Algorithmica (2019) 81:4238–4257
https://doi.org/10.1007/s00453-018-00534-9

Optimal Online Two-Way Trading with Bounded Number of Transactions

Stanley P. Y. Fung[1]

Received: 29 September 2017 / Accepted: 26 November 2018 / Published online: 11 December 2018

Abstract
We consider a two-way trading problem, where investors buy and sell a stock whose price moves within a certain range. Naturally they want to maximize their profit. Investors can perform up to k trades, where each trade must involve the full amount. We give optimal algorithms for three different models which differ in the knowledge of how the price fluctuates. In the first model, there are global minimum and maximum bounds m and M. We first show an optimal lower bound of φ (where $\varphi = M/m$) on the competitive ratio for one trade, which is the bound achieved by trivial algorithms. Perhaps surprisingly, when we consider more than one trade, we can give a better algorithm that loses only a factor of $\varphi^{2/3}$ (rather than φ) per additional trade. Specifically, for k trades the algorithm has competitive ratio $\varphi^{(2k+1)/3}$. Furthermore we show that this ratio is the best possible by giving a matching lower bound. In the second model, m and M are not known in advance, and just φ is known. We show that this only costs us an extra factor of $\varphi^{1/3}$, i.e., both upper and lower bounds become $\varphi^{(2k+2)/3}$. Finally, we consider the bounded daily return model where instead of a global limit, the fluctuation from one day to the next is bounded, and again we give optimal algorithms, and interestingly one of them resembles common trading strategies that involve stop loss limits.

Keywords Online algorithms · Competitive analysis · Trading · Stop loss

1 Introduction

1.1 The Model

We consider a scenario commonly faced by investors. The price of a stock varies over time. In this paper we use a 'day' as the smallest unit of time, so there is one new

✉ Stanley P. Y. Fung
pyf1@leicester.ac.uk

[1] Department of Informatics, University of Leicester, Leicester LE1 7RH, UK

price each day. Let $p(i)$ be the price at day i. The investor has some initial amount of money. Over a time horizon of finite duration T, the investor wants to make a bounded number of trades of this one stock. Each trade (b, s) consists of a buy transaction at day b, followed by a sell transaction at day s where $s > b$. (Thus one trade consists of two transactions.) Both transactions are 'all-in': when buying, the investor uses all the money available, and when selling, all stock they currently own is sold. A sale must be made before the next purchase can take place. Also, no short selling is allowed, i.e., there can be no selling if the investor is not currently holding stock. When the end of the time horizon is reached, i.e., on the last day, no buying is allowed and the investor must sell off all the stocks that they still hold back to cash at the price of the day.

There are a number of rationales for considering a bounded number of trades and/or that trades must involve all the money available. Individual, amateur investors typically do not want to make frequent transactions due to high transaction fees. Often transaction fees have a fixed component (i.e., a fixed amount or a minimum tariff per transaction, irrespective of the trading amount) which makes transaction fees disproportionally high for small trades. Frequent trading also requires constant monitoring of the markets which amateur investors may not have the time or resources for; often they only want to change their investment portfolios every now and then. Also, for investors with little money available, it is not feasible or sensible to divide them into smaller pots of money, in arbitrary fractions as required by some algorithms. The finiteness of the time horizon (and that its length is possibly unknown as well) corresponds to situations where an investor may be forced to sell and leave the market due to unexpected need for money elsewhere, for example.

Each trade with a buying price of $p(b)$ and a selling price of $p(s)$ gives a *gain* of $p(s)/p(b)$. This represents how much the investor has after the trade if they invested 1 dollar in the beginning. Note that this is a ratio, and can be less than 1, meaning there is a loss, but we will still refer to it as a 'gain'. If a series of trades are made, the overall gain or the *return* of the algorithm is the product of the gains of each of the individual trades. This correctly reflects the fact that all the money after each trade is re-invested in the next.

Since investors make decisions without knowing future stock prices, the problem is *online* in nature. We measure the performance of online algorithms with *competitive analysis*, i.e., by comparing it with the optimal offline algorithm OPT that knows the price sequence in advance and can therefore make optimal decisions. The *competitive ratio* of an online algorithm ONL is the worst possible ratio of the return of OPT to the return of ONL, over all possible input (price) sequences. The multiplicative nature of the definition of the return (instead of specifying a negative value for a loss) means that the competitive ratio can be computed in the normal way in the case of a loss: for example, if OPT obtains a return of 2 and ONL gets a return of 1/3, then the competitive ratio is 6.

1.2 Three Models on the Knowledge of the Online Algorithm

We consider three different models on how the price changes, or equivalently, what knowledge the online algorithm has in advance. In the first model, the stock prices

are always within a range $[m..M]$, i.e., m is the minimum possible price and M the maximum possible price. Both m and M are known to the online algorithm up front. In the second model, the prices still fluctuate within this range, but m and M are not (initially) known; instead only their ratio $\varphi = M/m$, called the *fluctuation ratio*, is known. In both these models the length of the time horizon (number of days) is unknown (until the final day arrives). In the third model, called the *bounded daily return model*, there is no global minimum or maximum price. Instead, the maximum fluctuation from day to day is bounded: namely, the price $p(i+1)$ of the next day is bounded by the price $p(i)$ of the current day by $p(i)/\beta \leq p(i+1) \leq \alpha\, p(i)$ for some $\alpha, \beta > 1$. This means the prices cannot suddenly change a lot. Many stock markets implement the so-called 'circuit breakers' where trading is stopped when such limits are reached. Here α, β and the trade duration T are known to the online algorithm. All three models are well-established in the algorithms literature; see e.g. [1,5].

1.3 Previous Results and Related Work

Financial trading and related problems are obviously important topics and have been much studied from the online algorithms perspective. A comprehensive survey is given by Mohr et al. [10]. Here we only sample some of the more important results and those closer to the problems we study here. In the *one-way search problem*, the online player chooses one moment of time to make a single transaction from one currency to another currency. Its return is simply the price at which the transaction takes place. A reservation price (RP) based policy is to buy as soon as the price reaches or goes above a pre-set *reservation price*. It is well-known that, if m and M are known, the RP policy with a reservation price of $\sqrt{Mm}$ is optimal and achieves a competitive ratio of $\sqrt{\varphi}$. If only φ is known, then no deterministic algorithm can achieve a ratio better than φ. With the help of randomization, however, a random mix of different RPs gives a competitive ratio of $O(\log \varphi)$ if φ is known. Even if φ is not known, a competitive ratio of $O(\log \varphi \cdot \log^{1+\epsilon}(\log \varphi))$ can be achieved. See [5] for all the above results and more discussions.

In the *one-way trading problem*, the objective is again to maximize the final amount in the other currency, but there can be multiple transactions, i.e., not all the money has to be traded in one go. (This distinction of terminology between search and trading was used in [5], but was called non-preemptive vs. preemptive in [10]. We prefer calling them *unsplittable* vs. *splittable* here.) A 'threat-based' algorithm for one-way trading was given in [5]; it has the optimal competitive ratio. They also established a relation between one-way trading and randomized algorithms for one-way search. Many variations of one-way search or one-way trading problems have since been studied; some examples include the bounded daily return model [1,14], searching for k minima/maxima instead of one [9], time-varying bounds [4], unbounded prices [2], search with advice complexity [3], etc.

