# Can online trading algorithms beat the market? An experimental evaluation

**Javeria Iqbal[1], Iftikhar Ahmad[1], and Günter Schmidt[1,2]**

1 **Chair of Information and Technology Management**
**University of Saarland, Germany**
2 **Department of Statistical Sciences**
**University of Cape Town, South Africa**
{ji,ia,gs}@itm.uni-sb.de

**Abstract**

From experimental evaluation, we reasonably infer that online trading algorithms can beat the market. We consider the scenario of trading in financial market and present an extensive experimental study to answer the question "Can online trading algorithms beat the market?". We evaluate the selected set of online trading algorithms on $DAX30$ and measure the performance against buy-and-hold strategy. In order to compute the experimentally achieved competitive ratio, we also compare the set of algorithms against an optimum offline algorithm. To add further dimensionality into experimental setup, we use trading periods of various lengths and apply a number of evaluation criteria (such as annualized geometric returns, average period returns and experimentally achieved competitive ratio) to measure the performance of algorithms in short vs. long term investment decisions. We highlight the best and worst performing algorithms and discuss the possible reasons for the performance behavior of algorithms.

**1998 ACM Subject Classification** F.2 Analysis of Algorithms and Problem Complexity

**Keywords and phrases** Online Algorithms, Experimental Evaluation, Competitive Analysis

**Digital Object Identifier** 10.4230/OASIcs.SCOR.2012.43

## 1 Introduction

The major concern of an investor (both individual and corporate) in financial market is the profitability of the underlying trading strategy. As financial markets are highly volatile and risky, an investor wants to ensure that the trading strategy that he/she relies on, is profitable and provides better returns in unforeseen circumstances. Performance evaluation of the trading strategies is required to ensure that the selected strategies are thoroughly tested in real world scenario. Trading in financial markets is based on the principle of maximizing the difference between selling and buying, i.e., buying at minimum possible price and selling at maximum possible price. However, as the investor (henceforth called as player) cannot see the future prices, his decision is based on limited knowledge.

Online algorithms can be used to support trading decisions in financial markets where complete knowledge about future is not available. Online algorithms are based on the paradigm that the player has no knowledge about future and every decision the algorithm makes is based on the current knowledge of the algorithm. Online algorithms are evaluated through competitive analysis paradigm. Competitive analysis is used to measure the performance of an online algorithm against an optimum offline algorithm. Let $\mathcal{P}$ be a maximization problem and $\mathcal{I}$ be the set of all input instances, $ON$ be an online algorithm for problem $\mathcal{P}$

3rd Student Conference on Operational Research (SCOR 2012).
Editors: Stefan Ravizza and Penny Holborn; pp. 43–52

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and $ON(I)$ be the performance of algorithm $ON$ on input instance $I \in \mathcal{I}$. Let $OPT$ be an optimum offline algorithm for the same problem $\mathcal{P}$, $ON$ is said to be *c-competitive* if $\forall I \in \mathcal{I}$

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

We consider a set of online algorithms for trading and evaluate them from empirical perspective by executing the set of algorithms on real world. We evaluate the performance of algorithms using different lengths of trading periods (10, 20, 60, 130 and 260 days) and use a variety of evaluation measures such as annualized geometric returns ($AGR$), average period return ($APR$) and experimentally achieved competitive ratio ($c_e$) to evaluate the selected set of online algorithms.

**Definitions:** We present a set of basic terms and definitions with reference to online trading algorithms.

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 trading periods in an investment horizon is represented by $P$.
iii. *Investment Horizon*: The total time duration in which all transactions are carried out. The investment horizon can be divided into one or more time segments for trading.
iv. *Duration (T)*: The length of a trading period.
v. *Offered Price ($q_t$)*: The offered price on day $t$.
vi. *Upper Bound (M)*: The upper bound on possible offered prices during the trading period.
vii. *Lower Bound (m)*: The lower bound on possible offered prices during the trading period.
viii. *Amount Converted ($s_t$)*: Specifies which fraction of the amount available (e.g. wealth) is to be converted at price $q_t$ on day $t$, with $0 \leq s_t \leq 1$.

