q^*_i = (q^*_1, q^*_2, \ldots q^*_u)$, where $i = 1, \ldots, u$. As the adversary unfolds the prices, the algorithm accepts the first price which is at least $q^*_1$. The player then waits for the next price which is at least $q^*_2$, and so on. If there are still some units of asset left on day T, then all remaining units must be sold at the last offered price, which may be at the lowest price $m$.

$$q^*_i = m \left[ 1 + (c^* - 1) \left( 1 + \frac{c^*}{u} \right)^{i-1} \right]$$ (50)

Where $c^*$ is the competitive ratio for the max-search (selling) problem.

2. Min-search (buying) Problem: Follows the same procedure as for max-search problem, the reservation prices are computed as follows:

$$q^*_i = M \left[ 1 - \left( 1 - \frac{1}{c^*} \right) \left( 1 + \frac{1}{u \cdot c^*} \right)^{i-1} \right]$$ (51)

Where $c^*$ is the competitive ratio for the min-search (buying) problem.

Theorem 10. Let $u \in N$, $\phi > 1$, there exists a $c^*$-competitive deterministic algorithm for u max-search problem where $c^* = c^*(u, \phi)$ is the unique solution of

$$\frac{\phi - 1}{(c^* - 1)} = \left( 1 + \frac{c^*}{u} \right)^u.$$

Theorem 11. Let $u \in N$, $\phi > 1$, there exists a $c^*$-competitive deterministic algorithm for u min-search problem where $c^* = c^*(u, \phi)$ is the unique solution of

$$\frac{1 - \frac{1}{\phi}}{(1 - \frac{1}{c^*})} = \left( 1 + \frac{1}{c^* \cdot u} \right)^u.$$
4. Bi-directional Search

Bi-directional search allows the player to convert asset $D$ into asset $Y$, and asset $Y$ back into asset $D$ during a time interval. We assume that the objective is to maximize the amount of $D$ at day $T$, i.e. the player has the objective to maximize his final wealth in terms of asset $D$. We classify the bi-directional search problem into two main classes based on the amount of wealth converted.

4.1. Bi-directional Non-Preemptive

Bi-directional non preemptive algorithms allow the player to conduct bi-directional search with the restriction to convert the whole amount of wealth at one point during a conversion. This implies that only two transactions are permissible during a single trade. This however, does not restrict the player to trade only once in the time interval, the player can either trade only once (single trading), and can repeat the trading (buying followed by selling) as many times (multiple trading) as he wishes. Kao and Tate [22] presented an algorithm for profit maximization (named difference maximization), Mohr and Schmidt [31] extended the reservation price algorithm for selling by El-Yaniv [12] to buying and selling.

4.1.1. Problem: bi|non-pmtn| –

i. Algorithm by Kao and Tate [22]

Kao and Tate [22] presented a solution to the bi-directional search problem without any assumptions made regarding the future. The prices are arbitrary real numbers, for each price $q_t$, a rank $x_t$ is calculated. The value of $x_t$ represents the rank of $q_t$ in the already observed sequence of prices. The algorithm attempts to achieve the maximum possible profit by buying at low and selling at high prices while maximizing the difference in ranks between the buying and selling prices.

The authors addressed two scenarios, the first scenario is called single pair selection, solves the single trade problem and the second scenario is called multiple pair selection, solves the multiple trade problem.

• Single pair selection: The player is allowed to make two selections, one for buying (low selection) $q_l$, and one for selling (high selection) $q_h$. The difference ($q_h - q_l$) is the profit. Alternatively, the profit can also be the difference in the rank of two selections, i.e. $x_h - x_l$. 
Multiple pair selection: The player is allowed to make multiple low and high selections during the time interval. The sum of the differences thus is the profit.

No assumptions are made regarding the distribution of the sequence of prices. It is obvious to assume that all permutations of the final ranks are equally likely. If the rank of a price $q_t$ is $x_t$ among the first $t$ prices, then the expected final rank will be $\left(\frac{T+1}{T+1}\right) x_t$.

Let $H_T(t)$ be a high selection limit, and $R_T(T)$ the expected final rank of the high selection if the optimal algorithm $OPT$ is followed starting at the time $t$. Let $L_T(t)$ be a low selection limit, and $P_T(t)$ be the expected high-low difference if the optimal algorithm $OPT$ for making the low and high selections is followed starting at time $t$, with

$$P_T(t) = \begin{cases} 0 & t = T, \\ P_T(t+1) + \frac{L_T(t)}{t} \cdot \left( R_T(t+1) - P_T(t+1) - \frac{T+1}{T+1} \cdot \frac{L_T(t+1)}{2} \right) & t < T. \end{cases}$$

$$H_T(t) = \left\lceil \frac{t+1}{T+1} \cdot R_T(t+1) \right\rceil. \tag{53}$$

$$L_T(t) = \begin{cases} 0 & t = T, \\ \left\lfloor \frac{t+1}{T+1} \cdot (R_T(t+1) - P_T(t+1)) \right\rfloor & t < T. \end{cases} \tag{54}$$

If no selection is made before the last offered price $q_T$, the last price $q_T$ has to be accepted with rank $R_T(T) = \frac{n+1}{2}$.

Kao and Tate [22] stated that the competitive ratio for single pair selection equals one, and for multiple pair selection equals $\frac{4}{3}$. The proof for the competitive ratios is not given here due to its length. The reader is referred to Kao and Tate [22], Section 3. Further work can be carried out by investigating to maximize quantities other than the difference in rank.
ii. Heuristic Conversion Algorithms

In the following we present the competitive analysis of three heuristic conversion algorithms, namely Moving Average Crossover (MA), Trading Range Breakout (TRB), and Momentum (MM) which are based on technical indicators.

In general, heuristic conversion algorithms are also reservation price (RP) algorithms. Reservation price(s) are calculated based on the offered price(s) \( q_t \). Using the RP, the algorithm determines intersection points specifying when to buy or sell.

For each \( i \)-th trade we assume a worst-case time series of prices containing only minimum prices \( m(i) \), and maximum prices \( M(i) \). At best the considered algorithm buys at price \( m(i) \), and sells at price \( M(i) \) resulting in an optimum return \( OPT = M(i)/m(i) \). In the worst-case the above heuristic conversion algorithms \( ON \in \{MA, TRB, MM\} \) achieve the worst possible return of \( ON = m(i)/M(i) = 1/OPT \), resulting in a competitive ratio of

\[
c = \prod_{i=1}^{p} \left( \frac{M(i)}{m(i)} \right)^2 ,
\]

and in case \( m(i) \) and \( M(i) \) are constants

\[
c = \left( \frac{M}{m} \right)^{2p} .
\]

To prove the competitive ratio given in eq (56) we assume that an algorithm \( ON \in \{MA, TRB, MM\} \) is allowed to trade only once, i.e. \( p = 1 \).

Theorem 12. The competitive ratio of the heuristic conversion algorithms MA, TRB, and MM equals \( c = \left( \frac{M}{m} \right)^2 \).

1. Algorithms by Brock et al. [6]

Brock et al. [6] introduced the algorithms MA and TRB. These algorithms are of major interest in the literature, and have been empirically analyzed by several researchers, cf. Bessembinder and Chan [3]; Hudson et al. [19]; Mills [29]; Ratner and Leal [33]; Parisi and Vasquez [32]; Gunasekarage and Power [16]; Kwon and Kish [24]; Chang et al. [7]; Bokhari et al. [5]; Marshall and Cahan [28]; Ming-Ming and Siok-Hwa [30]; Hatgioannides and Mesomeris [17]; Lento and Gradojevic [26]; Lagoarde-Segot and Lucey [25];
1. Moving Average Crossover (MA).

Assume the following worst-case time series $m, \ldots, m, M, m, \ldots, m$. Hence, the prices $q_1, \ldots, q_{t-1} = m$, $q_0 = M$, and $q_{t+1}, \ldots, q_T = m$. The MA algorithm suggested by Brock et al. [6] is:

**Algorithm 12.** Buy on day $t$ if $MA(S)_t > uB(L)_t$ and $MA(S)_{t-1} \leq uB(L)_{t-1}$, and sell on day $t$ if $MA(S)_t < lB(L)_t$ and $MA(S)_{t-1} \geq lB(L)_{t-1}$.

Where $MA(S)_t$ is a short moving average, $MA(L)_t$ a long moving average $(S < L)$, and the value $n \in \{L, S\}$ defines the number of previous data points (days) considered to calculate $MA(n)_t = \frac{\sum_{i=t-n+1}^{t} q_i}{n}$. Prices $q_i$ are lagged by bands, the upper band is $uB(L)_t = MA(L)_t \cdot (1 + b)$, and the lower band is $lB(L)_t = MA(L)_t \cdot (1 - b)$ with $b \in [0.00, \infty]$.

**Proof of Theorem 12 for Algorithm 12:** Assume $S = 1, L \leq (t^* - 1)$, and $b = 0.00$. This corresponds to increasing prices generating a buy signal if the price crosses the long MA from below. Similarly, this corresponds to decreasing prices generating a sell signal if the price crosses the long MA from above. The MA algorithm

1. buys on day $t^*$ at price $q_{t^*} = M$. Because $MA(1)_{t^*} = q_{t^*} = M > uB(t^* - 1)_{t^*} = MA(t^* - 1)_{t^*} = \frac{(t^* - 2)m + M}{(t^* - 1)} < M$, and $MA(1)_{t^* - 1} = q_{t^* - 1} = m \leq uB(t^* - 1)_{t^* - 1} = MA(t^* - 1)_{t^* - 1} = \frac{(t^* - 1)m + M}{t^* - 1} = m$.