What we study here, however, is a *two-way* version of the unsplittable trading problem,[1] which is far less studied. Here the online player has to first convert from

[1] In the terminology of [5] this should be called '*two-way search*', but we feel that the term does not convey its application in stock market trading.

one currency (say cash) to another (a stock), hopefully at a low price, and then convert back from the stock to cash at some later point, hopefully at a high price. All the investment must be converted back to the first currency when or before the game ends. This model is relevant where investors are only interested in short term, speculative gains. For the models with known m, M or known φ and with one trade, Schmidt et al. [11] gave a φ-competitive algorithm; it uses the same RP for buying and selling. But consider the DO- NOTHING algorithm that makes no trades at all. Clearly it is also φ-competitive as ONL's gain is 1 and OPT's gain is at most φ (if the price goes from m to M). A number of common trading strategies, such as those based on moving averages, were also studied in [10]. It was shown that they are φ^2-competitive (and not better), which are therefore even worse. It is easy to show that these algorithms have competitive ratios φ^k and φ^{2k} respectively when extended to k trades. Schroeder et al. [13] gave some algorithms for the bounded daily return model, without limits on the number of trades. However, most of the above algorithms tend to make decisions that are clearly bad, have the worst possible performance (like losing by the largest possible factor every day throughout), and have competitive ratios no better than what is given by DO- NOTHING.

Very recently Schmidt [12] considered the same two-way unsplittable problem, but parameterized by the number of 'runs' (a sequence of monotonically increasing or decreasing prices), not the number of trades permitted. In other words they bound the number of local maxima/minima. Kao and Tate [8] also considered a kind of 'two-way trading' problem, but there are several crucial differences with ours: (1) their 'prices' are ranks from 1 to n; (2) the 'profit' of each trade is the difference in ranks, and the objective is to maximise the sum of profits; (3) the input is in uniformly random order, as in the *secretary problem*; (4) they consider either one trade or an unrestricted number of trades.

Finally, for the splittable version of the two-way trade problem, El-Yaniv et al. [5] gave an algorithm based on their threat-based one-way trading algorithm. The competitive ratio achieved and also the lower bound they gave are exponential in the number of local optima of the price sequence.

1.4 Our Results

In this paper we consider the two-way unsplittable trading problem where a bounded number k of trades are permitted, and derive optimal algorithms. First we consider the model with known m and M. We begin by considering the case of $k = 1$. Although some naive algorithms are known to be φ-competitive and seemingly nothing better is possible, we are not aware of any matching general lower bound. We give a general lower bound of φ, showing that the naive algorithms cannot be improved. The result is also needed in subsequent lower bound proofs.

It may be tempting to believe that nothing can beat the naive algorithm also in the case of more trades. Interestingly, we prove that for $k \geq 2$ this is not true. While naive algorithms like DO- NOTHING are no better than φ^k-competitive, we show that a reservation price-based algorithm is $\varphi^{(2k+1)/3}$-competitive. For example, when $k = 2$,

it is $\varphi^{5/3}$-competitive instead of trivially φ^2-competitive. Furthermore, we prove a matching lower bound, showing that the algorithm is optimal.

Next, we consider the model where only φ is known, and give an algorithm with a competitive ratio of $\varphi^{(2k+2)/3}$, i.e., only a factor $\varphi^{1/3}$ worse than that of the preceding model that has more knowledge. Again we show that this bound is optimal.

Finally we consider the bounded daily return model, and give two optimal algorithms where the competitive ratio depends on α, β and T. For example, with one trade and in the symmetric case $\alpha = \beta$, the competitive ratio is $\alpha^{2T/3}$. While this is exponential in T (which is unavoidable), naive algorithms could lose up to a factor of $\max(\alpha, \beta)$ every day, and DO- NOTHING has a competitive ratio of α^T. One of the algorithms uses the 'stop loss / lock profit' strategy commonly used in real trading; as far as we are aware, this is the first time where competitive analysis justifies this common stock market trading strategy, and in fact suggests what the stop loss limit should be. There were precedents where competitive analysis derived known investment strategies: El-Yaniv et al. [5] observed that their algorithm for one-way splittable trading, when given the worst possible price sequence, obeys the conventional wisdom of 'dollar-cost averaging'. However, in the bounded daily return model, dollar-cost averaging is not optimal [1].

2 Known *m* and *M*

In this section, where we consider the model with known m and M, we can without loss of generality assume that $m = 1$. It also means M and φ are equal and are sometimes used interchangeably. Furthermore, we note that we can assume without loss of generality that any algorithm does not buy at price M, since any trade with such a buying price can only have a gain at most 1, and removing the trade cannot worsen the total return. Similarly we can assume no algorithms sell at price m.

Lemma 1 *For any $\epsilon > 0$ and any deterministic online algorithm ONL, there exist a price sequence such that OPT obtains a return more than that of ONL by a factor of $\varphi^{1-\epsilon}$. This applies even if ONL can make an unlimited number of trades and OPT only uses one trade.*

Proof Choose $n = \lceil 1/\epsilon \rceil$ and define $v_i = M^{i/n}$ for $i = 0, 1, \ldots, n$. The following price sequence is released until ONL buys:

$$v_{n-1}, M, v_{n-2}, M, \ldots, v_i, M, \ldots, v_1, M, v_0$$

If ONL does not buy at any point, then its return is at most 1. Then OPT buys at v_1 and sells at M to get a return of $M^{1-1/n}$. So suppose ONL buys at v_i for some $1 \le i \le n-1$. (It cannot buy at v_0 as it is the last day.) As soon as ONL bought, the rest of the sequence is not released; instead the price drops to m and the game ends. ONL's return is m/v_i. If $i = n-1$, then OPT makes no trade and its return is 1, so the ratio between their returns is $v_{n-1}/m = M^{1-1/n}$. Otherwise, if $i < n-1$, OPT buys at v_{i+1} (two days before ONL's purchase) and sells at the next day at price M, giving a return of M/v_{i+1}. The ratio between their returns is therefore $Mv_i/(mv_{i+1}) = M^{1-1/n}$.

Thus in all cases the ratio between their returns is at least $M^{1-1/n} \geq \varphi^{1-\epsilon}$. □

For $k = 1$ (both OPT and ONL are allowed one trade), the above lemma immediately implies that no deterministic algorithm is better than $\varphi^{1-\epsilon}$-competitive. It is also obvious that the DO- NOTHING algorithm is φ-competitive, thus the bounds are (almost) tight.

For general $k \geq 1$, we analyze the following algorithm.

Algorithm 1 The reservation price algorithm, with known price range $[m..M]$

```
Upon release of the i-th price p(i):
if i = T then
  if currently holding stock then
    sell at price p(T) and the game ends
else
  if p(i) ≤ M^{1/3} and currently not holding stock and not used up all trades then
    buy at price p(i)
  else if p(i) ≥ M^{2/3} and currently holding stock then
    sell at price p(i)
```

Theorem 1 *Algorithm 1 has competitive ratio $\varphi^{(2k+1)/3}$, for $k \geq 1$.*

Proof First we make a few observations. We call an ONL trade *winning* if its gain is higher than 1, and *losing* otherwise. Any winning trade has a gain of at least $M^{1/3}$. If ONL makes a losing trade, it must be a forced sale at the end and the gain is not worse than $M^{-1/3}$. Moreover, it follows that the algorithm cannot trade any more after it.

We consider a number of cases separately based on the sequence of win/loss trades. Let W and L denote a winning trade and a losing trade, respectively. Following the above discussion, the possible sequences of trades are: NT (no trade), W^j for $1 \leq j \leq k$, and $W^j L$ for $0 \leq j \leq k-1$. Here W^j denotes a sequence of j consecutive W's.

Case NT: Since ONL has never bought, the prices were never at or below $M^{1/3}$ (except possibly the last one, but neither OPT nor ONL can buy there) and hence OPT's return is at most $(M/M^{1/3})^k = M^{2k/3}$. ONL's return is 1. So the competitive ratio is at most $M^{2k/3}$.