## 2 Literature Review

In this section, we briefly describe different performance evaluation techniques for online algorithms and provide a short review of the literature in relation to experimental analysis of different trading strategies.

The performance of online algorithms are analyzed from three different perspectives, *Bayesian Analysis* assumes that the input instance is drawn from a known probability distribution and expected performance of the algorithm is investigated on the assumed probability distribution, *Competitive Analysis* compares the performance of online algorithms against that of optimum offline algorithm, and *Experimental Analysis* uses the back testing technique to evaluate the performance of online algorithm on real world (as well as synthetic) data. Each of these techniques has its own limitations and drawbacks. The Bayesian analysis is considered to be too optimistic and relies heavily on underlying distribution. The competitive analysis is criticized to be too pessimistic as online algorithm is compared with an optimum offline algorithm which has complete knowledge about the future, thus experimental analysis can be used as a useful tool in conjunction with worst case competitive analysis to evaluate the performance of online algorithms and to measure the disparity between theoretical and real world performance of online algorithms.

The experimental analysis of online trading algorithms is not widely addressed in the literature. Mohr and Schmidt [7] presented an empirical study where the reservation price algorithm of El-Yaniv et al. [2] is compared with buy-and-hold (BH). Schmidt et al. [9] evaluated the reservation price algorithm and threat based algorithm [2] to dollar average

strategy (DAS) and BH. However, there is no such study which considers a broader set of online algorithms for trading and evaluate them extensively on real world dataset.

The experimental evaluation of heuristics trading algorithms has received considerable attention from researchers. In these studies BH is used as a benchmark for comparing different strategies. Due to space constraint the details cannot be included and the reader is referred to [1, 4, 6, 8, 10, 11].

# 3 Online Trading Algorithms

We briefly discuss the set of online trading algorithms. The proof of competitive ratio and optimality of algorithm is left out due to space constraint and the reader is referred to corresponding research papers. The online trading algorithms are broadly classified as *Non Preemptive* and *Preemptive* algorithms.

## 3.1 Non Preemptive Algorithms

In non preemptive algorithms (also referred to as reservation price algorithms), the player invests (buys or sells) his whole wealth at one point of time in investment horizon. The player computes a reservation price and compares each offered price with reservation price. A reservation price is the maximum(minimum) price which a player will accept for buying (selling). The player takes a buy decision and invests at the offered price if it is less or equal than the pre-computed reservation price, and a sell decision is taken when the offered price is greater (or equal) than the pre-computed reservation price. We discuss algorithms presented by El-Yaniv et al. [2] and Kao and Tate [3] and BH strategy.

### 3.1.1 El-Yaniv et al. reservation price algorithm

El-Yaniv et al. [2] presented reservation price algorithm based on the assumption that the lower and upper bound of offered price $m$ and $M$ are known to the player. El-Yaniv et al. presented the strategy for max-search (sell) problem. Schmidt et al. [9] analogously extended the strategy for buy. The buy and sell strategies are given as following;

- **Algorithm 1** (*RPMm*)
  - Buy at the first price less than or equal to $q^* = \sqrt{M \cdot m}$.
  - Sell at the first price greater than or equal to $q^* = \sqrt{M \cdot m}$.

### 3.1.2 Kao and Tate online difference maximization approach

Kao and Tate [3] presented a non-preemptive strategy maximizing the difference between the rank of selected buy and sell items. Let $x_t$ be the rank of $q_t$ in the $t$ prices observed so far, Kao and Tate [3] described the buy and sell strategies as following;

- **Algorithm 2** (*KT*)
  - Buy at price $q_t$ if $x_t \leq \mathcal{L}_T(t)$.
  - Sell at price $q_t$ if $x_t \geq \mathcal{H}_T(t)$ .