2. sells on day $t^* + 1$ at price $q_{t^* + 1} = m$. Because $MA(1)_{t^* + 1} = q_{t^* + 1} = m < lB(t^* - 1)_{t^* + 1} = MA(t^* - 1)_{t^* + 1} = \frac{(t^* - 3)m + M + m}{(t^* - 1)} > m$, and $MA(1)_{t^*} = q_{t^*} = M \geq lB(t^* - 1)_{t^*} = MA(t^* - 1)_{t^*} = \frac{(t^* - 2)m + M}{(t^* - 1)} < M$.

Taking these decisions into account algorithm MA achieves a return of $m/M$. Comparing this to the optimum return achieved by algorithm $OPT$, the worst-case competitive ratio equals $OPT/MA = (\frac{M}{m})^2$.

1.2. Trading Range Breakout (TRB).

Assume the following worst-case time series $m + \epsilon, \ldots, m + \epsilon, M, m, \ldots, m$. Hence, the prices $q_1, \ldots, q_{t-1} = m + \epsilon$, $q_0 = M$, and $q_{t+1}, \ldots, q_T = m$. The TRB algorithm suggested by Brock et al. [6] is:

**Algorithm 13.** Buy on day $t$ if $q_t > uB(n)_t$ and $q_{t-1} \leq uB(n)_{t-1}$, and sell on day $t$ if $q_t < lB(n)_t$ and $q_{t-1} \geq lB(n)_{t-1}$.
Where lower band \( lB(n)_t = q_{t-1}^{\min}(n) \cdot (1-b) \) with \( q_{t-1}^{\min}(n) = \min \{ q_i | i = t-n, \ldots, t-1 \} \), and upper band \( uB(n)_t = q_{t-1}^{\max}(n) \cdot (1-b) \) with \( q_{t-1}^{\max}(n) = \max \{ q_i | i = t-n, \ldots, t-1 \} \) where \( b \in [0.00, \infty) \), and \( n < t \) is the number of previous data points (days) considered.

**Proof of Theorem 12 for Algorithm 13:** Assume \( n \leq (t^*-2) \), and \( b = 0.00 \). This corresponds to increasing prices generating a buy signal if the price crosses \( uB \) from below. Similarly, this corresponds to decreasing prices generating a sell signal if the price crosses \( lB \) from above. The TRB algorithm achieves a return of \( m/M \). Comparing this to the optimum return achieved by algorithm OPT, the worst-case competitive ratio equals \( OPT/TRB = \left(\frac{M}{m}\right)^2 \). 

2. **Momentum (MM) [21]**

Assume the following worst-case time series \( m + \epsilon, m, \ldots, m, M, m, \ldots, m \).

Hence, the prices \( q_1 = m + \epsilon, q_2, \ldots, q_{t-1} = m, q_t = M \), and \( q_{t+1}, \ldots, q_T = m \). The MM algorithm suggested by Jagadeesh and Titman [21] is:

**Algorithm 14.** Buy on day \( t \) if \( MM_t(n) \geq 0 \) and \( MM_{t-1}(n) < 0 \), and sell on day \( t \) if \( MM_t(n) \leq 0 \) and \( MM_{t-1}(n) > 0 \).

Where the momentum \( MM_t(n) = q_t - q_{t-n+1} \), and \( n \leq t \) is the number of previous data points (days) considered.

**Proof of Theorem 12 for Algorithm 14:** Assume \( n \leq (t^*-1) \) and \( 0 < m < M \). This corresponds to increasing prices after a series of decreasing prices (trend revision) generating a buy signal if the MM crosses the zero line from below. Similarly, this corresponds to decreasing prices after a series of increasing prices (trend revision) generating a sell signal if the MM crosses the zero line from above. The MM algorithm
1. buys on day $t^*$ at price $q_{t^*} = M$. Because $MM_t^*(t^* - 1) = q_{t^*}^* - q_2 = M - m \geq 0$, and $MM_{t^* - 1}(t^* - 1) = q_{t^* - 1}^* - q_1 = m - (m + \epsilon) < 0$.

2. sells on day $t^* + 1$ at price $q_{t^* + 1} = m$. Because $MM_{t^* + 1}(t^* - 1) = q_{t^* + 1} - q_3 = m - m \leq 0$, and $MM_{t^*}(t^* - 1) = q_t^* - q_2 = M - m > 0$.

Taking these decisions into account algorithm $MM$ achieves a return of $m/M$. Comparing this to the optimum return achieved by algorithm $OPT$, the worst-case competitive ratio equals $OPT/MM = (\frac{M}{m})^2$. ■

Thawornwong et al. [35] gives a further heuristic conversion algorithm, called Relative Strength Index ($RSI$). Worst-case analysis can be done in the same manner; the worst-case time series used for $MA$ must be considered.

4.1.2. bi|$\text{non-pmtn}$|$M$,$m$

Schmidt et al. [34] extended the uni-directional reservation price algorithm for selling by [12] (cf. Problem: uni|$\text{non-pmtn}$|$M$,$m$) to buying and selling, i.e. introduce a rule for min-search. In this case the optimal deterministic bi-directional algorithm is the following $RPP$.

**Algorithm 15.** Buy at the first price smaller or equal, and sell at the first price greater or equal to reservation price $q^* = \sqrt{M \cdot m}$.

If $m$ and $M$ are constants, the worst-case competitive ratio assuming $p \geq 1$ trades then equals

$$c = \left( \frac{M}{m} \right)^p,$$

otherwise

$$c = \prod_{i=1}^{p} \left( \frac{M(i)}{m(i)} \right)$$

as for each $i$-th transaction ($i = 1, \ldots, p$) different upper bounds $M(i)$ and lower bounds $m(i)$ are assumed.

4.2. Bi-directional preemptive

Bi-directional preemptive allows player to follow either the single trade or multiple trade policy. El-Yaniv et al. [13, 14], and Danoura and Sakurai [11] extended their work for uni-directional preemptive search to allow bi-directional preemptive search.

30
4.2.1. bidpmtn|M,m

El-Yaniv et al. [13] considered bi-directional run search under the assumption that the upper and lower bounds, $M$ and $m$, on possible prices are known. To solve bi-directional problem, the player does not need to know the number of days $T \geq k$ (El-Yaniv et al. [14, p. 136]). The suggested algorithm divides the sequence of prices into upward and downward runs and repeats the uni-directional threat-based algorithm presented in Algorithm 5. Asset $D$ is converted into $Y$ (max-search) if the price is on an upward trend (run). $Y$ is converted into $D$ (min-search) if the price is on a downward trend (run). Assuming $p/2$ upward runs, and $p/2$ downward runs, the online investor achieves an overall competitive ratio of $c = (\ln(\frac{M}{m} - 1))^{p}$ as the overall number of $p$ trades is carried out [13, p. 7].

Dannoura and Sakurai [11] improved the bi-directional algorithm of [13] by making the threat smaller, and thus achieve a better competitive ratio $c = (\ln(\frac{M}{m} - 1))^{p}$. Dannoura and Sakurai [11] also improved the upper and lower bound for bi-directional run search given in the previous work of El-Yaniv et al. [13]. The improved algorithm is not yet optimal, thus the challenge of designing an optimal algorithm for bi-directional search remains [11, p. 33].

5. Conclusion

Though a considerable amount of work addresses the online conversion problems, a number of questions are still unanswered, and require further consideration. These questions relate to theoretical and practical aspects. In order to verify the applicability of the suggested algorithms to practical problems more experimental studies are required. From the experimental studies competitive ratios can be defined and compared to worst-case theoretical ratios. Especially information about future prices of a time series in most practical cases is not available. To apply the online conversion algorithms, we need estimates of this information which are necessarily bound to errors. It would be helpful to investigate competitive ratios which depend on given errors due to the input data of the algorithms. If we assume that information about the future is available it will be of great interest which information is more valuable, for instance the knowledge of the upper bound $M$, or the knowledge of fluctuation ratio $\phi$. Similarly an experimental study to investigate the worth of future information available may also be of inter-
est. Intuitively, the more information available to an algorithm, the better it should perform in the worst-case; e.g. an algorithm which utilizes $m$ and $M$ should perform better than then one which utilizes only $\phi$ as input. Experimental studies can be conducted to verify the claim.

A significant drawback of threat-based algorithms is the large number of transactions carried out. As in the real world, each transaction has an associated transaction fee, so the large number of transactions adversely affects the practical performance of these algorithms. Hence, designing a strategy that reduces the number of transactions while maintaining the competitive ratio needs further research. Similarly, the algorithms designed for bi-directional search do not perform optimally and pose themselves as an open question.

Al-Binali [1] introduced the notion of acceptable level of risk in terms of competitive ratio. When risk in terms of competitive ratio is considered, the question remains open if the competitive ratio is a coherent measure of risk [2] or not. Further, our proposed classification scheme can be used to address the unaddressed areas of online conversion problems.