Case W^j, $1 \leq j \leq k-1$: Suppose ONL buys at days $b_1, b_2, \ldots, b_j$ and sells at days $s_1, s_2, \ldots, s_j$. Define $s_0 = 0$. Partition the timeline into regions $[s_{i-1}, b_i)$ when ONL is not holding stock, and $[b_i, s_i)$ when it is. In each $[s_{i-1}, b_i)$ and also $[s_j, T)$, the prices must be at least $M^{1/3}$ or else ONL would have bought (note that ONL must have remaining trades to use since it made at most $k-1$ trades under the case assumption; the case W^k is considered separately); in each $[b_i, s_i)$, the prices must be at most $M^{2/3}$ or else ONL would have sold.

If an OPT trade does not 'cross' any s_i's (i.e., both its buying and selling dates are within the same $[s_{i-1}, s_i)$), its gain can be at most $M^{2/3}$: if it bought in $[s_{i-1}, b_i)$ then its buying price is already at least $M^{1/3}$ and its selling price is obviously at most M; and if it buys in $[b_i, s_i)$ then its selling price is already at most $M^{2/3}$.

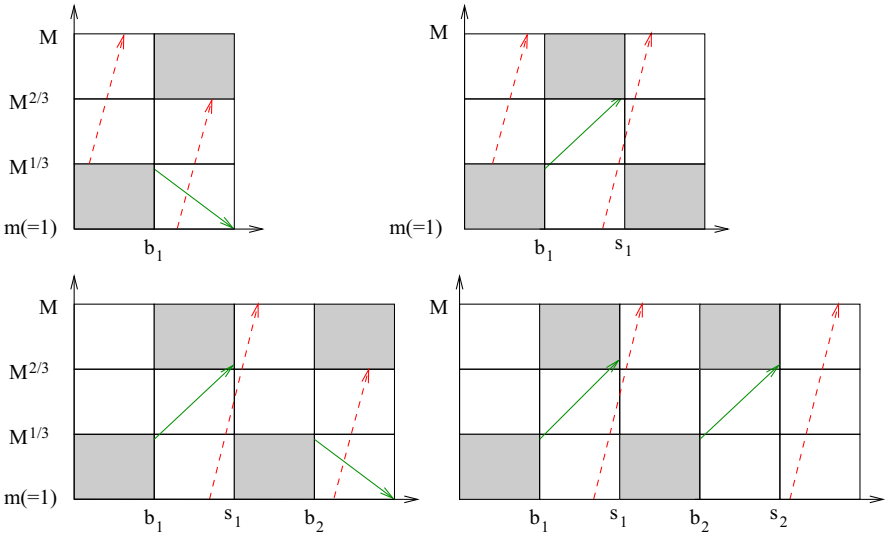

Fig. 1 Four cases illustrated, for $k = 2$. Horizontal axis is time, vertical axis is price. Shaded regions are the regions where the prices cannot fall into. Green solid arrows depict possible buying and selling actions of ONL, red dashed arrows for OPT. Top left: case L, Top right: case W, Bottom left: case WL, Bottom right: case WW

In other words, if an OPT trade has a gain more than $M^{2/3}$, then it must cross a selling day of ONL. See Fig. 1 for some examples.

Since there are at most j selling days to cross, at most j trades of OPT have gain higher than $M^{2/3}$ (but at most M) and the rest have gain at most $M^{2/3}$. OPT's return is therefore at most $M^j(M^{2/3})^{k-j}$. ONL's return is at least $(M^{1/3})^j$. So the competitive ratio is at most $(M^j(M^{2/3})^{k-j})/(M^{1/3})^j = M^{2k/3}$.

Case W^jL, $0 \le j \le k-1$: The difference with the previous case is that there is a forced, losing sale at the last day. The analysis of OPT in the previous case still applies here. (The last sale at day T is losing and so the price is at most $M^{2/3}$ and does not provide an opportunity for OPT to make a gain higher than $M^{2/3}$.) So OPT's gain remains the same, $M^j(M^{2/3})^{k-j}$. ONL's gain is at least $(M^{1/3})^j M^{-1/3}$ because of the final losing trade. Thus the competitive ratio is $(M^j(M^{2/3})^{k-j})/((M^{1/3})^j M^{-1/3}) = M^{(2k+1)/3}$.

Case W^k: In this case OPT gains at most M per trade while ONL gains at least $M^{1/3}$ per trade. Hence the competitive ratio is simply $M^k/M^{k/3} = M^{2k/3}$.

□

Theorem 2 *No deterministic algorithm has a competitive ratio better than* $\varphi^{(2k+1)/3-\epsilon}$, *for any* $\epsilon > 0$ *and* $k \ge 1$.

Proof The prices are released in up to k rounds. Round k is a special round. For all other normal rounds, we maintain the following invariants. Just before the i-th normal round starts, OPT completed exactly $i-1$ trades, is holding no stock, and accumulated a return of exactly M^{i-1}, while ONL completed at most $i-1$ trades, is holding no

stock, and accumulated a return of at most $M^{(i-1)/3}$. Obviously the invariants hold just before the first round.

A normal round i ($i < k$) begins with the price sequence $M^{1/3}, M, M^{1/3}, M, \ldots$ until either (1) ONL buys at one of the $M^{1/3}$ prices (recall that ONL does not buy at M), or (2) $k - i$ such pairs of oscillating prices have been released.

> Case 1: ONL buys at one of the $M^{1/3}$ prices. As soon as this happens, the rest of that sequence will not be released. Instead, the price sequence that follows is $m, M^{2/3}, m, M^{2/3}, \ldots$ until either (1.1) ONL sells at one of the $M^{2/3}$ prices (recall that ONL does not sell at m), or (1.2) $k - i + 1$ such pairs of oscillating prices have been released.
>
> > Case 1.1: ONL sells at one of the $M^{2/3}$ prices. As soon as this happens, the rest of that sequence will not be released; instead the price goes up to M and this round ends. OPT makes one trade (from m to M) in this round, gaining M, while ONL gains $M^{1/3}$. Thus the invariants regarding returns and number of trades are maintained. Furthermore, recall that ONL does not buy at the last price M, thus the invariant regarding ONL not holding stock is also maintained. We move on to the next normal round if $i < k - 1$. If $i = k - 1$, we move on to the special final round.
> > Case 1.2: If ONL does not sell at any of these prices, one final price m is released and the game ends (with no further rounds, not even the special round). ONL is forced to sell at m, making a gain of $M^{-1/3}$ in this round. Meanwhile OPT uses all its remaining $k - i + 1$ trades and gains $(M^{2/3})^{k-i+1}$. Combining with the returns from previous rounds, OPT's return is $M^{i-1}(M^{2/3})^{k-i+1}$ while ONL's return is at most $M^{(i-1)/3}M^{-1/3}$. The competitive ratio is therefore at least $M^{(i-1)+2(k-i+1)/3}/M^{(i-1)/3-1/3} = M^{(2k+1)/3}$.
>
> Case 2: If ONL does not buy at any of those prices, then the round ends. In this round, OPT makes $k - i$ trades with a total gain of $(M^{2/3})^{k-i}$. Thus up to this point OPT has used $(i - 1) + (k - i) = k - 1$ trades and obtained a return of $M^{i-1}(M^{2/3})^{k-i}$. ONL makes no further trade or gain, and thus the return is still at most $M^{(i-1)/3}$. Any remaining normal rounds are then skipped and we jump directly to the special last round k. Note that the invariants do not need to be maintained at this point, because we are now going to the special round.

Finally, consider the special round k. If we arrive at this after Case 1.1 happened $k - 1$ times, then the invariants guarantee that the ratio of OPT's return to ONL's return up to this point is at least $M^{2(k-1)/3}$. If we arrive at this after Case 2, then the invariants do not apply, but the ratio of their returns is $M^{i-1}(M^{2/3})^{k-i}/M^{(i-1)/3}$ which is also $M^{2(k-1)/3}$. In either case, at this point, OPT has exactly one trade left, and ONL has one or more trades left. We then release the price sequence of Lemma 1, which guarantees that OPT gains more than ONL by an additional factor of $M^{1-\epsilon}$. Thus the competitive ratio is not better than $M^{2(k-1)/3+1-\epsilon} = M^{(2k+1)/3-\epsilon}$. □

Note that, like Lemma 1, this result does not depend on how many trades ONL is allowed; more trades would not help.