Where $\mathcal{L}_T(t)$ and $\mathcal{H}_T(t)$ are the limits (reservation price) for buy and sell respectively and are calculated as following;

$$\mathcal{H}_T(t) = \left\lceil \frac{t+1}{T+1} \cdot R_T(t+1) \right\rceil . \tag{2}$$

$R_T(t)$ is the expected final rank of high selection (sell) if the optimal strategy is followed starting at the $t$-th time. $R_T(t)$ is calculated as following;

$$R_T(t) = \frac{\mathcal{H}_T(t) - 1}{t}\left(R_T(t+1) - \frac{T+1}{2(t+1)}\mathcal{H}_T(t)\right) + \frac{T+1}{2}. \tag{3}$$

$$\mathcal{L}_T(t) = \begin{cases} 0 & t = T, \\ \left\lfloor \frac{t+1}{T+1} \cdot (R_T(t+1) - P_T(t+1)) \right\rfloor & t < T. \end{cases} \tag{4}$$

$P_T(t)$ is the expected high-low (sell-buy) difference, following optimal strategy at step $t$, $P_T(t)$ is calculated as following;

$$P_T(t) = \begin{cases} 0 & t = T, \\ P_T(t+1) + \frac{\mathcal{L}_T(t)}{t}\left(R_T(t+1) - P_T(t+1) - \frac{T+1}{t+1} \cdot \frac{\mathcal{L}_T(t)+1}{2}\right) & t < T. \end{cases} \tag{5}$$

### 3.1.3 Buy-and-Hold

The buy-and-hold strategy is a long term investment policy which is used as a benchmark in financial markets for comparing the performance of other trading algorithms and strategies. The key idea is that financial markets returns are worthwhile in contempt of volatility and recession.

- **Algorithm 3** (*BH*)
  - Buy at the the first offered price $q_1$.
  - Sell at the the last offered price $q_T$.

## 3.2 Preemptive Algorithms

In preemptive algorithms, the player does not invest (buy and sell) at one point of time in investment horizon, instead the player invests a portion of wealth. The exact investment amount depends on the offered price and/or time of investment. We consider the preemptive algorithms of El-Yaniv et al. [2], Lorenz et al. [5] and dollar average strategy.

### 3.2.1 El-Yaniv et al. threat based algorithm

El-Yaniv et al. [2] strategy is based on the assumption that the adversary may drop the offered price to some minimum level $m$ and will keep it there for the rest of investment horizon, the strategy maximizes the performance of the algorithm while safeguarding itself against the assumed threat. The basic rules of the algorithm are as following;

- **Algorithm 4** (*YFKT*)
  1. Consider a conversion from asset $D$ into asset $Y$ only if the price offered is the highest (lowest for selling) seen so far.
  2. Whenever convert asset $D$ into asset $Y$, convert just enough $D$ to ensure that a competitive ratio $c$ would be obtained if an adversary drops the price to the minimum possible price $m$, and keeps it there afterwards.
  3. On the last day $T$, all remaining $D$ is converted into $Y$, possibly at price $m$.

El-Yaniv et al. [2] proposed different variants of Algorithm 3.2.1, each assuming different a-priori information. We consider the variant of Algorithm 3.2.1 where the player has knowledge about lower and upper bounds ($m$ and $M$) of offered prices.

### 3.2.2 Lorenz et al. algorithm

Lorenz et al. [5] proposed a strategy where the player has the information about the lower and upper bounds of offered prices. Two different strategies were proposed, one each for buying and selling.

- **Algorithm 5** (*LPS*)
  - Selling (Max-search) Problem: At the start of the game the player computes reservation prices $q_i^* = (q_1^*, q_2^*, ...q_u^*)$, where $i = 1, .., u$. As the prices are observed by the player, he 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 is still some wealth left on day $T$, it must be sold at the last offered price, which may be at the lowest price $m$.

$$q_i^* = m\left[1 + (c-1)\left(1 + \frac{c}{u}\right)^{i-1}\right]. \tag{6}$$

    Where $c$ is the competitive ratio for the max-search (selling) problem.
  - Buying (Min-search) Problem: Follows the same procedure as for max-search problem, the reservation prices are computed as follows;

$$q_i^* = M\left[1 - \left(1 - \frac{1}{c}\right)\left(1 + \frac{1}{u \cdot c}\right)^{i-1}\right]. \tag{7}$$

    Where $c$ is the competitive ratio for the min-search (buying) problem.

### 3.2.3 Dollar average strategy (*DAS*)

The dollar average strategy is based on fixed investments on predefined time intervals. We apply DAS to invest equal amount of wealth on each day of the investment horizon. Let $L = T/2$ be the length of sell (buy) period, we invest $1/L$ of the total wealth on each day.