References


[27] J. Lorenz, K. Panagiotou, and A. Steger. Optimal algorithms for k-

stock market which has characteristics that suggest it may be ineffi-

[29] T.C. Mills. Technical analysis and the london stock exchange: test-
ing trading rules using the FT30. *International Journal of Finance &

averages and trading range breakout in the asian stock markets. *Journal

[31] E. Mohr and G. Schmidt. Empirical analysis of an online algorithm for
multiple trading problems. In H.A. Le Thi, P. Bouvry, and T. Pham
Dinh, editors, *Modelling, Computation and Optimization in Informa-

[32] F. Parisi and A. Vasquez. Simple technical trading rules of stock returns:
164, September 2000.

emerging equity markets of latin america and asia. *Journal of Banking

[34] G. Schmidt, E. Mohr, and M. Kersch. Experimental analysis of an online
526, 2010.

[35] Cihan Dagli Suraphan Thawornwong, David Enke. Neural networks as
a decisionmaker for stocktrading: A technical analysis approach. *Interna-


6.4 Results of Mohr and Schmidt (2010)

Manuscript submitted for publication, November 30, 2010

Preface

Comparing an online conversion algorithm to the optimal offline algorithm can be thought of as measuring the value of the information of future prices (Larsen and Wøhlk, 2010, p. 685). Inspired by Karp (1992a,b) we answer the question ‘how much is it worth to know the future in online conversion problems’ using the competitive ratio as an indicator for the quality of information about the future. We define information to be more valuable if the worst-case competitive ratio can be improved by this information. We calculate the empirical-case competitive ratios of the different variants of the threat-based algorithm of El-Yaniv et al. (1992, 2001) (Algorithm 8, p. 92). Due to Rules (1) to (3) of Algorithm 8, for all variants of the threat-based algorithm, the prices considered for conversion are identical. Only the calculation of the amount to be converted $s_t$ differs based on the information assumed to be known a-priori.

For the empirical-case analysis transaction costs are not considered and the backtesting of the algorithms is done on the German Dax-30 index for the investment horizon 01-01-1998 to 12-31-2007; stylized facts are given in Example 2, p. 62. Only the index itself can be traded by the investigated algorithms $\text{ON} \in \{\text{Threat}(X), \text{BH}\}$ with $X \in \{(m, M, k), (m, M), (m, M, q_1), (\varphi, k), (\varphi)\}$, and $\text{OPT}$. The investment horizon is divided into several time intervals of different length $T$. Within each $T$ uni-directional search, solving either the \textit{min-search} problem for buying or the \textit{max-search} problem for selling, might be carried out. As suggested in the work of Borodin et al. (2004), again two consecutive time intervals of equal length $T$ built trading intervals of length $2 \cdot T$, with $T \in \{260, 130, 65, 20, 10\}$. In order to trade multiple times for example $2 \cdot T = 260$ days equal $T = 130$ days for buying and $T = 130$ days for selling, etc. The following questions are to be answered:

1. How do the worst-case competitive ratios $c^{wc}$ which could have been possible from the experimental data compare?

2. How do the empirical-case competitive ratios $c^{ec}$ found in the experiments compare?

3. Are the answers to Question 2 significant?
We compare our empirical-case results to the analytical worst-case results given in the literature. The empirical-case performance is evaluated by a $t$-test, as given in Algorithm 2, p. 67.

Analytical results show that the better the information the better the worst-case competitive ratios. However, experimental analysis gives a slightly different view. We show that better information does not always lead to a better performance in real-life applications. The empirical-case competitive ratio is not always better with better information, and some a-priori information is more valuable than other for practical settings. We conclude that the value of information can only be estimated by worst-case scenarios.
How Much is it Worth to Know the Future in Online Conversion Problems?

Esther Mohr\textsuperscript{a,*}, Günter Schmidt\textsuperscript{a,b}

\textsuperscript{a}Saarland University, P.O. Box 151150, D-66041 Saarbrücken, Germany
Phone +49-681-302-4559, Fax +49-681-302-4565
\textsuperscript{b}University of Liechtenstein, Fürst-Franz-Josef-Strasse, 9490 Vaduz, Liechtenstein

Abstract

We answer this question using the competitive ratio as an indicator for the quality of information about the future. Analytical results show that the better the information the better the worst-case competitive ratios. However, experimental analysis gives a slightly different view. We calculate the empirical-case competitive ratios of different variants of a threat-based online algorithm. The results are based on historical Dax data. We compare our empirical-case results to the analytical worst-case results given in the literature. We show that better information does not always lead to a better performance in real life applications. The empirical-case competitive ratio is not always better with better information, and some a-priori information is more valuable than other for practical settings.

Keywords: Online Algorithms, Competitive Analysis, Empirical-case Analysis, Worst-case Analysis, One-way Trading, Uni-directional Algorithm

1. Introduction

[1] answers the question considering multiprocessor scheduling, interval coloring, and the $k$-server problem. We want to answer the question for online conversion problems. A conversion problem deals with the scenario of converting an asset $D$ into another asset $Y$ with the objective to get the maximum amount of $Y$ after time $T$. The process can be repeated in both
directions, i.e. converting asset $D$ into asset $Y$, and asset $Y$ back to asset $D$. On each day $t$, the player is offered a price $q_t$ to convert $D$ to $Y$; he may accept the price $q_t$ or may decide to wait for a better price. The game ends when the player converts whole of the asset $D$ to $Y$.

Based on the amount converted $s_t$, two classes of online conversion algorithms exist, (i) preemptive online conversion algorithms - designed to convert asset $D$ at more than one price within the time interval, i.e. $s_t \in [0, 1]$, and (ii) non-preemptive online conversion algorithms - designed to convert asset $D$ at one single price within the time interval, i.e. $s_t \in \{0, 1\}$.

Several authors suggest uni-directional preemptive algorithms for (i) using the competitive ratio as performance measure [2, 3, 4, 5, 6, 7, 8]. An algorithm must determine which amount $s_t \in [0, 1]$ to be converted on days $t = 1, \ldots, T$ such that the amount of asset $Y$ is maximized on day $T$. The only restriction is that during the time interval the player must convert asset $D$ into the asset $Y$ completely, i.e. $\sum_{t=1}^{T} s_t = 1$, and conversion back to $D$ is forbidden.

Related work focuses on worst-case performance guarantees using competitive analysis [9]. The performance of an online algorithm $\text{ON}$ is compared to that of an adversary, the optimal offline algorithm $\text{OPT}$. Each input can be represented as a finite sequence $I$ with $t = 1, \ldots, T$ elements, and a feasible output can also be represented as a finite sequence with $T$ elements. An algorithm $\text{ON}$ computes online if for each $t = 1, \ldots, T - 1$, it computes an output for $t$ before the input for $t + 1$ is given. An algorithm $\text{OPT}$ computes offline if it computes a feasible output given the entire input sequence $I$ in advance. An online algorithm $\text{ON}$ is $c$-competitive if for any input $I$

$$\text{ON}(I) \geq \frac{1}{c} \cdot \text{OPT}(I).$$

(1)

If the competitive ratio is related to a performance guarantee it must be a worst-case measure. Any $c$-competitive algorithm $\text{ON}(I)$ is guaranteed a value of at least the fraction $1/c$ of the optimal offline value $\text{OPT}(I)$ no matter how unfortunate or uncertain the future will be. We consider converting assets as a maximization problem, i.e. $c \geq 1$. The smaller $c$ the more effective is algorithm $\text{ON}$.

In case the input data processed by an online (conversion) algorithm does not represent the worst-case input, its performance is often considerably better than the competitive ratio tells. For this reason competitive analysis is criticized as being too pessimistic. In terms of converting assets the com-
petitive ratio does not reveal which returns can be expected, nor whether these returns are positive or not. There is related work which conducts performance analysis assuming that input data is given according to a certain probability distribution. This approach is called ‘Bayesian Analysis’ [9, pp. 34-35]. The objective is to optimize the performance of an algorithm assuming a specific stochastic model [1]. Either assumptions about the distribution of the input data are made, or the distribution of the input data is assumed to be known beforehand [10]. However, this approach can often not be applied as distributions are rarely known precisely. Thus we will not make assumptions about input distributions or probabilities.

This leads to an exploratory approach. The algorithms are implemented, and the analysis is done on historic or artificial data by simulation runs. The objective of exploratory data analysis (EDA) is to 1) suggest hypotheses to test (statistically) based on the results generated, 2) assess assumptions on the statistical inference, 3) support the selection of appropriate statistical tools and techniques for further analysis, 4) provide a basis for further data collection through experiments. It is important to distinguish the EDA approach from the classical empirical approach, which starts with a-priori formulated hypotheses [11]. By applying EDA the observed empirical-case results are evaluated statistically, mainly by hypothesis tests, bootstrap procedures, or Monte Carlo simulation, cf. [12, 13, 14].

We apply the experimental approach (EDA) as well as competitive analysis, considering a worst-case and an empirical-case point of view, and limit to uni-directional preemptive algorithms introduced by [2, 3]. The investigated online conversion algorithms are based on the assumption that there exists a threat that at some stage during the time interval, namely on day $k \leq T$, the offered price will drop to a minimum level $m$, and will remain there until the last day $T$. We assume a time series of prices $Q = q_1, q_2, \ldots, q_k, m, m, \ldots, m$ where $t = 1, \ldots, k \leq T$. The algorithms proposed are commonly referred to as the threat-based, and the basic rules are [3, p. 109]:

**Algorithm 1.**

*Rule 1.* Consider a conversion from asset $D$ into asset $Y$ only if the price offered is the highest seen so far.