3 Known φ Only

For $k = 1$ Do- Nothing is clearly still φ-competitive, and Lemma 1 also works in this model to give a lower bound of $\varphi^{1-\epsilon}$, so we focus on $k > 1$. We adapt Algorithm 1 by buying only when it is certainly 'safe', i.e., when it is certain that the price is within the lowest $\varphi^{1/3}$ of the actual price range, and sells when it is guaranteed a gain of $\varphi^{1/3}$. The formal description is given in Algorithm 2. Let M_t be the maximum price observed up to and including day t. Note that M_t is a stepwise increasing function of t.

Algorithm 2 The reservation price algorithm, with known φ

```
Upon release of the i-th price p(i):
if i = T then
  if currently holding stock then
    sell at price p(T) and the game ends.
else
  if i = 1 then
    M_1 := p(1)
  else
    M_i := max(M_{i-1}, p(i))
  if p(i) ≤ M_i/φ^{2/3} and currently not holding stock and not used up all trades then
    buy at price p(i)
  else if currently holding stock bought at price p(b) and p(i) ≥ φ^{1/3} p(b) then
    sell at price p(i)
```

Theorem 3 *Algorithm 2 has competitive ratio $\varphi^{(2k+2)/3}$, for any $k \geq 2$.*

Proof Clearly ONL gets the same as in Theorem 1: each winning trade has gain at least $\varphi^{1/3}$ and a losing trade, which can only appear as the last trade, has gain at least $\varphi^{-1/3}$. The difference is in how we bound OPT's gain.

If ONL makes k winning trades, then the competitive ratio is simply $\varphi^k/\varphi^{k/3} = \varphi^{2k/3}$. (This is the same argument as Case W^k of Theorem 1.) So in the following we only consider the case where ONL did not use up all its trades, i.e., it is always able to buy if it is not holding.

A *sell event* happens at a day when ONL sells and makes a profit (i.e., it excludes the forced sale at day T). An *M-change event* happens at day t if $M_t \neq M_{t-1}$. Each OPT trade (b^*, s^*) can be classified into one of the following types:

(1) There is at least one sell event during $[b^*, s^*]$. Clearly the number of such OPT trades is limited by the number of sell events. Each such trade can gain up to φ.
(2) There is no sell event during $[b^*, s^*]$, and at b^* ONL is either holding or buying. Suppose ONL's most recent purchase on or before b^* is at day b. Then $p(b) \leq M_b/\varphi^{2/3} \leq M_{b^*}/\varphi^{2/3}$. It is holding stock throughout and still did not sell at s^* (or is forced to sell if s^* is the last day), hence $p(s^*) < p(b)\varphi^{1/3} \leq M_{b^*}/\varphi^{1/3}$. But clearly $p(b^*) \geq M_{b^*}/\varphi$ (the price at day b^* must be within a factor of φ from the maximum seen up to day b^*), hence the gain of this OPT trade is $p(s^*)/p(b^*) < \varphi^{2/3}$.

(3) There is no sell event during $[b^*, s^*]$, at b^* ONL is neither holding nor buying, and there is no M-change event in $(b^*, s^*]$. We have $p(b^*) > M_{b^*}/\varphi^{2/3}$ as otherwise ONL would have bought at b^*. Clearly $p(s^*) \le M_{s^*} = M_{b^*}$. Hence the gain of this OPT trade is $p(s^*)/p(b^*) < \varphi^{2/3}$.

(4) There is no sell event during $[b^*, s^*]$, at b^* ONL is neither holding nor buying, and there is/are M-change event(s) in $(b^*, s^*]$. Suppose there are a total of x such OPT trades, $(b_1^*, s_1^*), (b_2^*, s_2^*), \ldots, (b_x^*, s_x^*)$, in chronological order. Note that $p(b_i^*) > M_{b_i^*}/\varphi^{2/3}$ for each i, or else ONL would have bought at b_i^*. So for all i, $p(b_{i+1}^*) > M_{b_{i+1}^*}/\varphi^{2/3} \ge M_{s_i^*}/\varphi^{2/3} \ge p(s_i^*)/\varphi^{2/3}$. Thus the total gain of these x trades is

$$\begin{aligned}\prod_{i=1}^{x} \frac{p(s_i^*)}{p(b_i^*)} &= \frac{1}{p(b_1^*)} \frac{p(s_1^*)}{p(b_2^*)} \cdots \frac{p(s_{x-1}^*)}{p(b_x^*)} \frac{p(s_x^*)}{1} \\ &< \frac{p(s_x^*)}{p(b_1^*)} (\varphi^{2/3})^{x-1} \le \varphi (\varphi^{2/3})^{x-1} = \varphi^{(2x+1)/3}\end{aligned}$$

Suppose ONL makes y winning trades and one losing trade. Then OPT makes at most y trades of type (1), gaining at most φ^y from those. Then, if x of OPT's trades are of type (4), they in total gives another gain of at most $\varphi^{(2x+1)/3}$. The remaining trades are of types (2) and (3), gaining $\varphi^{2/3}$ each. The competitive ratio is therefore at most

$$\frac{\varphi^y \varphi^{(2x+1)/3} \varphi^{2(k-x-y)/3}}{\varphi^{y/3} \varphi^{-1/3}} = \varphi^{(2k+2)/3}$$

If ONL makes $y < k$ winning trades and no losing trade, the competitive ratio can only be better, as OPT's return is as above but ONL's is $\varphi^{1/3}$ better. □

Theorem 4 *No deterministic algorithm is better than $\varphi^{(2k+2)/3-\epsilon}$-competitive, for any $\epsilon > 0$ and $k \ge 2$.*

Proof We first give an initial sequence of prices that will let OPT get a factor of φ better than ONL in its return, but will afterwards reveal the values of m and M to ONL. Afterwards we simply apply Theorem 2.

For the initial sequence, we first release a price of 1. If ONL does not buy, then the price goes up to φ. At this point, ONL knows that the price range is $[1..\varphi]$. OPT makes one trade from 1 to φ and gains φ. ONL would not buy at the highest price φ. Hence both OPT and ONL are not holding stock, OPT made one trade and ONL none, but ONL is a factor of φ behind in the return.

Otherwise, if ONL buys at 1, then the subsequent price sequence is $1/\varphi, 1, 1/\varphi, 1, \ldots$ for up to k such pairs, until ONL sells. Now ONL knows the range is $[1/\varphi..1]$. Again we can assume ONL does not sell at $1/\varphi$. If ONL does not sell at any point, then the game ends immediately. OPT makes k trades gaining φ^k, and ONL's gain is 1. The competitive ratio is φ^k, which is at least $\varphi^{(2k+2)/3}$ for $k \ge 2$. If ONL sells at some point with price 1, then OPT buys at $1/\varphi$ and sells at 1, getting a gain of φ. ONL's gain is 1. At this point, both OPT and ONL used one trade, are not holding stock, and OPT is a factor of φ ahead of ONL.

We now perform the same construction as in Theorem 2, but with a maximum of $k-1$ trades instead of k. Specifically, we release up to $k-2$ normal rounds, possibly followed by a special round, where each round works in exactly the same way as described in Theorem 2. This gives another factor of $\varphi^{(2(k-1)+1)/3-\epsilon}$ to the competitive ratio. Thus combining with the initial sequence, the competitive ratio is at least $\varphi \cdot \varphi^{(2(k-1)+1)/3-\epsilon} = \varphi^{(2k+2)/3-\epsilon}$. □

4 Bounded Daily Return, Known Duration

Recall that in this model, the prices are bounded by $p(i)/\beta \le p(i+1) \le \alpha p(i)$ for some $\alpha, \beta > 1$. Trades can take place at days $0, 1, \ldots, T$. Consider the generalisation of this model to shorter time spans between price points: we receive a new price point (and can trade) every few minutes, every few seconds, etc., rather than every day. The values T, α and β are adjusted accordingly: if a day is subdivided into n points in time then the new values T', α' and β' are such that $T' = nT$, $\alpha' = \alpha^{1/n}$ and $\beta' = \beta^{1/n}$. As the time intervals get shorter, this approaches a continuous model, where time is continuous and the price is a continuous function of time.