## 4 Experimental Evaluation

We discuss the basic experimental setup, dataset, assumptions and results as follows;

### 4.1 Settings

We discuss the datasets utilized for performance analysis, the assumptions considered for the trading algorithms and methodology adapted for evaluation process.

**Dataset:** We consider real world dataset of $DAX30$ (Jan 1st 2001 - Dec 31st 2010) for experimental evaluation.

**Assumptions:** We consider the following set of assumptions.

1. The initial wealth assumed is always 1 unit.
2. All prices are daily closing prices.
3. In Frankfurt stock exchange, the transaction cost is calculated as $\min\{\max\{0.60, 0.0048\%\}, 18\}$, i.e 0.0048% of the market value (minimum of 60 cents) upto maximum of 18 euros. Thus, the transaction costs are highly dependent on amount transacted, but for the sake of simplicity and other factors such as liquidity (which is limited in stock market) we consider a transaction cost of 0.0048% of volume traded.

4. The length of trading period is T $\in$ {10,20,60,130,260} days.
5. The interest rate considered is zero as wealth (money) deposited in broker account is interest free.
6. We assume that the algorithm knows the a-priori information such as $m$ and $M$ etc, as this is required for the proper working of algorithm.

**Evaluation Criteria:** We use annualized geometric returns ($AGR$), average trading period return ($APR$), experimentally achieved competitive ratio ($c_e$), and average number of transactions ($Tx$) as criteria to evaluate the performance of trading algorithms. The different measures provide a deeper insight on profitability of an algorithms in short and long runs. We discuss the number of transactions performed by each algorithm to discuss the potential impact of transaction cost on performance of algorithm.
Let $D_i$ and $d_i$ be the amount of wealth at the start and end of trading period $i$. Return of the trading period $i$, is given as;

$$r_i = d_i/D_i. \tag{8}$$

Geometric return is based on the assumption that the wealth at hand at start of the period $i$ is invested in the next period $i+1$. Geometric returns can be used to evaluate the annualized performance that algorithm achieves in investment horizon. Let $P$ be the number of trading periods in an investment horizon, if the number of years in the investment horizon, $y \geq 1$, then $AGR$ is calculated as;

$$AGR(P) = \left(\prod_{i=1}^{P} r_i\right)^{1/y}. \tag{9}$$

The average period return ($APR$) is used for performance evaluation of algorithms, where we assume trading periods of same length and averages the results over all trading periods of same length. $APR$ reflects the expected average performance within a trading period of given length.

$$APR(P) = \left(\prod_{i=1}^{P} r_i\right)^{1/P}. \tag{10}$$

Although, $AGR$ and $APR$ provides a useful insight about the profitability of an algorithm, it does not tell the whole story, as it is a stand alone measure and fails to measure the performance of an algorithm against the optimum possible result. Experimentally achieved competitive ratio ($c_e$) measures the performance of an algorithm against that of the optimal algorithm and can be use in conjunction with $AGR$ and $APR$ to report the profitability of an algorithm.

## 4.2 Results

In the following, we present the performance evaluation of algorithms on $DAX30$. We analyze the average number of transactions as well.

**Performance evaluation:** We discuss annualized geometric returns ($AGR$), average period return ($APR$) and experimentally achieved competitive ratio ($c_e$).
Table 1 describes the $AGR$ of algorithms on $DAX30$ dataset for trading periods of different lengths. It can be observed that the performance of YFKT is the best whereas BH and

**Table 1** Annualized Geometric Return ($AGR$) over $DAX30$.

| Period | 10 | 20 | 60 | 130 | 260 |
|---|---|---|---|---|---|
| ***OPT*** | 1.6976 | 1.7317 | 1.5165 | 1.4180 | 1.2694 |
| ***RPMm*** | 1.3715 | 1.2336 | 1.1493 | 1.0982 | 1.0851 |
| ***KT*** | 0.7698 | 0.8599 | 0.9822 | 1.0261 | 1.0172 |
| ***BH*** | 0.7485 | 0.8654 | 0.9634 | 0.9729 | 0.9852 |
| ***YKFT*** | **1.4411** | **1.4112** | **1.2525** | **1.2091** | **1.1585** |
| ***LPS*** | 0.9733 | 1.0526 | 1.0654 | 1.0909 | 1.0319 |
| ***DAS*** | 0.7592 | 0.8729 | 0.9637 | 1.0068 | 0.9870 |