*Rule 2.* Whenever you convert asset $D$ into asset $Y$, convert just enough $D$ to ensure that a competitive ratio $c$ would be obtained if an adversary dropped the price to the minimum possible price $m$, and kept it there afterwards.
Rule 3. On the last trading day $T$, all remaining $D$ must be converted into $Y$, possibly at the minimum price $m$.

[2, 3] discussed four variants of the above algorithm, each assuming a different information about the future. [4] improved the algorithm by improving the lower bound given in [2, 3]. It is shown that the lower bound (the threat) equals $c \cdot m \geq m$ (where $c \geq 1$ is the competitive ratio), and not $m$ as assumed in [2, 3]. The basic rules of the five variants of the threat-based algorithm remain the same. The variants differ in how the amount to be converted $s_t$ is computed, and $s_t$ is dependent on the worst-case competitive ratio $c$ the algorithms are approaching. This leads to the following research question to be answered:

Can better competitive ratios be explained by higher quality of a-priori information about the future used in online conversion algorithms?

When answering this question we do not refer to the work of [5, 6, 7, 8]. These uni-directional preemptive conversion algorithms are not threat-based, and thus not comparable on a mutual basis. In addition, the authors make assumptions which do not hold in most practical settings. [5] assume a price function $g(q)$.

The constants $A$ and $B$ ($A, B \geq 1$) determine the prices offered on day $t$, and $q_t$ is modelled as $q_{t-1}/B \leq q_t \leq A \cdot q_{t-1}$. Further, [7] assume that the upper bound of the prices $M$ is a decreasing function of time and modelled by $M_t = M/t$. The algorithm by [8] requires specifying the maximum number of preemptions.

Our aim in this paper is twofold. First, we want to experimentally evaluate the performance of the uni-directional preemptive threat-based algorithms suggested by [2, 3, 4]. Then we apply EDA as well as competitive analysis considering a worst-case and an empirical-case point of view. In related work it is shown that the analytic worst-case competitive ratio $c_{wc}$ is the better the better the quality of the information about the future is.

We will investigate if this also holds for the empirical-case competitive ratio $c_{ec}$. The better the competitive ratio, the better should be the quality of information. This presumption is to be evaluated through experiments.

The reminder of this paper is organized as follows. In the next section the problem is formulated, and the algorithms considered are presented in detail. Section 3 presents the experimental design as well as the experimental findings from our simulation runs. We finish with some conclusions and suggestions for future research in the last section.
2. Problem Formulation

Each threat-based algorithm ON considered converts asset D to asset Y according to the rules given in Algorithm 1. Algorithm ON obtains price quotations \( q_t \) \((m \leq q_t \leq M, \text{ and } 0 < m < M)\) at points of time \( t = 1, \ldots, T \). For each price \( q_t \) ON calculates the amount to be converted \( s_t \in [0, 1] \) according to Rules 1 to 3. Remaining open positions must be converted at the latest on the last possible price \( q_T \), which might be the worst-case.

Let us consider the multiple conversion problem, i.e. we want to convert asset D more than once. As we consider uni-directional search, to convert asset D \( p \geq 1 \) \((i = 1, \ldots, p)\) times, the investment horizon must be divided into time intervals of length \( T \) days. As in [15] we assume two consecutive time intervals of equal length \( T \) are pooled, resulting in trading periods of length \( 2 \cdot T \). Within the first \( T \) days min-search is carried out in order to buy at possibly low prices, and within the second \( T \) days max-search is carried out in order to sell at possibly high prices. With this setting we ensure that each \( i \)-th trade consists of exactly one complete buying and one complete selling. Buying (selling) is complete as soon as the whole amount of D is converted, i.e. \( \sum_{t=1}^{T} s_t = 1 \). At the beginning of each time interval of length \( T \) days let \( d_0 = 1 \) be the amount of asset D remaining, and let \( y_0 = 0 \) the amount of already accumulated asset Y. Let \( d_t \) be the amount of asset D remaining after day \( t \), and \( y_t \) be the amount of asset Y accumulated after day \( t \). For \( t = 1, \ldots, T \) the amount of asset D remaining equals \( d_t = d_{t-1} - s_t \) and the accumulated amount of asset Y equals \( y_t = s_t \cdot q_t + y_{t-1} \).

In the following we present the five variants of the threat-based algorithm suggested by [2, 3, 4]. Based on the assumed a-priori information about the future, each algorithm determines \( s_t \) such that \( c \) holds in case the price drops to \( m \), i.e. the worst-case occurs.

2.1. Algorithm: Threat\( (m, M, k) \)

[2, 3] addressed the scenario where the player knows the upper and lower bounds of prices, \( m \) and \( M \), as well as the number of days \( k \leq T \). Rules 1 to 3 of Algorithm 1 ensure that at time \( t \), ‘just enough’ of asset D is converted.
that Threat\((m, M, k)\) achieves a competitive ratio \(c\). Thus

\[
\frac{OPT}{ON} = \frac{q_t}{y_t + m \cdot d_t} = \frac{q_t}{(y_{t-1} + s_t \cdot q_t) + m \cdot (d_{t-1} - s_t)} \leq c.
\]

The denominator \(y_t + m \cdot d_t\) represents the return of \(ON\) if an adversary drops the price to \(m\) and the nominator \(q_t\) is the return of \(OPT\) for this case, as \(q_t\) is the maximum and \(OPT\) will convert the whole asset \(D\) at price \(q_t\). According to Rule 3 \(ON\) must spend the minimum \(s_t\) that satisfies eq (2). Solving eq (2) as an equality constraint with respect to \(s_t\) results in eq (3). Thus, we get

\[
s_t = \frac{q_t - c \cdot (y_{t-1} + d_{t-1} \cdot m)}{c \cdot (q_t - m)}.
\]

It remains to determine the global competitive ratio \(c\) used in eq (3) that is attainable by algorithm Threat\((m, M, k)\). For every day \(t\) let \(k' = k - t + 1\) be the remaining days of the time series considered. Let \(q'_1\) be the first price of this time series. Let \(c^k(q'_1)\) be a local (lower bound) competitive ratio which is achievable on a sequence of \(k' \leq T\) remaining prices assuming \(d_t = 1\) and \(y_t = 0\) [3, Formula 15]

\[
c^k(q'_1) = 1 + \frac{q'_1 - m}{q'_1} \cdot (k' - 1) \cdot \left(1 - \left(\frac{q'_1 - m}{M - m}\right)^{(1/(k' - 1))}\right).
\]

Let \(c\) be a global (upper bound) competitive ratio assuming that \(q'_1\) is the highest price of the whole time series, i.e. \(OPT\) converts the whole amount of asset \(D\) to asset \(Y\) at price \(q'_1\), and \(ON\) converts the remaining amount of asset \(D\) to asset \(Y\). Thus [3, Formula 28a]

\[
c = \frac{q'_1}{d_{t-1} \cdot q'_1 + y_{t-1}} \cdot c^k(q'_1)
\]

The denominator \(d_{t-1} \cdot q'_1 + y_{t-1}\) represents the return of \(ON\), and the nominator \(q'_1\) is the return of \(OPT\). We now have to calculate which worst-case competitive ratio we could reach taking into account the following cases:

1. \(q'_1\) is a global maximum and \(OPT\) will convert the whole of asset \(D\) at price \(q'_1 = M\). Then from eq (5) the worst-case competitive ratio equals \(c(m, M, k) = c^k(q'_1)\) with \(q'_1 = q_1 = M\).
2. \( q'_1 \) is not a global maximum and \( OPT \) will convert the whole of asset \( D \) at a future rate. Then from eq (4) we get
\[
c(m, M, k) = \max \{ c^k(q'_1) | k' = 1, \ldots, k \leq T \} = c^T(q'_1).
\]
Note that when experiments are carried out the empirical-case competitive ratio \( c^{ec} \) of \( \text{Threat}(m, M, k) \) equals \( c^k(q'_1) \) where \( k' = 1 \).