The following algorithms and lower bounds involve computing the lengths (durations) of some time intervals, e.g. how many days we hold the stock, but those calculations may return non-integers. Another situation is that we want to sell as soon as the price drops below some threshold x, but in the discrete model, the price is not continuous and so the selling price may not be exactly x (but within a factor of at most $\max(\alpha, \beta)$). To simplify our analysis, in the following we will first assume that time is continuous and that the price is a continuous function of time, whenever it is convenient, so these problems do not arise. This allows the intuition of the algorithms and proofs to be more easily seen. We will briefly discuss the effect of rounding at the final subsection; they have a small effect on the competitive ratio.

4.1 Lower Bound and a Static Algorithm

Theorem 5 *No deterministic algorithm has a competitive ratio better than* $\alpha^{T(2k\log\beta)/((k+1)\log\beta+k\log\alpha)}$.

Proof The adversary strategy is very simple and natural: whenever ONL is not holding stock, the price goes up by a factor of α every day, and while it is holding stock it goes down by β every day. Let the ONL trades be (b_i, s_i), $i = 1, \ldots, k$, in chronological order. (If there are fewer than k trades, dummy ones with $b_i = s_i$ can be added.) For convenience, define $s_0 = 0$ and $b_{k+1} = T$, and for $1 \le i \le k+1$, define $t_{2i-1} = b_i - s_{i-1}$ and $t_{2i} = s_i - b_i$. They are the lengths of time ONL is not holding stock and holding stock, respectively. ONL's return is $1/(\beta^{t_2}\beta^{t_4}\cdots\beta^{t_{2k}})$. OPT's optimal actions, if allowed $k+1$ trades, is to hold during the exact opposite intervals as ONL, i.e., buy at s_i and sell at b_{i+1} for $0 \le i \le k$. But since it can make at most k trades, its possible course of actions include skipping one of those trades, or

making one of the trades 'span across two intervals', e.g., buying at s_i and selling at b_{i+2}. Thus OPT's return is the maximum out of

$$\begin{array}{ll} \alpha^{t_3}\alpha^{t_5}\cdots\alpha^{t_{2k-1}}\alpha^{t_{2k+1}} & \alpha^{t_1}\alpha^{t_3}\cdots\alpha^{t_{2k+1}}/\beta^{t_2} \\ \alpha^{t_1}\alpha^{t_5}\cdots\alpha^{t_{2k-1}}\alpha^{t_{2k+1}} & \alpha^{t_1}\alpha^{t_3}\cdots\alpha^{t_{2k+1}}/\beta^{t_4} \\ \vdots & \vdots \\ \alpha^{t_1}\alpha^{t_3}\cdots\alpha^{t_{2k-3}}\alpha^{t_{2k+1}} & \alpha^{t_1}\alpha^{t_3}\cdots\alpha^{t_{2k+1}}/\beta^{t_{2k}} \\ \alpha^{t_1}\alpha^{t_3}\cdots\alpha^{t_{2k-3}}\alpha^{t_{2k-1}} & \end{array} \tag{1}$$

The (worst-case) competitive ratio is given by the maximum of these expressions, divided by ONL's return. A lower bound on the competitive ratio therefore arises when we minimize, over all choices of $t_1, \ldots, t_{2k+1}$, the maximum of these expressions, which happens when these expressions are equal. Note that there are $2k+1$ variables and $2k+1$ equations, $2k$ from the equality of these $2k+1$ expressions and another from $\sum t_i = T$. Equating all the $2k+1$ expressions gives $t_1 = t_3 = \cdots = t_{2k+1}$ and $t_2 = t_4 = \cdots = t_{2k}$, and further implies $\alpha^{kt_1} = \alpha^{(k+1)t_1}/\beta^{t_2}$, which gives $\alpha^{t_1} = \beta^{t_2}$. Together with $t_1 + \cdots + t_{2k+1} = (k+1)t_1 + kt_2 = T$, this gives

$$t_1 = \frac{\log\beta}{(k+1)\log\beta + k\log\alpha}T, \quad t_2 = \frac{\log\alpha}{(k+1)\log\beta + k\log\alpha}T \tag{2}$$

and thus the competitive ratio is at least

$$\alpha^{kt_1}/(1/\beta^{kt_2}) = \alpha^{2kt_1} = \alpha^{T(2k\log\beta)/((k+1)\log\beta + k\log\alpha)}.$$

□

For instance, for the symmetric case where $\alpha = \beta$, this gives $t_1 = t_2 = T/(2k+1)$ and a competitive ratio of $\alpha^{2kT/(2k+1)}$.

Algorithm 3 Static algorithm for known α, β and T

Set $t_1 = \frac{\log\beta}{(k+1)\log\beta + k\log\alpha}T$ and $t_2 = \frac{\log\alpha}{(k+1)\log\beta + k\log\alpha}T$.
Upon release of the i-th price $p(i)$:
if $i = T$ **then**
  if currently holding stock **then**
    sell at price $p(T)$ and the game ends
else
  if have not been holding stock for t_1 days and not used up all trades **then**
    buy at price $p(i)$
  else if have been holding stock for t_2 days **then**
    sell at price $p(i)$

Next we consider Algorithm 3. To prove its upper bound, we first show the following, which allows us to only consider some 'canonical' price sequences.

Lemma 2 *Consider any given price sequence $p(0), p(1), \ldots, p(T)$ and an arbitrary day i. The sequence can be transformed in the following ways:*

(i) *If ONL is holding stock or selling at day i, then $p(i) = p(i-1)/\beta$, i.e., the price change from day $i-1$ to i is the maximum possible drop;*
(ii) *If ONL is buying or is not holding stock at day i, then $p(i) = p(i-1)\alpha$, i.e., the price change from day $i-1$ to i is the maximum possible rise.*

If ONL does not change its trading days as a result of this transformation, then the ratio of OPT's and ONL's returns of the new sequence is at least that of the original sequence.

Proof Suppose at day i ONL is holding stock or selling. If $p(i) > p(i-1)/\beta$, i.e., the price change from day $i-1$ to i is not the maximum possible drop, we raise $p(i-1)$ to $\beta p(i)$ and make a corresponding change to all earlier prices, i.e., multiply each of them by a factor of $\beta p(i)/p(i-1)$. For any ONL trade completed (i.e., bought and sold) on or before day $i-1$, their gains are unaffected since both buying and selling prices are multiplied by the same factor. For the one trade where it is holding or selling at day i, ONL's gain reduces by a factor of $\beta p(i)/p(i-1)$ since the buying price of this trade is raised but the selling price is not. The gains of all future trades are unaffected.

As for OPT, assume for the moment that its trading days also will not change as a result of the transformation. If OPT is holding or selling at day i, then it suffers the same change as ONL; otherwise its return is not affected. Hence, in all cases, the ratio of OPT's to ONL's return can only increase.

Similarly, suppose at day i ONL is buying or is not holding stock. If $p(i) < p(i-1)\alpha$, i.e., the price change from day $i-1$ to i is not the maximum possible rise, we lower $p(i-1)$ to $p(i)/\alpha$ and make a corresponding change to all earlier prices, i.e., multiply each of them by a factor of $p(i)/(\alpha p(i-1))$. For any ONL trade already completed before day i, its gain is unaffected since both buying and selling prices are multiplied by the same factor. All future trades are unaffected. For OPT (again assuming no change to its trading days for now), if it is buying or holding at day $i-1$, then the gain of that trade improves from the lower buying price while the selling price (on or after day i) is unchanged. Otherwise its return is unaffected as in ONL. Hence the ratios of OPT's to ONL's return can only increase.

