Algorithmica
DOI 10.1007/s00453-007-9156-9

Online Search with Time-Varying Price Bounds

Peter Damaschke · Phuong Hoai Ha · Philippas Tsigas

Received: 30 May 2007 / Accepted: 3 December 2007

Abstract Online search is a basic online problem. The fact that its optimal deterministic/randomized solutions are given by simple formulas (however with difficult analysis) makes the problem attractive as a target to which other practical online problems can be transformed to find optimal solutions. However, since the upper/lower bounds of prices in available models are constant, natural online problems in which these bounds vary with time do not fit in the available models.

We present two new models where the bounds of prices are not constant but vary with time in certain ways. The first model, where the upper and lower bounds of (logarithmic) prices have decay speed, arises from a problem in concurrent data structures, namely to maximize the (appropriately defined) freshness of data in concurrent objects. For this model we present an optimal deterministic algorithm with competitive ratio $\sqrt{D}$, where D is the known duration of the game, and a nearly-optimal randomized algorithm with competitive ratio $\frac{\ln D}{1+\ln 2-\frac{2}{D}}$. We also prove that the lower bound of competitive ratios of randomized algorithms is asymptotically $\frac{\ln D}{4}$.

The second model is inspired by the fact that some applications do not utilize the decay speed of the lower bound of prices in the first model. In the second model, only the upper bound decreases *arbitrarily* with time and the lower bound is constant. Clearly, the lower bound of competitive ratios proved for the first model holds

P. Damaschke · P. Tsigas
Department of Computer Science and Engineering, Chalmers University of Technology, Götenborg, Sweden

P. Damaschke
e-mail: ptr@cs.chalmers.se

P. Tsigas
e-mail: tsigas@cs.chalmers.se

P.H. Ha (✉)
Department of Computer Science, Faculty of Science, University of Tromso, Tromsø, Norway
e-mail: phuong@cs.uit.no

also against the stronger adversary in the second model. For the second model, we present an optimal randomized algorithm. Our numerical experiments on the freshness problem show that this new algorithm achieves much better/smaller competitive ratios than previous algorithms do, for instance 2.25 versus 3.77 for $D = 128$.

Keywords Search algorithms · Online algorithms · Competitive analysis · Game theory

1 Introduction

The online search problem is a fundamental on-line problem [5, 7]. In the problem, a player searches for the maximum (minimum) price in a sequence of prices that unfolds daily and varies unpredictably. For each day i, the player observes a price p_i and must decide whether to accept this price or to wait for a better one. The game ends when the player accepts a price, which is also the result. The *randomized* online search can be considered as a deterministic one-way trading problem in which a player is exchanging her initial wealth in one currency (e.g. dollar) to another currency (e.g. yen) so as to maximize her final payoff while the price (or exchange rate from dollar to yen) varies unpredictably. There is a known simple transformation of (randomized) online search to (deterministic) one-way trading [2, 5–7]: The budget corresponds to probability 1 and exchanging some fraction of money (in deterministic one-way trading) means to stop the game with exactly that probability (in randomized online search) [5, 7]. El-Yaniv et al. suggested optimal solutions for several slight variants of the online search problem in which the upper/lower bounds of prices are constants and known a priori to the player [5–7]. Since the optimal solutions for these variants are quite simple computationally (that is, the amounts to exchange are easy to compute, but the analysis is quite sophisticated), practical issues can be transformed to online search in order to find optimal solutions [8–10]. Chen et al. suggested another variant of the problem in which the next price r' depends on the current price r in a geometric manner: $r/\theta \leq r' \leq r\theta$, where $\theta > 1$ is the daily fluctuation ratio [2]. For the variant, these authors presented closed-form solutions to the competitive ratio.

However, there are still natural problems that cannot be transformed to any of these variants in a tight way. One of them comes from the freshness problem of concurrent data objects. Freshness is a significant property of shared data objects and has achieved great concerns in databases [3, 13, 17] as well as in caching systems [14–16]. It is used to evaluate how recent the data (of a shared object) returned by a read operation is when the object is continuously written by concurrent write operations. Note that the object can be a set of registers that need to be read and written as a whole atomically; the read/write operations in this case are multi-register read/write operations [11]. From the correctness point of view, the read-operation is allowed to return any value concurrently written by one of the write-operations. However, from the application point of view the read-operation is preferred to return the latest/freshest one of the valid values, especially in reactive/detective systems. For instance, monitoring sensors continuously concurrently input data via a shared object and the processing unit periodically reads the data to make the system react accordingly. In such systems, the freshness of data influences how fast the system reacts to environment changes.