2.2. Algorithm: \( \text{Threat}(m, M) \)

[2, 3] addressed the scenario where the player knows only the lower and upper bound, \( m \) and \( M \), of the offered prices. As the player is oblivious about the length of the time interval \( T \), it is assumed that the adversary selects \( T \rightarrow \infty \). In order to meet the ratio \( c \) the \( d_t \) must be determined such that the whole (remaining) amount of \( D \) is converted in case the highest possible price \( M \) occurs on day \( t \). From this follows that \( d_t \) equals [2, p. 4, Case 1]
\[
d_t = 1 - \frac{1}{c} \cdot \ln \frac{M - m}{m \cdot (c - 1)} \tag{6}
\]
with \( s_t = d_{t-1} - d_t \) and \( d_0 = 1 \). From eq (6) the worst-case competitive ratio, denoted by \( c^{\infty}(m, M) \), can be derived using \( c \cdot m \) as lower bound
\[
d_t &= 1 - \frac{1}{c} \cdot \ln \left( \frac{M - m}{c \cdot m - m} \right) \tag{7}
\]
\[
= 1 - \frac{1}{c} \cdot c
\]
\[
= 0.
\]
Thus \( c^{\infty}(m, M) \), is the unique solution \( c \) [3, Formula 29]
\[
c = \ln \frac{M}{m} - 1 \tag{8}
\]

2.3. Algorithm: \( \text{Threat}(m, M, q_1) \)

[2, 3] and [4] addressed the scenario where the player knows the lower and upper bound, \( m \) and \( M \), of the offered prices, as well as the first price \( q_1 \). For calculating the worst-case competitive ratio an arbitrary number of trading days \( T \rightarrow \infty \) must be considered. Thus the worst-case competitive ratio, denoted by \( c^{\infty}(m, M, q_1) \), is the unique solution of \( c \) [4, p. 29]
\[
c = \begin{cases} 
\ln \frac{M}{m} - 1 & q_1 \in [m, c \cdot m] \\
1 + \frac{q_1 - m}{q_1} \cdot \ln \frac{M - m}{q_1 - m} & q_1 \in [c \cdot m, M]. 
\end{cases} \tag{9}
\]
Further, depending on the value of $q_1$ the amount of $D$ remaining $d_t$ equals

\[ dt = \begin{cases} 1 - \frac{1}{c} \cdot \ln \frac{m - m}{q_1 - q_1/(c/m)} & q_1 \in [m, c \cdot m] \\ \frac{1}{c} \cdot \ln \frac{m - m}{q_1 - m} & q_1 \in [c \cdot m, M] \end{cases} \]  

(10)

with $s_t = d_{t-1} - d_t$ and $d_0 = 1$.

2.4. Algorithm: Threat($\phi$)

[2, 3] addressed the scenario where the player knows only the price fluctuation ratio, $\phi = M/m$, of the offered prices, but the real bounds on $M$ and $m$ are unknown. As the player does not know $T$, the player assumes the adversary to choose $T \to \infty$. The worst-case competitive ratio, denoted by $c^\infty(\phi)$, is computed as follows. Let $c^\infty(\phi) = \lim_{T \to \infty} c_T(\phi)$, then

\[
\lim_{T \to \infty} \frac{(\phi - 1)^T}{(\phi^{T/(\phi - 1)} - 1)^{T-1}} = (\phi - 1) \exp \left( -\frac{\phi \ln \phi}{\phi - 1} \right).
\]

(11)

Therefore

\[
c^\infty(\phi) = \phi \left( 1 - (\phi - 1) \exp \left( -\frac{\phi \ln \phi}{\phi - 1} \right) \right).
\]

(12)

2.5. Algorithm: Threat($\phi, k$)

[2, 3] addressed the scenario where the player knows the price fluctuation ratio $\phi$ with the duration $k \leq T$. [3, p. 122] observed that the minimum price offered on day $t$ is at least $q_t/\phi$. Therefore, the worst-case competitive ratio, denoted by $c(\phi, k)$, can be derived as in the analysis of Algorithm Threat($m, M, k$) specializing to the case in which $m = q_t/\phi$, resulting in [3, p. 126, Theorem 6]

\[
c(\phi, k) = \phi \left( 1 - (\phi - 1)^k / (\phi^{k/(\phi - 1)} - 1)^{k-1} \right).
\]

(13)

It remains to compute the amount to be converted $s_t$ for the Algorithms Threat($\phi$) and Threat($\phi, k$). For both [2, 3] observed that the minimum
price offered on day $t$ is at least $q_t/\phi$. By replacing the minimum possible price $m$ by $q_t/\phi$ we get

$$y_t + d_t \frac{q_t}{\phi} = \frac{q_t}{c}$$

$$\Rightarrow d_t = \phi \left( \frac{1}{c} - \frac{y_t}{q_t} \right).$$

(14)

From eq (3), we get

$$s_t = \frac{q_t - c(y_{t-1} + d_{t-1} \cdot q_t/\phi)}{c(q_t - q_t/\phi)}.$$  

(15)

Where $c$ equals $c(\phi)$ for Algorithm Threat($\phi$), and $c(\phi, k)$ for Algorithm Threat($\phi, k$). In the following the results of our simulation runs are presented.

3. Results

In the following we present the assumed test design, the performance measure as well as the computational results.

3.1. Test Design

Our experiments are based on the Dax-30 index prices for the investment horizon 01-01-1998 to 12-31-2007. We excluded weekends and country-specific holidays resulting in overall 2543 trading days. To ensure an identical number of trades for all algorithms considered we divide the investment horizon into trading periods of length $2 \cdot T$ where $T \in \{130, 65, 33(32), 10, 5\}$ resulting in trading periods of length 260, 130, 65, 20 and 10 days. We assume asset $D$ to be cash and asset $Y$ to be Dax-30 index. Within each ‘first’ time interval of length $T$ uni-directional search is carried out in order to convert all cash into index, and within the ‘second’ $T$ days the index is converted back to cash. As the threat-based algorithms are allowed to convert in maximum $T$ fractions ($s_t \in [0, 1]$), this setting ensures that one trade is completed within each $2 \cdot T$ days. We assume that in each time interval for buying $b$ (selling $s$) of length $T$ there are precise estimates of the possible maximum prices $M_b(i)$ ($M_s(i)$), and the possible minimum prices $m_s(i)$ ($m_s(i)$). In our experiments we compare the following algorithms.
3.1.1. Algorithm OPT

OPT is an offline algorithm which achieves the best possible return within each trading period of length $2 \cdot T$. It is assumed that OPT knows all prices. OPT will buy at minimum prices $m_b(i) \geq m$, and will sell at maximum prices $M_s(i) \leq M$ within each $T$.

3.1.2. Algorithm Threat($x$)

Each variant $x \in \{(m, M, k), (m, M), (m, M, q_1), (\phi, k), (\phi)\}$ of the threat-based algorithm converts according to Rules 1 to 3 given in Algorithm 1. We assume the number of remaining trading days to be $T' = T - t + 1$. Each algorithm Threat($x$) calculates the achievable worst-case competitive ratio $c_{wc}$ for each time interval and converts the corresponding quantities such that this $c_{wc}$ would be realized in case the price drops to $m$. There might be as many buying (selling) transactions as there are days $T$ in each time interval.

3.1.3. Algorithm BH

BH buys the index on the first day $t = 1$ of each trading period and sells it $2 \cdot T$ days later. BH is used as a benchmark.

3.2. Performance Measurement

The following assumptions apply for algorithms 3.1.1 to 3.1.3.

1. There is an initial amount of cash greater zero.
2. Possible transaction prices are daily closing prices.
3. Transaction costs are not considered.
4. Interest rate on cash is assumed to be zero.

The empirical-case competitive ratio $c_{ec}$ of the above algorithms is derived by the return achieved. Let $r_i$ be the trading period returns, calculated by (accumulated) selling price divided by (accumulated) buying price for each $i$-th trade ($i = 1, \ldots, p$). Then the overall return $r(p)$ after the last trade equals

$$r(p) = \prod_{i=1}^{p} r_i.$$  \hspace{1cm} (16)
From eq (16) we get the annualized return

\[ R(y) = r(p) \left( \frac{1}{y} \right) \quad (17) \]

where \( y \) equals the number of years within the investment horizon considered. For the considered 10-year investment horizon the annualized return is calculated for \( y = 10 \), and tells us which return we could expect within one year.

We calculate the competitive ratios \( c \) of the considered conversion algorithms according to eq (1)

\[ c \geq \frac{OPT(I)}{ON(I)} \quad (18) \]

where \( ON \in \{ Threat(x), BH \} \).

Let \( c^{wc} \) be the worst-case competitive ratio, and let \( c^{ec} \) be the empirical-case competitive ratio. When calculating \( c^{wc} \) we assume algorithm \( ON \) is confronted with the worst possible sequence of prices \( i = 1, \ldots, p \) times, and derive the \( c^{wc} \) of each threat-based algorithm as given in Section 2 taking the data of the problem instance into account. To calculate \( c^{wc} \) for \( BH \) we assume \( BH \) buys \( i \) times at the maximum possible price \( M_i(b) \), and sells \( i \) times at the minimum possible price \( m_s(i) \). Thus \( c^{wc} \) of the \( BH \) algorithm equals \( \prod_i = 1^p (M_s(i) \cdot M_b(i)) / (m_s(i) \cdot m_b(i)) \) as shown in [16].

When calculating \( c^{ec} \) the return which actually was achieved by \( ON \) and \( OPT \) is used, thus \( c^{ec} \leq c^{wc} \).

### 3.3. Computational Results

In this section we present the numerical results achieved by the online conversion algorithms presented above. For each trading period of length \( 2 \cdot T \in \{260, 130, 65, 20, 10\} \) the algorithms \( ON \in \{ Threat(x), BH \} \) and \( OPT \) are run. As performance measure we consider the worst-case competitive ratio \( c^{wc} \), and the empirical-case competitive ratio \( c^{ec} \). Clearly, the algorithms \( ON \) cannot outperform the optimal offline algorithm \( OPT \). We carried out 35 simulation runs in order to find out how the following measures compare:

1. the worst-case competitive ratios \( c^{wc} \) taking the data of the problem instance into account, and

2. the empirical-case competitive ratios \( c^{ec} \) found in the experiments.

---

Results of Mohr and Schmidt (2010)
Table 1 to 7 present the computational results. We answer these questions conducting experiments using the LifeTrader system.

**Question 1:** How do the worst-case competitive ratios which could have been possible from the experimental data compare?