**Table 2** Average Period Return ($APR$) over $DAX30$.

| Period | 10 | 20 | 60 | 130 | 260 |
|---|---|---|---|---|---|
| ***OPT*** | 1.0206 | 1.0431 | 1.1097 | 1.1908 | 1.2694 |
| ***RPMm*** | 1.0122 | 1.0163 | 1.0354 | 1.0479 | 1.0851 |
| ***KT*** | 0.9900 | 0.9885 | 0.9955 | 1.0130 | 1.0172 |
| ***BH*** | 0.9889 | 0.9889 | 0.9907 | 0.9864 | 0.9852 |
| ***YKFT*** | **1.0142** | **1.0268** | **1.0579** | **1.0996** | **1.1585** |
| ***LPS*** | 0.9990 | 1.0040 | 1.0160 | 1.0445 | 1.0319 |
| ***DAS*** | 0.9895 | 0.9896 | 0.9908 | 1.0034 | 0.9870 |

DAS are least profitable strategies. The performance of RPMm is second only to YFKT. The performance of algorithms in term of "average period return" on DAX30 dataset is summarized in Table 2. The performance of threat based algorithm of El-Yaniv et al. [2] (YFKT) is the best among all online algorithms. YFKT performs consistently better over trading periods of different lengths. The BH is least productive algorithm with minimum returns for all trading periods except on trading period of length 20, where Kao and Tate [3] (KT) return is the minimum among the set of considered algorithms. Figure 1 summarizes the performance of algorithms in terms of experimentally achieved competitive ratio. The figure shows the corresponding performance ratio (OPT/ON) for each algorithm. As competitive ratio measures the performance of algorithm against optimum offline algorithms, thus, a value close to 1 shows the better performance of algorithm. For all trading periods, YFKT performance remains the best in terms of competitive ratio. DAS and BH are among the worst performing algorithms.

**Number of Transactions:** Table 3 summarizes the number of transactions carried out by each algorithm on $DAX30$ dataset. It can be seen that irrespective of the length of trading periods, non-preemptive algorithms RPMm, KT and BH have only two transactions per trading period. This is because the working of non-preemptive algorithm which invests the wealth (for both buy and sell) at one point, thus one buy and sell transaction in a trading period. In preemptive algorithms, the number of transactions varies and depends on the length of investment horizon. Among preemptive algorithms, YFKT has the least number of transactions, which varies from 4.36 (trading period of length 10) to 25.7 (trading period of length 260), whereas the maximum number of transactions are recorded for DAS, which amounts to the number of trading days in the investment horizon.

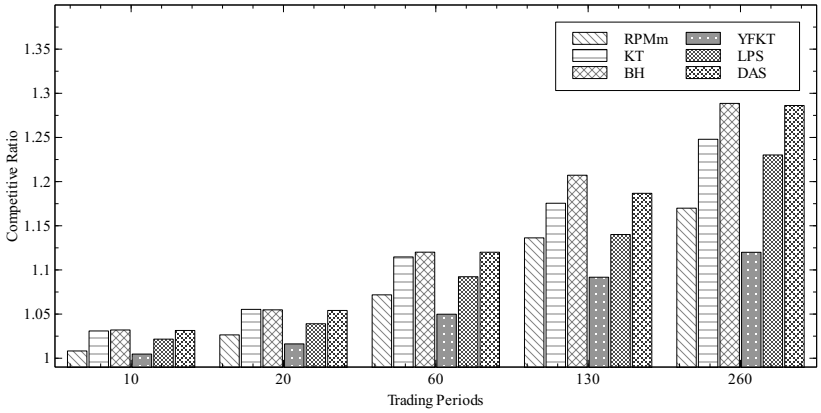


**Figure 1** Experimentally achieved competitive ratio on DAX30.

**Table 3** Average number of transactions per period.

| Period | 10 | 20 | 60 | 130 | 260 |
|---|---|---|---|---|---|
| RPMm | 2 | 2 | 2 | 2 | 2 |
| KT | 2 | 2 | 2 | 2 | 2 |
| BH | 2 | 2 | 2 | 2 | 2 |
| YFKT | 4.36 | 6.272 | 11.3 | 18.45 | 25.7 |
| LPS | 5.98 | 10.96 | 32.7 | 67.15 | 131.3 |
| DAS | 10 | 20 | 60 | 130 | 254.2 |