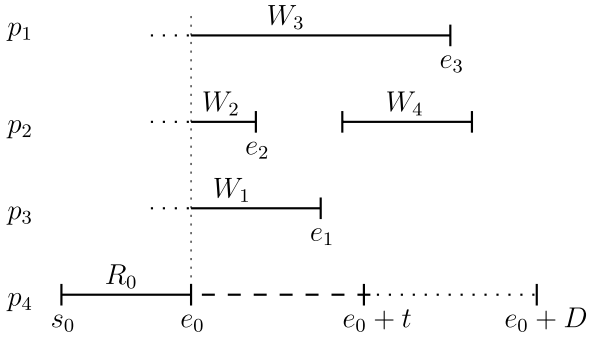

Fig. 1 An illustration for the freshness problem

Figure 1 illustrates the freshness problem of concurrent data objects. The time axis is from left to right. A read-operation R_0 is running concurrently with three write-operations W_1, W_2 and W_3 and all the read/write operations access the same data object. Each operation i starts at a point s_i and takes effect at a point e_i (i.e. linearization point [12]). The value returned by R_0 will become fresher if more endpoints e_i appear in the interval $[s_0, e_0]$. In the illustration, if R_0 delays its endpoint e_0 by a time t, the value R_0 returns at the time-point $e'_0 = e_0 + t$ will be fresher than that returned at e_0 since there are two more endpoints e_2 and e_1 included in the interval $[s_0, e'_0]$. Note that either of e_0 and e'_0 is valid to be R_0's linearization point without changing R_0's semantics. However, the delay also makes R_0 respond more slowly. The freshness problem is to find an optimal delay t to maximize R_0's freshness value f_t, for instance $f_t = \frac{|e_t|}{t}$ where $|e_t|$ is the number of new endpoints gained by the delay t. In other words, if there are many points that are valid to be R_0's linearization point, which point should R_0 choose? The challenge is that endpoints of concurrent write-operations will appear unpredictably and R_0 must decide on-the-fly whether to accept the current freshness value f_t or to wait for a better one and lose the current one. The freshness problem is a restricted case of online search [7] since the upper/lower bounds of freshness values vary with time. Obviously, applying the traditional online search algorithms [7] on the freshness problem will not give an optimal result.

For more information on the freshness problem and how to model it as an online search problem with time-varying bounds, the reader is referred to [4].

1.1 Our Contributions

Motivated by the freshness problem, we consider the following new online search models in which the lower/upper bounds of prices vary with time instead of being constants. Time-varying price bounds can be also an interesting problem in the financial setting where the time-varying bounds are expected to be determined sometimes by microeconomic results.

The first model is a continuous model on time interval $[1, D]$ with known duration D and prices r_t at time t that fulfill

$$r_t \leq r_u \cdot \frac{u}{t} \tag{1}$$

for any times $t < u$ and

$$\frac{M}{D} \le r_t \le \frac{M}{t}, \quad \forall t \in [1, D], \tag{2}$$

where M is the maximal allowed price at $t = 1$ and known a priori to the player. With known D and M, the player needs to search for the maximum price, which is unfolded on-the-fly over a continuous time interval. Given a current price, the player has to decide whether to accept this price or to wait for a better one. The game ends when the player accepts a price, which is also the result. The model is motivated by the constraints on the freshness values in the freshness problem (cf. [4]). Particularly, inequalities (1) and (2) come from the fact that the number of endpoints, which is $t \cdot r_t$ at time t in the model, always increases with time.

The second model is time-discrete with known duration D and known upper/lower bounds of prices r_t: $m \le r_t \le M(t)$, where *the upper bound $M(t)$ is a decreasing function of time t*. Prices are unfolded on-the-fly over a discrete time interval and when a new price is observed, a new period starts. Note that the second model "contains" the first one when D is large. Any instance of the first model can be transformed to the second model where $m = \frac{M}{D}$ and $M(t) = \frac{M}{t}$.