Answering this question we calculated the worst-case competitive ratios $c_{wc}$ based on the Dax-30 data. For each buying and selling period we determine $m_b(i)$ ($m_s(i)$) and $M_b(i)$ ($M_s(i)$) and calculate the possible worst-case ratios according to eq (18). The results are shown in Table 1. In case of BH the ratio $c_{wc}$ grows exponentially with $p$, i.e. the greater the number of trades, the worse BH gets. Column 2 to 6 give the worst-case ratios $c_{wc}$ for each algorithm and trading period length considered. As expected, the results are consistent with the analytical results by [2, 3, 4]. When comparing Threat($m, M$) and Threat($\phi, k$) knowing the exact upper and lower bounds, $m$ and $M$, is more valuable than knowing $\phi = M/m$ and $k \leq T$ as it leads to a better $c_{wc}$. Similarly, it is more valuable to know $k \leq T$ as $c_{wc}$ of Threat($m, M, k$) is better than $c_{wc}$ of Threat($m, M, q_1$). From this we conclude that some information is more valuable than other. We also conclude that the better $c_{wc}$ the more valuable the information is.

1LifeTrader is a software system for the evaluation of conversion algorithms, details can be found in [17].

<table>
<thead>
<tr>
<th>1998-2007: Worst-case ratio $c_{wc} = OPT/ON$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2 \cdot T$</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>$OPT/BH$</td>
</tr>
<tr>
<td>$OPT$/Threat($\phi$)</td>
</tr>
<tr>
<td>$OPT$/Threat($\phi, k$)</td>
</tr>
<tr>
<td>$OPT$/Threat($m, M$)</td>
</tr>
<tr>
<td>$OPT$/Threat($m, M, q_1$)</td>
</tr>
<tr>
<td>$OPT$/Threat($m, M, k$)</td>
</tr>
</tbody>
</table>

Table 1: Worst-case competitive ratios $c_{wc}$ for 1998 to 2007
Question 2: How do the empirical-case competitive ratios found in the experiments compare?

Answering this question we calculated the empirical-case competitive ratios \( c^{ec} \) taking the Dax-30 data into account, given in Table 2. When calculating the ratios \( c^{ec} \) the empirical-case return which actually was achieved by \( ON \) is compared to \( OPT \), where \( ON \in \{ Threat(x), BH \} \). The results which are not consistent with the worst-case results given in Table 1 are marked bold.

In three cases, for 20, 65 and 260 days, \( BH \) achieves a greater value of \( OPT \) than \( Threat(\phi) \), as \( BH \) achieves a better \( c^{ec} \). This is due to the time series considered, for example if price \( q_1 << q_T \) for several periods. Further, in two cases, for 65 and 260 days, \( Threat(\phi) \) achieves a better \( c^{ec} \) than \( Threat(\phi, k) \). Following Rule 1 both variants convert at identical prices \( q_t \). But within some periods \( i \) (due to luck) \( Threat(\phi) \) calculated a greater \( s_t \) and thus converts more at a higher price than \( Threat(\phi) \). Resulting in a higher accumulated amount of index after time \( T \). For example \( Threat(\phi) \) outperforms \( Threat(\phi, k) \) if the prices in the time series considered are decreasing. In contrast, the analytical worst-case competitive ratio \( c^{wc} \) is improved by knowing \( k \leq T \), as given in Table 1. This is also true for the case where \( Threat(m, M) \) achieves a better \( c^{ec} \) than \( Threat(m, M, k) \). From this we conclude that some information is more valuable than other.

Surprisingly, the best results are achieved by \( Threat(m, M, q_1) \), i.e. for all trading period lengths the maximum amount \( 1/c^{ec} \) of \( OPT \) can be achieved for the time series considered. Due to luck regarding the value of the first price \( q_1 \) the empirical-case ratio \( c^{ec} \) of \( Threat(m, M, q_1) \) is always better than

<table>
<thead>
<tr>
<th>( 2 \cdot T )</th>
<th>Trades ( p )</th>
<th>10 days</th>
<th>20 days</th>
<th>65 days</th>
<th>130 days</th>
<th>260 days</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT/BH</td>
<td>2.2104</td>
<td>1.9194</td>
<td>1.5081</td>
<td>1.4387</td>
<td>1.2586</td>
<td></td>
</tr>
<tr>
<td>OPT/Threat(\phi)</td>
<td>2.1574</td>
<td>1.9258</td>
<td>1.5208</td>
<td>1.4199</td>
<td>1.2850</td>
<td></td>
</tr>
<tr>
<td>OPT/Threat(\phi, k)</td>
<td>2.1378</td>
<td>1.9222</td>
<td>\textbf{1.5214}</td>
<td>1.4198</td>
<td>\textbf{1.2850}</td>
<td></td>
</tr>
<tr>
<td>OPT/Threat(m, M)</td>
<td>1.1613</td>
<td>1.2309</td>
<td>1.2125</td>
<td>1.18056</td>
<td>1.1244</td>
<td></td>
</tr>
<tr>
<td>OPT/Threat(m, M, q_1)</td>
<td>1.1606</td>
<td>1.2307</td>
<td>1.2122</td>
<td>1.1802</td>
<td>1.1239</td>
<td></td>
</tr>
<tr>
<td>OPT/Threat(m, M, k)</td>
<td>\textbf{1.2012}</td>
<td>\textbf{1.2459}</td>
<td>\textbf{1.2149}</td>
<td>\textbf{1.1809}</td>
<td>\textbf{1.1241}</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Empirical-case competitive ratios \( c^{ec} \) for 1998 to 2007
$c^{ee}$ of Threat($m, M, k$). In contrast, the analytical worst-case competitive ratio $c^{wcc}$ is improved by knowing $k \leq T$ instead of $q_1$, as given in Table 1.

**Question 3:** Are the answers to Question 2 significant?

In order to answer this question we use a student $t$-test to show significance of results. The $t$-test generates useful output if the sample size (number of period returns) is greater than 30 or the period returns are normal distributed. To test for the normality assumption of the $t$-test we use the Jarque-Bera ($JB$) test. The null hypothesis of $JB$ is that the period returns of each algorithm and each trading period length are normal distributed, i.e. for the six algorithms and five different period lengths we conducted 30 $JB$ tests. We found out that all period returns are normal distributed, or that the sample size is greater than 30.

Based on the empirical findings given in Table 2 the null hypothesis ($H_0$) to be rejected is:

The empirical-case competitive ratio of an algorithm $ON$ using more valuable information is greater or equal ($\geq$) to the empirical-case competitive ratio of an algorithm $ON$ using less valuable information.

Before running a $t$-test we check if the returns generated by the compared two algorithms ($t$-test samples) have equal variances or not. Depending on the results on the variances different $t$-test variants are used [12]. The sample sizes for each $t$-test refers to the number of returns generated from 01-01-1998 to 12-31-2007, i.e. for a trading period length of 10 days we have a sample of 254 returns, for trading period length 20 we have a sample of 127 returns, etc.

The $t$-test statistics given in Tables 3 to 7, and are calculated depending on the results of the normality test and the variance equality test for the algorithms. We use a significance level of 5%. Overall we conducted 15 $t$-tests for each trading period length (10, 20, 65, 130, 260 days), resulting in overall 75 statistical tests. The lower the $p$-value, the more 'significant' is the result of the $t$-test concerning the rejection of $H_0$. In case the $p$-values are greater than 5% the null hypothesis $H_0$ cannot be rejected. In case $H_0$ can be rejected the $p$-values are marked bold with $x \in \{(m, M, k), (m, M), (m, M, q_1), (\phi, k), (\phi)\}$.

Results show that for all trading period lengths the returns generated by 1) Threat($m, M$), 2) Threat($m, M, k$) and 3) Threat($m, M, q_1$) are significantly greater ($>$) than the returns by $BH$ and Threat($\phi$). Thus we conclude the
<table>
<thead>
<tr>
<th>1998-2007: p-values of Test 1 for $2 \cdot T = 10$ days</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_0$: Threat($x$)</td>
</tr>
<tr>
<td>$\geq$</td>
</tr>
<tr>
<td>Threat($\phi$)</td>
</tr>
<tr>
<td>Threat($\phi, k$)</td>
</tr>
<tr>
<td>Threat($m, M$)</td>
</tr>
<tr>
<td>Threat($m, M, k$)</td>
</tr>
<tr>
<td>Threat($m, M, q_1$)</td>
</tr>
</tbody>
</table>

Table 3: Student $t$-test results for 10 days from 1998 to 2007

<table>
<thead>
<tr>
<th>1998-2007: p-values of Test 2 for $2 \cdot T = 20$ days</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_0$: Threat($x$)</td>
</tr>
<tr>
<td>$\geq$</td>
</tr>
<tr>
<td>Threat($\phi$)</td>
</tr>
<tr>
<td>Threat($\phi, k$)</td>
</tr>
<tr>
<td>Threat($m, M$)</td>
</tr>
<tr>
<td>Threat($m, M, k$)</td>
</tr>
<tr>
<td>Threat($m, M, q_1$)</td>
</tr>
</tbody>
</table>

Table 4: Student $t$-test results for 20 days from 1998 to 2007

<table>
<thead>
<tr>
<th>1998-2007: p-values of Test 3 for $2 \cdot T = 65$ days</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_0$: Threat($x$)</td>
</tr>
<tr>
<td>$\geq$</td>
</tr>
<tr>
<td>Threat($\phi$)</td>
</tr>
<tr>
<td>Threat($\phi, k$)</td>
</tr>
<tr>
<td>Threat($m, M$)</td>
</tr>
<tr>
<td>Threat($m, M, k$)</td>
</tr>
<tr>
<td>Threat($m, M, q_1$)</td>
</tr>
</tbody>
</table>

Table 5: Student $t$-test results for 65 days from 1998 to 2007

higher the value of the information the significantly better the empirical-case competitive ratios are. But this is not true for $BH$ as the empirical-case competitive ratios of Threat($\phi, k$) are only significantly higher for 10 and 130 days, cf. column $BH$ in Tables 3 and 6.