Now, the optimal trades for this new price sequence may not be the same as the original sequence, but that can only improve OPT's return. □

Theorem 6 *Algorithm 3 has competitive ratio $\alpha^{T(2k\log\beta)/((k+1)\log\beta+k\log\alpha)}$.*

Proof We only need to consider price sequences that satisfy the description given in Lemma 2, since the lemma proves that those sequences give the worst possible competitive ratios. Note that ONL's buy/sell decisions do not depend on the prices at all, and so are not affected by the transformation in the prices and Lemma 2 can indeed be applied.

By applying Lemma 2 to all trading days, and given how the algorithm works, we know the worst case price sequence is as follows. It first goes up by a factor of α for t_1 days, during which ONL does not hold stock, then goes down by a factor of β for t_2 days, during which ONL holds stock. Repeat this for k times, and then there is a final upward run of length t_1. This covers the exact length of T, since the values of t_1 and t_2

are such that $(k+1)t_1 + kt_2 = T$. ONL's return is $1/\beta^{kt_2}$. As for OPT, because there are $k+1$ periods when the price goes up, its optimal action is either (i) hold during k of those periods, getting a return of α^{kt_1}, or (ii) as in (i) but 'merge' two trades to span over a downward period, which gives a return of $\alpha^{kt_1}(1/\beta^{t_2})\alpha^{t_1} = \alpha^{kt_1}$ as $\alpha^{t_1} = \beta^{t_2}$. (See also the proof of Theorem 5.) The competitive ratio is thus $\alpha^{kt_1}/(1/\beta^{kt_2}) = \alpha^{2kt_1}$, which is equal to the one given in the theorem statement. □

4.2 A Dynamic, Trailing Stop Based Algorithm

Algorithm 3 may feel unnatural since it does not depend on the price sequence at all (this is called 'static' in [1]). But we prove that the following variation of the algorithm has the same competitive ratio: it sells only when the current price falls below h/β^{t_2} where h is the highest price seen since the last purchase. It has a more intuitive appeal and it coincides with the 'stop loss' strategy very common in real trading (more precisely 'trailing stop' [6] where the stop loss limit is not fixed but tracks the highest price seen thus far, to potentially capture the most profit).

Algorithm 4 Stop loss based algorithm for known α, β and T

```
Set t1 and t2 as in Algorithm 3
Upon release of the i-th price p(i):
if i = T then
  if currently holding stock then
    sell at price p(T) and the game ends
else
  if have not been holding stock for t1 days and not used up all trades then
    buy at price p(i)
    set h = p(i)
  else if currently holding stock then
    set h = max(h, p(i))
    sell at price p(i) if p(i) < h/β^t2
```

Before we prove the competitive ratio, we first make a couple of remarks. Both the static (Algorithm 3) or dynamic (Algorithm 4) algorithms spend exactly the same time (t_1) waiting from one sale to the next purchase. The static algorithm holds stock for exactly t_2 days each time, while the dynamic one holds at least t_2 days, as it takes at least that many days for the price to drop by a factor of β^{t_2}. This means that the dynamic version never make more trades than the static version, so may save transaction costs.

Neither the static or dynamic version always outperform the other. For example, if the price keeps rising for all T days, then the static algorithm will only gain α^{kt_2} but the dynamic one will hold all the way through and gains α^{T-t_1} (Fig. 2a). However, if the prices are like in Fig. 2b (here we set $\alpha = \beta$) where the price drops by just less than β^{t_1} for each static algorithm's non-holding period, and rises by just less than α^{t_2} for each holding period, then the dynamic algorithm will make no profit while the static one gains about α^{kt_2}.

We introduce some notations and prove some basic properties. Recall (from the proof of Theorem 5) that $\alpha^{t_1} = \beta^{t_2}$. Let r denote this common value, and the compet-

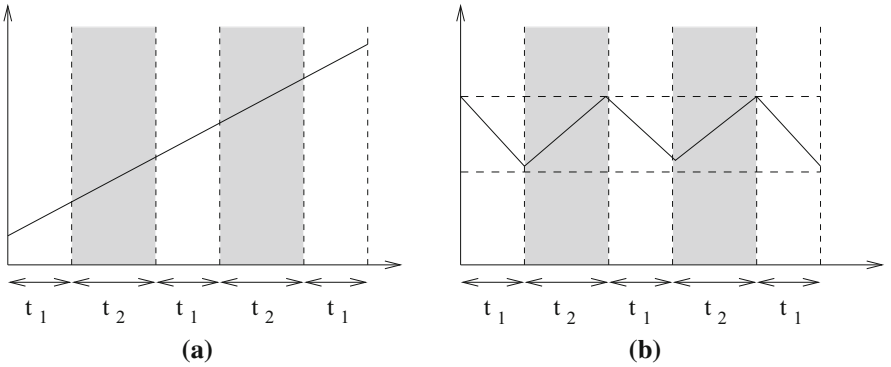

Fig. 2 **a** The static algorithm only holds stocks during the shaded regions, but the dynamic algorithm holds throughout starting at day t_1. **b** Again the static algorithm holds during the shaded regions and the dynamic algorithm holds till the end, but this time it is bad for the dynamic algorithm

itive ratio we want to prove is then equal to r^{2k}. Roughly speaking, our approach is to partition the time horizon so that, in each partition, x trades in OPT are associated to y trades in ONL such that the ratio between their gains is at most r^{x+y}. Since each of OPT and ONL makes at most k trades, this establishes the competitive ratio.

Suppose ONL completed ℓ trades, $\ell \le k$. Denote by (b_i, s_i) ONL's i-th trade. For technical reasons, also define $b_0 = s_0 = 0$ and $b_{\ell+1} = s_{\ell+1} = T$. Let $H_i = [b_i, s_i]$, $0 \le i \le \ell$, be the i-th holding period, i.e., the (closed) time interval where ONL is holding stock in its i-th trade, and let $N_i = (s_{i-1}, b_i)$, $1 \le i \le \ell + 1$, be the (open) i-th non-holding period. H_0 is the holding period for the trivial trade (b_0, s_0) defined for analysis purposes only. If H_ℓ does not contain time T (i.e., ONL is not holding stock till the end when a forced sale happens), also define $H_{\ell+1} = [b_{\ell+1}, s_{\ell+1}]$ for the trivial trade $(b_{\ell+1}, s_{\ell+1})$. Each N_i (except possibly the last one, $N_{\ell+1}$) has length exactly t_1. The last non-holding period $N_{\ell+1}$ may be shorter than t_1 because day T arrives first, or that it is empty since the algorithm holds until day T. Let h_i be the highest price during H_i.

Lemma 3 (i) *For any two days x and y in the same H_i, where $x < y$, we have $p(y) \ge p(x)/r$. In particular, $p(s_i) \ge h_i/r$.*
(ii) *Without loss of generality we can assume OPT would not buy or sell strictly within any N_i.*
(iii) *For any i, $p(b_{i+1}) \le p(s_i)r$.*

Proof (i) This is because at day y the highest price seen thus far is at least $p(x)$, and so the stop loss threshold is at least $p(x)/r$. If $p(y)$ is below this value then a sale would have been triggered earlier, ending the holding period. (Note that this assumes the continuous time model where there is a precise moment when the price is exactly $p(x)/r$.) As a direct consequence, $p(s_i) \ge h_i/r$ since s_i is the last day of this holding period. If the sale is a forced sale at day T, it means it has not reached the threshold and so the inequality still holds.
(ii) We apply Lemma 2 to transform the prices so that within each N_i it goes up by α every day. Recall that ONL always waits t_1 days before the next purchase,

and thus is independent of price changes during this period. The transformation also changes the earlier prices (from day 0 up to just before this N_i), and in general price changes might affect the holding periods of the algorithm; however, since all the prices are changed proportionally, ONL's buying and selling actions during that period are unaffected. Thus Lemma 2 can indeed be applied.