## 5 Discussion

Based on the different evaluation criteria and the resultant performance, we observe that the YFKT is the best among set of considered algorithms. The performance gap between YFKT and the rest of the considered set of algorithms can be summarized by the fact that YFKT remains the best performing algorithm for trading periods of all lengths. Considering average period return ($APR$) the performance of YFKT is found on average 2% to 11% better over a trading period of length 10 to 260. Another significant observation is the performance of RPMm, which was found to be the second best. The performance difference in term of RPMm and YFKT, considering $APR$ is on average 3.02%, with a minimum difference of 0.19% (trading period of length 10) and maximum difference of 6.76% (trading period of length 260).

BH and DAS are the two least performing algorithms considering $APR$ as evaluating criterion. On $DAX30$, BH returns are negative (less than 1) for all trading periods. While considering $AGR$ as performance evaluation criterion, there is no significant change in the performance ordering of the algorithms, YFKT and RPMm are the best performing whereas BH and DAS are the least performing algorithms.

The performance comparison of algorithm based on $AGR$ and $APR$ reflects that investment horizon of smaller length results in over all higher returns as the accumulated wealth after each trading period is invested over and over again in the next trading period. For instance, if we consider the $AGR$ of YFKT for trading period of length 10, YFKT returns are 1.4411 in comparison to the $APR$ of same trading period which is 1.0142 only.

Analyzing performance of algorithms, the better performance of YFKT can be attributed

**Table 4** Gap between theory and practice.

| **Algorithm** | $r_e$ | $r_w$ | **Gap** $(r_e/r_w)$ |
|---|---|---|---|
| RPMm | 1.0104 | 0.795 | 1.271 |
| BH | 0.8204 | 0.5573 | 1.472 |
| YFKT | 1.0694 | 0.878 | 1.218 |
| LPS | 1.1249 | 1.0975 | 1.024 |
| DAS | 0.81 | 0.5192 | 1.559 |

to the underlying principles and assumptions of the algorithm, YFKT assumes information about the lower and upper bound of offered prices and it neither invests at one point of time nor on all days but invests a portion of wealth when it encounters a new maximum (minimum for buying). Thus, it results in better performance in comparison to other algorithms which either converts at single price based on some pre-calculated reservation price or converts on all days irrespective of the offered price. Although, LPS works on the same principle as YFKT, i.e., convert only when a new maximum (minimum for buying) is encountered, it is not as competitive as YFKT, the main reason can be traced to the amount of wealth invested ($s_t$) after an investment decision is made. YFKT considers the offered price $q_t$ when calculating $s_t$ whereas LPS does not take into account the offered price $q_t$, but instead invests an equal portion of remaining wealth based on the remaining numbers of days.

In terms of number of transactions, all non-preemptive algorithms (RPMm, KT, BH) carries 2 transactions in each trading period, one each for buying and selling, whereas the number of transactions performed by preemptive algorithms (YFKT, LPS, DAS) varies depending on the length of trading period. In comparison to other preemptive algorithms, YFKT performs the least transactions. On dataset $DAX30$, the average number of transactions varies from 4 to 25 for trading period of lengths 10 to 260. The highest number of transactions are performed by DAS which is equal to the number of days in the investment horizon.

The relatively low number of transactions of YFKT can be attributed to the working principle of the algorithm, as it does not convert on every offered price but considers progressively higher (lower for buying) prices only, this not only results in better performance of algorithm but also reduces the number of transactions. DAS has the largest number of transaction, one each per day, whereas LPS which also invests only on the highest (lowest for buying) price seen, resulting in lower number of transactions than DAS.