When comparing the empirical-case competitive ratios of Threat($\phi, k$) and Threat($m, M$) we conclude knowing the real bounds on the prices is
1998-2007: \( p \)-values of Test 4 for \( 2 \cdot T = 130 \) days

\[
\begin{array}{cccccc}
H_0: \text{Threat}(x) & BH & \text{Threat} & \text{Threat} & \text{Threat} & \text{Threat} \\
\ge & (\phi) & (\phi, k) & (m, M) & (m, M, k) \\
\text{Threat}(\phi) & 3.71\% & - & - & - & - \\
\text{Threat}(\phi, k) & 3.69\% & 49.99\% & - & - & - \\
\text{Threat}(m, M) & 0.00\% & 0.09\% & 0.09\% & - & - \\
\text{Threat}(m, M, k) & 0.00\% & 0.09\% & 0.09\% & 50.28\% & - \\
\text{Threat}(m, M, q_1) & 0.00\% & 0.09\% & 0.09\% & 49.83\% & 49.55\% \\
\end{array}
\]

Table 6: Student \( t \)-test results for 130 days from 1998 to 2007

1998-2007: \( p \)-values of Test 5 for \( 2 \cdot T = 260 \) days

\[
\begin{array}{cccccc}
H_0: \text{Threat}(x) & BH & \text{Threat} & \text{Threat} & \text{Threat} & \text{Threat} \\
\ge & (\phi) & (\phi, k) & (m, M) & (m, M, k) \\
\text{Threat}(\phi) & 21.29\% & - & - & - & - \\
\text{Threat}(\phi, k) & 21.26\% & 49.97\% & - & - & - \\
\text{Threat}(m, M) & 0.53\% & 3.95\% & 3.96\% & - & - \\
\text{Threat}(m, M, k) & 0.54\% & 3.94\% & 3.95\% & 49.84\% & - \\
\text{Threat}(m, M, q_1) & 0.53\% & 3.92\% & 3.93\% & 49.75\% & 49.91\% \\
\end{array}
\]

Table 7: Student \( t \)-test results for 260 days from 1998 to 2007

more valuable as \( c^{ec} \) and \( c^{ec} \) are always significantly better for \( \text{Threat}(m, M) \).

When comparing \( \text{Threat}(m, M, q_1) \) to \( \text{Threat}(m, M, k) \) the empirical-case ratios \( c^{ec} \) of \( \text{Threat}(m, M, q_1) \) are not significantly better than those of \( \text{Threat}(m, M, k) \). From this we conclude that due to luck regarding the value of first price \( q_1 \) the \( c^{ec} \) of \( \text{Threat}(m, M, q_1) \) is better than the \( c^{ec} \) of \( \text{Threat}(m, M, k) \).

4. Conclusions

Due to Rules 1 to 3 of Algorithm 1 for all the five variants of the threat-based algorithm the prices considered for conversion are identical; but the calculation of \( s_t \) is different for the algorithms based on the information assumed to be known.

In order to answer the question how much it is worth to know the future in online conversion problems we have suggested to identify a strict order of the value of information using worst-case competitive ratios \( c^{ec} \). We have defined information to be more valuable if the worst-case competitive ratio can be improved by this information.
Taking the problem data into account we could identify a strict order on the value of information based on the worst-case ratios $c^{\text{wc}}$ (Table 1). For the empirical-case scenarios (Table 2) this was not possible. In contrast to the worst-case scenarios we could see here that the value of a-priori information is not as powerful as a ‘luckily’ behaving time series. We conclude that the value of information can only be estimated by worst-case scenarios.

We assumed the precise values for $m$, $M$, $\phi$, $q_1$ and $k \leq T$ to be known for calculating competitive ratios. This assumption might be to optimistic. An open question would be to weaken this assumption and considering errors in forecasts. Further it would be interesting to take transaction costs into account as in the worst-case a preemptive conversion algorithm converts at each price presented, i.e. at all $T$ prices.

References


References for Chapter 6


Chapter 7

Conclusions and Future Work

This chapter summarizes, in a nutshell, the answers to the research questions on the applicability of the investigated non-preemptive and preemptive online conversion algorithms. We conclude indicating some open questions for future research and give a selective bibliography.

7.1 Conclusions

In finance, the traditional approach when analyzing online conversion algorithms is to derive the return to be expected through experiments. The competitive ratio, giving a performance guarantee assuming worst-case scenarios, is not considered. Traditional empirical-case analysis assumes the input follows a particular distribution, and aims to analyze and optimize the empirical-case performance of an algorithm assuming a specific stochastic model. But in case an investor does not want to rely on a stochastic model, or it is unknown, the worst-case competitive analysis approach provides an attractive alternative to this traditional approach. Whatever the reason for the absence of information about stochastic processes is, worst-case competitive analysis offers a reasonable initial solution upon which a more elaborate online conversion algorithm can be chosen after additional information is determined. Empirical-case analysis provides this additional information.

The suggested conjoint approach provides bounds that minimize the maximum regret based on worst-case scenarios. In addition, the empirical-case results can be used to draw conclusions on the statistical inference of the return to be expected. The outcome is an answer to the research questions stated.

First, we stated the question ‘can the applicability of heuristic conversion algorithms be verified through competitive analysis, and which worst-case competitive ratio do they achieve?’ addressing the new field of worst-case analysis.
of heuristic conversion algorithms. From a conceptual point of view, a heuristic conversion algorithm that performs well in an experiment is not necessarily a 'good' online conversion algorithm. In contrast to the worst-case scenarios there is always the probability of a 'luckily' behaving time series. But in case of a stock market meltdown worst-case performance guarantees are essential, as they provide a definite upper bound. This is what competitive analysis offers. Thus it is reasonable to apply the analytical competitive analysis approach to heuristic conversion algorithms when analyzing their applicability.

Second, we stated the question 'can the applicability of guaranteeing conversion algorithms be verified through experiments, and which empirical-case performance do they achieve?' addressing the new field of empirical-case analysis of guaranteeing conversion algorithms. An algorithm that guarantees a small worst-case competitive ratio does not necessarily achieve a 'good' empirical-case performance. The assumptions made when the competitive ratio is derived analytically are often far from reality. Backtesting solves this problem by taking an algorithm, and going back in time in order to see what would have happened if the algorithm had been followed in practice. This is what empirical-case analysis offers. Thus it is essential to apply experimental analysis to guaranteeing conversion algorithms when analyzing their applicability.

Our experimental results provide support for utilizing the considered guaranteeing conversion algorithms Threat and Sqrt in practice. In case the data processed by those algorithms does not represent the worst-case input the return to be expected is significantly better than the worst-case competitive ratio tells. Results show that the five threat-based algorithms Threat($X$) with $X \in \{(m, M, k), (m, M), (m, M, q_1), (\varphi, k), (\varphi)\}$ clearly outperform constant rebalancing as well as classical buy-and-hold. To reduce the number of conversions the non-preemptive algorithm Sqrt is a good alternative to the preemptive threat-based algorithms as Sqrt also outperforms buy-and-hold. For example if we want to reduce transaction costs. The results could be confirmed statistically.

In contrast, the worst-case competitive ratio of the considered heuristic conversion algorithms MA and TRB does not provide support for utilizing these algorithms in practice. The worst-case competitive ratio equals $\left(\frac{M}{m}\right)^{2p}$, as we found the worst-case return of $ON \in \{MA, TRB\}$ to be $\frac{m}{M}$. Even worse, the worst-case ratio grows exponential with $p$, where $f(x) = x^{2p}$ and $x = \frac{M}{m}$. The greater $p$ and/or the $\frac{M}{m}$-ratio get, the greater is the worst-case competitive ratio.

We conclude that an online conversion algorithm should only be chosen for practical application in case both measures, its competitive ratio and the return to be expected, are promising.

Besides answering the general question on (how to measure) the quality of
Future Work

an online conversion algorithm, as addressed in the works given in Chapter 6, several related questions can be answered by computing both the empirical-case performance as well as the worst-case performance. We compare an online conversion algorithm to the optimal offline algorithm. In this way, we get a measure of the return obtained by \( ON \) compared to the optimal return that could have been obtained if we had known all future prices. This can be thought of as measuring the value of the information of future prices. We answered the question ‘how much is it worth to know the future in online conversion problems?’ addressing the value of the assumed a-priori knowledge of different online conversion algorithms. We conclude that the value of information can only be estimated by worst-case scenarios, and define information to be more valuable if the worst-case competitive ratio can be improved by this information.

In the following we give different directions for future work. We suggest to answer these open questions using the worst-case as well as the empirical-case competitive ratio, and the return to be expected.