It is easy to see that OPT only buys at local minima and sells at local maxima. As a result of the above transformation, it would not buy or sell strictly within any N_i.

(iii) Since each non-holding period lasts at most t_1 days, the price rises by at most a factor of α^{t_1} over each non-holding period. Note that this is in fact true even without the transformation in (ii). (With the transformation, the stronger condition $p(b_{i+1}) = p(s_i)r$ holds for all except possibly the last non-holding period, but this is not needed for the proof.)

□

Theorem 7 *Algorithm 4* has competitive ratio $\alpha^{T(2k\log\beta)/((k+1)\log\beta+k\log\alpha)}$.

Proof Consider an OPT trade (b^*, s^*). By Lemma 3(ii), we know that b^* or s^* would not fall within any N_i. Suppose b^* falls within H_u and s^* falls within H_v, where $v \geq u$. Its gain g^* is equal to

$$g^* = \frac{p(s^*)}{p(b^*)} \leq \frac{h_v}{p(b_u)/r} \leq \frac{p(s_v)r}{p(b_u)/r} = \frac{p(s_v)}{p(b_u)}r^2 \tag{3}$$

where the inequalities are due to Lemma 3(i). Note that if $u = 0$, then $p(b^*) = p(b_u)$ and thus a factor of r can be removed from the above bound; similarly if $v = \ell+1$ then $p(s_v) = h_v$ and another factor of r can be removed. Thus, if we define an indicator variable I_i which is 0 if $i = 0$ or $i = \ell + 1$ and 1 otherwise, then

$$g^* \leq \frac{p(s_v)}{p(b_u)}r^{I_u+I_v}$$

Then

$$\begin{aligned} g^* &\leq \frac{p(s_v)}{p(b_v)}\frac{p(b_v)}{p(s_{v-1})}\frac{p(s_{v-1})}{p(b_{v-1})}\frac{p(b_{v-1})}{p(s_{v-2})}\cdots\frac{p(s_{u+1})}{p(b_{u+1})}\frac{p(b_{u+1})}{p(s_u)}\frac{p(s_u)}{p(b_u)}r^{I_u+I_v} \\ &\leq \frac{p(s_v)}{p(b_v)}r\frac{p(s_{v-1})}{p(b_{v-1})}r\ldots r\frac{p(s_u)}{p(b_u)}r^{I_u+I_v} \\ &= r^{v-u+I_u+I_v}\prod_{i=u}^{v} g_i \end{aligned} \tag{4}$$

where $g_i = p(s_i)/p(b_i)$ is the gain of the i-th ONL trade. The second inequality is due to Lemma 3(iii).

If no other OPT trade sells during H_u or buys during H_v, we can associate this OPT trade with these $v-u+1$ ONL trades $(b_u, s_u), \ldots, (b_v, s_v)$. No other OPT trades would be associated with these ONL trades. But if other OPT trades fall within H_u or H_v then this cannot be done directly. Instead we consider groups of OPT trades, determined as

follows. Suppose there are two successive OPT trades (b_1^*, s_1^*) and (b_2^*, s_2^*) such that the earlier one sells at the same holding period as the later one buys, i.e., b_1^* is in H_u, s_1^* and b_2^* are in the same H_v, and s_2^* is in H_w, where $u \le v \le w$. Then $p(b_2^*) \ge p(s_1^*)/r$ due to Lemma 3(i), and so $(p(s_2^*)/p(b_2^*))(p(s_1^*)/p(b_1^*)) \le (p(s_2^*)/p(b_1^*))r$, and by losing a factor of at most r we can replace the two OPT trades with one (b_1^*, s_2^*).

By repeatedly applying the argument, we can partition the time horizon into disjoint parts: in part j, OPT has x_j trades $(b_{j,1}^*, s_{j,1}^*), \ldots, (b_{j,x_j}^*, s_{j,x_j}^*)$, $b_{j,1}^*$ falls within some H_{u_j} and s_{j,x_j}^* falls within some H_{v_j}, and no other OPT trade sells during H_{u_j} or buys during H_{v_j} (otherwise they would have been merged into this as well). ONL made a total of $y_j = v_j - u_j + 1$ trades here, each of gain $g_{j,i}$. The x_j OPT trades, each of gain $g_{j,i}^*$, are merged into one trade $(b_{j,1}^*, s_{j,x_j}^*)$ of gain g_j^*, such that $g_j^* \ge \prod_i g_{j,i}^* / r^{x_j - 1}$. Since $g_j^* \le r^{v_j - u_j + I_{u_j} + I_{v_j}} \prod_{i=u_j}^{v_j} g_{j,i}$ (from (4)), we have

$$\prod_i g_{j,i}^* \le r^{x_j - 1} r^{v_j - u_j + I_{u_j} + I_{v_j}} \prod_{i=u_j}^{v_j} g_{j,i} = r^{x_j + y_j} r^{I_{u_j} + I_{v_j} - 2} \prod_{i=u_j}^{v_j} g_{j,i}$$

Over all partitions, therefore, the overall gains ratio between OPT and ONL is $\prod_j \left(\prod_i g_{j,i}^* / \prod_{i=u_j}^{v_j} g_{j,i} \right) \le \prod_j r^{x_j + y_j} r^{I_{u_j} + I_{v_j} - 2}$.

If none of u_j or v_j involve the two trivial holding periods H_0 and $H_{\ell+1}$, then all I_{u_j} and I_{v_j} are equal to 1, and $\sum_j x_j \le k$, $\sum_j y_j \le \ell$. Hence

$$\prod_j r^{x_j + y_j} r^{I_{u_j} + I_{v_j} - 2} \le r^{k+\ell} r^{1+1-2} \cdots r^{1+1-2} \le r^{2k}$$

If one trivial holding period is involved, e.g., $b_{1,1}^*$ falls into H_0 and thus $u_1 = 0$, then $I_{u_1} = 0$ and $\sum_j x_j \le k$, $\sum_j y_j \le \ell + 1$. Then

$$\prod_j r^{x_j + y_j} r^{I_{u_j} + I_{v_j} - 2} \le r^{k+\ell+1} r^{0+1-2} r^{1+1-2} \cdots r^{1+1-2} \le r^{2k}$$

Finally if both trivial holding periods are involved, i.e., $b_{1,1}^*$ falls into H_0 and the last sale falls into $H_{\ell+1}$, then

$$\prod_j r^{x_j + y_j} r^{I_{u_j} + I_{v_j} - 2} \le r^{k+\ell+2} r^{0+1-2} r^{1+1-2} \cdots r^{1+1-2} r^{1+0-2} \le r^{2k}$$

□

4.3 Rounding Issues

It is unlikely that t_1 and t_2 as defined in (2) will be integers, especially if $\alpha \ne \beta$, and the arguments in the previous two subsections have to be adapted when we move away

from the idealized continuous model back to the discrete model. Here we show how they can be done and how much they affect the competitive ratio.