**Gap between theory and practice:** Analyzing the gap between the theoretical worst case and experimental observed performance of an algorithm is an important aspect of the experimental evaluation of algorithms. We consider yearly data (i.e., dataset of length 260 only and exclude the shorter trading periods) to observe the gap between the theoretical worst case and experimentally achieved performance. For each yearly data, we record the (possible) worst case return ($r_w$) of the algorithm as well as the experimentally observed return ($r_e$) and select where the gap (ratio of worst case to that of observed performance) is the highest. For example to calculate worst case return ($r_w$) of RPMm, we assume that the algorithm achieves the worst case competitive ratio in both buy and sell periods (i.e., RPMm buys at $q_b = \sqrt{Mm}$, whereas OPT buys at $m$, RPMm sells at $q_s = \sqrt{Mm}$ and OPT sells at $M$. Thus the worst case return of RPMm is $q_s/q_b$). The algorithm suggested by Kao and Tate [3] is not included as the working of the algorithm is based on "rank" rather than "actual prices". (Please see [3] Theorem 2.1). It is pertinent to note that we selected only the

instances where the gap between the worst case and experimentally observed return is the maximum.

Table 4 summarizes the gap between theoretically worst case and experimentally observed returns of algorithms. It can be seen that DAS and BH has a considerable gap between the worst case and experimentally achieved returns. The least $r_e$ to $r_w$ ratio is observed for LPS, whereas RPMm and YFKT have almost identical gaps.

## 6 Conclusion

We present an experimental study of online algorithms for the trading problem and compare the results on real world data with an optimum offline algorithm and BH. We observe that all online algorithms perform better than BH and that performance of YFKT is the best among the considered set of algorithms on the $DAX30$ dataset. Further, we deduce that performance behavior of algorithms depends on two main factors, the extent of a-priori information available to the algorithm and the amount of wealth invested per transaction.

The study also finds a number of open questions, such as the performance of YFKT and RPMm is based on the a-priori information such as $m$ and $M$, however, in real world the a-priori information is subject to errors, it will be interesting to note the performance degradation if the a-priori information is erroneous.

## References


1 W.A. Brock, J. Lakonishok, and B. LeBaron. Simple technical trading rules and the stochastic properties of stock returns. *Journal of Finance*, 47:1731–1764, 1992.

2 R. El-Yaniv, A. Fiat, R.M. Karp, and G. Turpin. Optimal search and one-way trading algorithm. *Algorithmica*, 30:101–139, 2001.

3 K.-Y. Kao and S.R. Tate. On-line difference maximization. *SIAM Journal on Discrete Mathematics*, 12(1):78–90, 1999.

4 K.-Y. Kwon and R.J. Kish. A comparative study of technical trading strategies and return predictability: An extension of brock, lakonishok, and lebaron (1992) using NYSE and NASDAQ indices. *Quarterly Review of Economics and Finance*, 42(3):613–633, 2002.

5 J. Lorenz, K. Panagiotou, and A. Steger. Optimal algorithms for $k$-search with application in option pricing. *Algorithmica*, 55(2):311–328, 2009.

6 T.C. Mills. Technical analysis and the london stock exchange: testing trading rules using the FT30. *International Journal of Finance & Economics*, 2(4):319–331, 1998.

7 E. Mohr and G. Schmidt. Trading in financial markets with online algorithms. In B. Fleischmann, K.-H. Borgwardt, R. Klein, and A. Tuma, editors, *Operations Research Proceedings 2008*, pages 33–38. Selected Papers of the Annual International Conference of the German Operations Research Society (GOR) University of Augsburg, September 3-5, 2008, Springer, Heidelberg, 2008.

8 M. Ratner and R. P.C. Leal. Tests of technical trading strategies in the emerging equity markets of latin america and asia. *Journal of Banking and Finance*, 23:1887–1905, 1999.

9 G. Schmidt, E. Mohr, and M. Kersch. Experimental analysis of an online trading algorithm. *Electronic Notes in Discrete Mathematics*, 36:519–526, 2010.

10 P. Shen. Market-timing strategies that worked. *Jounal of Portfolio Management*, 29(2):57–68, 2003.

11 B.M. Tabak and E.J.A. Lima. Market efficiency of brazilian exchange rate: Evidence from variance ratio statistics and technical trading rules. *European Journal of Operational Research*, 194:814–820, 2009.