7.2 Future Work

When carrying out experiments, we assumed the precise values for \( m, M, \varphi, q_1 \) and \( k \leq T \) to be known for calculating the competitive ratios. This assumption might be too optimistic. A first open question would be to weaken this assumption, and to consider forecasts to estimate these values.

Al-Binali (1999) suggests the risk-reward competitive analysis approach which contains two approaches. The first approach is to allow an online conversion algorithm to benefit from the investors capability in correctly forecasting the future sequence(s) of prices. The second approach is to allow the investor to control the risk by selecting ‘near optimal’ algorithms subject to the personal risk tolerance. The result are online conversion algorithms with a bounded loss within a pre-specified risk tolerance. An open question is to analyze the applicability of the risk-reward approach in practice.

It would be favorable to ensure that a forecast is correct with a certain probability. An open question is whether the solution of the secretary problem can be exploited to calculate this probability. The solution is to observe the first \( T/e \) values, and then to accept the first value which is better than all the previous ones. For \( T \to \infty \), the probability of selecting the best value then goes to \( 1/e \), which is around 37% (Babaioff et al., 2008). An open question is to exploit this solution, and to analyze whether estimates for \( m, M, \varphi, q_1 \) and \( k \leq T \) are correct in about 37% of the cases in practice.
Further, when allowing forecasts on $m$, $M$, $\varphi$, $q_1$ and $k \leq T$, these values might be under- or overestimated. A related open question is how these errors in forecasts influence the performance of an online conversion algorithm. In would be of interest to find an algorithm that takes advantage of the forecasts when they are accurate, while at the same time maintaining a good worst-case competitive ratio in case they are incorrect (Mahdian et al., 2007, p. 288).

In case worst-case competitive analysis is applied, this leads to the development of online conversion algorithms with minimum relative performance risk. This property is favorable for risk-averse investors who prefer an inferior but guaranteed performance to a better but uncertain expected performance. The second approach suggested by al-Binali (1999) allows to control risk, not to avoid it. An investor has the possibility to take (or even increase) risk for some form of (higher) reward. On open question is to introduce risk levels an investor is willing to take, and to develop ‘optimal’ online conversion algorithms incorporating these levels (Iwama and Yonezawa, 1999). Further, the competitive ratio of an online conversion algorithm measures the return and the incorporated risk within a single number – the ratio $c$. When allowing a risk control mechanism based on the competitive ratio as suggested by al-Binali (1999), an open question is whether the competitive ratio is an appropriate measure of risk measure or not. Artzner et al. (1999) introduce coherent measures of risk. A set of four desirable properties are presented and justified; risk measures satisfying these properties are called ‘coherent’. It is to be shown whether the competitive ratio is ‘coherent’ or not.

When considering worst-case scenarios to derive a $c^{wc}$ an arbitrary volatility of the worst-case time series $Q$ is assumed. An open question is whether the worst-case competitive ratio can be improved by replacing ‘unrealistic’ worst-case scenarios. Considering the data history more realistic worst-case sequences of prices could be assumed taking a bounded volatility into account (Hu et al., 2005, p. 229).

In case an online conversion algorithm is considered for practical application it would be of interest to determine and analyze its empirical-case competitive ratio $c^{ec}$ assuming proper input distributions. Fujiwara et al. (2011) state the question ‘when it comes to average-case evaluation with an input distribution, what is an adequate measure?’ and suggest average-case competitive analysis: The competitive ratio of an online conversion algorithm is determined while making various assumptions on the underlying price processes. An open question is to analyze the presented online conversion algorithms assuming different input distributions. Further, empirical results show that price movements between two stocks are bounded in some markets (Zhang et al., 2010, p. 2). The considered online conversion algorithms assume that prices are bounded within an interval, for example $q_t \in [m, M]$ (El-Yaniv et al., 2001, p. 107). It would be of interest
to evaluate the performance of these algorithms assuming the prices itself are interrelated, for example by assuming that a price depends on its preceding price.

El-Yaniv et al. (1992, 2001) have shown the uni-directional threat-based algorithm to be optimal. But the suggested bi-directional algorithm, which repeats the uni-directional algorithm, is not (Dannoura and Sakurai, 1998, p. 28). Therefore, the problem of designing an optimal threat-based algorithm for bi-directional search remains unanswered so far. Moreover, it would be interesting to take transaction costs into account as in the worst-case a threat-based algorithm converts at each of the $T$ prices presented.

The outcome of any online conversion algorithm are buy and sell signals. As an order, these signals can be executed on the stock market. Before submitting an order it might be of interest that the signals produced are correct – in the sense that they are ‘bug-free’. Certifying algorithms solve this problem. With each output they produce a certificate or witness (easy-to-verify proof) that the particular output has not been compromised by a bug (Mehlhorn and Schweitzer, 2010). An open question is to apply this approach to online conversion algorithms.

References for Chapter 7


References

Chapter 1


Ignatovich, D.: 2006, Quantitative trading system, Technical Report HN-06-14 (honors), The University of Texas at Austin, Department of Computer Sciences.


Chapter 2


Chapter 3


Welch, B.: 1947, The generalization of ‘student’s’ problem when several different population variances are involved, Biometrika 34(1/2), 28–35.


Chapter 4


**Chapter 5**


**Chapter 6**


**Chapter 7**


Appendix A

Copyright Permissions

A.1 Springer, Heidelberg

Esther Mohr

Von: Schmidt-Loeffler, Barbara. Springer DE [Barbara.Schmidt-Loeffler@springer.com]
An: em@itm.uni-sb.de
Betreff: VG: Question from a Springer Journal Contributor

Empirical Analysis of an Online Algorithm for Multiple Trading Problems Esther Mohr and Günter Schmidt
DOI: 10.1007/978-3-540-87477-5_32

Trading in Financial Markets with Online Algorithms Esther Mohr and Günter Schmidt
DOI: 10.1007/978-3-642-00142-0_6

Sehr geehrte Frau Mohr,

Unter den folgenden Voraussetzung erteilen wir Ihnen hiermit die kostenlose Genehmigung, das angefragte Material zu übernehmen:

- es handelt sich um original in unserem Werk erschienenes Material, d.h. dass bei der Abbildung kein Hinweis auf eine andere Quelle vorhanden ist,
- dass die Originalquelle genau zitiert wird, einschließlich Copyrightvermerk des Springer-Verlages.

Das Material darf lediglich zum Zweck Ihrer Promotion mit max. 30 Kopien (Papier) benutzt werden.

Es ist Ihnen selbstverständlich gestattet, die Manuskriptversion Ihres Artikels als Teil Ihrer Dissertation zu verwenden und diese als PDF Datei auf Ihrer, bzw. der Homepage der Universität zu veröffentlichen (auch frei zugänglich). Wir gehen davon aus, dass die Originalveröffentlichung genau zitiert wird, und ein Link zur Springer Website eingefügt wird. Der Link sollte den folgenden Hinweis bekommen "Originalveröffentlichung erschienen auf www.springerlink.com".

Mit freundlichen Grüßen,

______________________________
Barbara Schmidt-Löffler
Springer
Rights and Permissions
______________________________
Tiergartenstr. 17 | 69121 Heidelberg | Germany barbara.schmidt-loeffler@springer.com
www.springeronline.com
A.2 Elsevier B.V.

As a journal author, you retain rights for large number of author uses, including use by your employing institute or company. These rights are retained and permitted without the need to obtain specific permission from Elsevier. These include:

- the right to make copies (print or electric) of the journal article for their own personal use, including for their own classroom teaching use;

- the right to make copies and distribute copies (including via e-mail) of the journal article to research colleagues, for personal use by such colleagues;

- the right to post a pre-print version of the journal article on Internet web sites including electronic pre-print servers, and to retain indefinitely such version on such servers or sites;

- indefinitely such version on such servers or sites;

- the right to post a revised personal version of the text of the final journal article (to reflect changes made in the peer review process) on the author’s personal or institutional web site or server, incorporating the complete citation and with a link to the Digital Object Identifier (DOI) of the article;

- the right to present the journal article at a meeting or conference and to distribute copies of such paper or article to the delegates attending the meeting;

- for the author’s employer, if the journal article is a ‘work for hire’, made within the scope of the author’s employment, the right to use all or part of the information in (any version of) the journal article for other intra-company use (e.g. training), including by posting the article on secure, internal corporate intranets;

- patent and trademark rights and rights to any process or procedure described in the journal article;

- the right to include the journal article, in full or in part, in a thesis or dissertation;

- the right to use the journal article or any part thereof in a printed compilation of works of the author, such as collected writings or lecture notes (subsequent to publication of the article in the journal); and

---

See also [http://www.elsevier.com/wps/find/authorsview.authors/copyright#whatrights](http://www.elsevier.com/wps/find/authorsview.authors/copyright#whatrights)
the right to prepare other derivative works, to extend the journal article into book-length form, or to otherwise re-use portions or excerpts in other works, with full acknowledgment of its original publication in the journal.

Elsevier Global Rights Department
phone (+44) 1865 843 830
fax (+44) 1865 853 333
e-mail: permissions@elsevier.com

Other uses by authors should be authorized by Elsevier through the Global Rights Department, and journal authors are encouraged to let Elsevier know of any particular needs or requirements.
Appendix B

Eidesstattliche Versicherung


Saarbrücken, 10.03.2011

(Unterschrift)