For Algorithm 3, it can be modified, for example, by setting the total length of all k holding periods to be $\lfloor kt_2 \rfloor$, to be spread as evenly as possible between them. The $k+1$ non-holding periods then have total length $T - \lfloor kt_2 \rfloor$, again to be spread as evenly as possible. Then the length of a non-holding period is at least

$$\tilde{t_1} \geq \left\lfloor \frac{T - \lfloor kt_2 \rfloor}{k+1} \right\rfloor \geq \frac{T - kt_2}{k+1} - 1 = t_1 - 1$$

and the length of a holding period is at least

$$\tilde{t_2} \geq \left\lfloor \frac{\lfloor kt_2 \rfloor}{k} \right\rfloor = \lfloor t_2 \rfloor \geq t_2 - 1$$

Note that the latter follows from the identity

$$\left\lfloor \frac{\lfloor x/m \rfloor}{n} \right\rfloor = \left\lfloor \frac{x}{mn} \right\rfloor$$

if n is an integer; see [7].

ONL's return now becomes $1/\beta^{\lfloor kt_2 \rfloor}$. As for OPT, similar to the proof of Theorem 6, if it holds in k upward periods then its return is at most $\alpha^{T-\lfloor kt_2 \rfloor - \tilde{t_1}}$; if one of its trades spans a downward period then its return is at most $\alpha^{T-\lfloor kt_2 \rfloor}/\beta^{\tilde{t_2}}$. Thus the competitive ratio is $\alpha^{T-\lfloor kt_2 \rfloor}\beta^{\lfloor kt_2 \rfloor}/\min(\alpha^{\tilde{t_1}}, \beta^{\tilde{t_2}})$. Comparing with the continuous case where the competitive ratio is equal to $\alpha^{T-kt_2-t_1}\beta^{kt_2}$, the discrete one is worse by a factor of at most

$$\begin{aligned}
\frac{\alpha^{T-\lfloor kt_2 \rfloor}\beta^{\lfloor kt_2 \rfloor}/\min(\alpha^{\tilde{t_1}}, \beta^{\tilde{t_2}})}{\alpha^{T-kt_2-t_1}\beta^{kt_2}} &= (\alpha/\beta)^{kt_2-\lfloor kt_2 \rfloor}\alpha^{t_1}/\min(\alpha^{\tilde{t_1}}, \beta^{\tilde{t_2}}) \\
&\leq (\alpha/\beta)^{kt_2-\lfloor kt_2 \rfloor}\alpha^{t_1}/\min(\alpha^{t_1-1}, \beta^{t_2-1}) \\
&= (\alpha/\beta)^{kt_2-\lfloor kt_2 \rfloor}\max(\alpha, \beta)
\end{aligned}$$

which is at most α^2/β if $\alpha > \beta$ and β otherwise.

For the lower bound in Theorem 5, if an algorithm cannot choose the t_i values to make all the cases in (1) equal due to the integrality constraints, the lower bound may then become slightly stronger, but the original result stands.

For Algorithm 4, we make the following changes to the algorithm:

(1) Round t_1 up to the nearest integer. Let $\tilde{t_1}$ be the rounded value. The algorithm will now wait $\tilde{t_1}$ days after a sale before the next purchase. Lemma 3(iii) becomes $p(b_{i+1}) \leq p(s_i)\alpha^{\tilde{t_1}}$.
(2) The selling threshold is still h_i/r. However, as the prices are not continuous, the price may jump from just above h_i/r (when we would not sell) to below h_i/r (when the sale is triggered). Thus the selling price may not be exactly h_i/r, but it must still be within a factor β of h_i/r. Lemma 3(i) becomes $p(y) \geq p(x)/(r\beta)$.

Previously, in the continuous case, $r = \alpha^{t_1} = \beta^{t_2}$. Now, let $\tilde{r} = \max(\alpha^{\tilde{t_1}}, \beta r)$. Then effectively, any occurrence of r in the proof can be replaced and upper bounded by $\tilde{r}$, and the proof will still work. For example, formula (3) now becomes

$$g^* = \frac{p(s^*)}{p(b^*)} \leq \frac{h_v}{p(b_u)/(r\beta)} \leq \frac{p(s_v)r\beta}{p(b_u)/(r\beta)} = \frac{p(s_v)}{p(b_u)}(r\beta)^2 \leq \frac{p(s_v)}{p(b_u)}\tilde{r}^2 \quad (5)$$

and formula (4) now becomes

$$g^* \leq \frac{p(s_v)}{p(b_v)}\alpha^{\tilde{t_1}}\frac{p(s_{v-1})}{p(b_{v-1})}\alpha^{\tilde{t_1}}\ldots\alpha^{\tilde{t_1}}\frac{p(s_u)}{p(b_u)}(r\beta)^{I_u+I_v} \leq \tilde{r}^{v-u+I_u+I_v}\prod_{i=u}^{v} g_i \quad (6)$$

Thus the rest of the proof remains unchanged; we simply need to replace r with $\tilde{r}$. The competitive ratio changes from r^{2k} to $\tilde{r}^{2k}$. Since $\tilde{r} = \max(\alpha^{\tilde{t_1}}, \beta r) \leq \max(\alpha, \beta)r$, the competitive ratio at most worsens by a factor of $\max(\alpha, \beta)^{2k}$.

We note that the above analyses are rather crude and it may well be possible to make improvements; we leave this as future work.

5 Conclusion

There are still many possible directions that can be explored. Some examples include randomized algorithms, allowing short-selling, trading two or more stocks, or allowing the money to be split into a small number of bets (e.g., the investor can sell half of its shares first, and let the other half potentially earn more profit). The last one models the intermediate case between the splittable and unsplittable versions of the problem.

References

1. Chen, G.-H., Kao, M.-Y., Lyuu, Y.-D., Wong, H.-K.: Optimal buy-and-hold strategies for financial markets with bounded daily returns. SIAM J. Comput. **31**(2), 447–459 (2001)
2. Chin, F.Y.L., Fu, B., Guo, J., Han, S., Hu, J., Jiang, M., Lin, G., Ting, H.F., Zhang, L., Zhang, Y., Zhou, D.: Competitive algorithms for unbounded one-way trading. Theor. Comput. Sci. **607**(1), 35–48 (2015)
3. Clemente, J., Hromkovic, J., Komm, D., Kudahl, C.: Advice complexity of the online search problem. In: Proceedings of 27th International Workshop on Combinatorial Algorithms (IWOCA), pp. 203–212, (2016)
4. Damaschke, P., Ha, P.H., Tsigas, P.: Online search with time-varying price bounds. Algorithmica **55**, 619–642 (2009)
5. El-Yaniv, R., Fiat, A., Karp, R.M., Turpin, G.: Optimal search and one-way trading online algorithms. Algorithmica **30**(1), 101–139 (2001)

6. Glynn, P.W., Iglehart, D.L.: Trading securities using trailing stops. Manag. Sci. **41**(6), 1096–1106 (1995)
7. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics. Addison-Wesley, Reading, MA (1994)
8. Kao, M.-Y., Tate, S.R.: On-line difference maximization. SIAM J. Discrete Math. **12**(1), 78–90 (1999)
9. Lorenz, J., Panagiotou, K., Steger, A.: Optimal algorithms for k-search with application in option pricing. Algorithmica **55**(2), 311–328 (2009)
10. Mohr, E., Ahmed, I., Schmidt, G.: Online algorithms for conversion problems: a survey. Surv. Oper. Res. Manag. Sci. **19**, 87–104 (2014)
11. Schmidt, G., Mohr, E., Kersch, M.: Experimental analysis of an online trading algorithm. Electron. Notes Discrete Math. **36**, 519–526 (2010)
12. Schmidt, G.: Competitive analysis of bi-directional non-preemptive conversion. J. Comb. Optim. **34**, 1096–1113 (2017)
13. Schroeder, P., Schmidt, G., Kacem, I.: Optimal on-line algorithms for bi-directional non-preemptive conversion with interrelated conversion rates. In: Proceedings of 4th IEEE Conference on Control, Decision and Information Technology, pp. 28–33 (2016)
14. Zhang, W., Xu, Y., Zheng, F., Dong, Y.: Optimal algorithms for online time series search and one-way trading with interrelated prices. J. Comb. Optim. **23**, 159–166 (2012)