For the first model we present an optimal deterministic algorithm with competitive ratio $\sqrt{D}$, where D is the known duration of the game, and a nearly-optimal randomized algorithm with competitive ratio $\frac{\ln D}{1+\ln 2-\frac{2}{D}}$. We also prove that no randomized algorithm can achieve a competitive ratio better/less than, asymptotically, $(\ln D)/4$. Our analysis can be easily extended to more general models in which t in inequalities (1) and (2) is replaced by t^S, where S is any constant:

$$r_t \le r_u \cdot \frac{u^S}{t^S}, \quad \forall t < u,$$

$$\frac{M}{D} \le r_t \le \frac{M}{t^S}, \quad \forall t \in [1, D].$$

For the second model, we suggest an optimal randomized algorithm with competitive ratio

$$c^* = \max_{1<t\le D} \left\{ c \;\middle|\; c = t\left(1 - \left(\frac{c-1}{\frac{M(t)}{m} - 1}\right)^{1/t}\right)\right\}. \tag{3}$$

Since this expression is hard to evaluate analytically, we add some numerical results for $M(t) = \frac{M}{t}$ and $m = \frac{M}{D}$ in order to compare the competitive ratios in the case of the freshness problem at the end of Section 3. We compare the new algorithm with the previous algorithms, which are devoted to the previous models but now apply to the new more restricted model. The results show that the new algorithm achieves much better/smaller competitive ratios than the previous algorithms do, for instance 2.25 versus 3.77 for $D = 128$.

As for the relation between the new competitive ratio and the new lower bound, since the adversary in the second model is less restricted (or stronger) than one in the first model, the lower bound of competitive ratios $(\ln D)/4$ holds also for the second model. We chose to consider the stronger adversary in the second model because the

randomized algorithm considers to stop with a certain probability only at increasing prices, thus it does not even exploit the limited decay speed of prices in the first model. Our numerical experiments suggest that the randomized algorithm with the new competitive ratio c^* is still not too far from the lower bound, despite the stronger adversary. This is explained by the observation that slowly increasing prices seem to be the worst case for the online player. In the lower-bound proof we consider continuous time only because this simplifies the arguments. Note that when D is large, the difference between continuous and discrete time models disappears.

The rest of this paper is organized as follows. Section 2 presents an optimal deterministic algorithm and a nearly-optimal randomized algorithm for the first model. A lower bound of competitive ratios for randomized algorithms in the first model is also presented in this section. Section 3 presents an optimal randomized algorithm for the second model. Section 4 concludes the paper with some remarks.

2 The First Model

In this section, we present an optimal deterministic algorithm and a nearly-optimal randomized algorithm for the first model. We also present a lower bound on competitive ratios of randomized algorithms.

We repeat our first online search model. With a duration D and the maximal allowed price M at time $t = 1$ known a priori to the player, and prices r_t at time t that fulfill:

$$r_t \le r_u \cdot \frac{u}{t}, \quad \forall t < u, \tag{4}$$

$$\frac{M}{D} \le r_t \le \frac{M}{t}, \quad \forall t \in [1, D] \tag{5}$$

the player needs to search for the maximum price, which is unfolded on-the-fly over a continuous time interval $[1, D]$. Given a current price, the player has to decide whether to accept this price or to wait for a better one. The game ends when the player accepts a price, which is also the result.

We define some *notations*, which are adapted to the geometry of the problem. In particular, we work with logarithmic axes for both price and time. In the following, let *log-price* be the logarithm of price. We normalize the log-price axis in such a way that price $\frac{M}{D}$ corresponds to point 0 and price M to point $\ln D$. This is convenient because now, going one unit up the log-price axis increases the price by factor e (Euler's number).

We introduce some *parameters* characterizing the status of the game between online player and adversary at any moment. Let t denote the time, initially $t = 1$. The horizontal axis is for the logarithm of time t. We normalize it so that $t = 1$ corresponds to point 0 and $t = D$ corresponds to point $\ln D$.

Defining f: Let f be the maximum log-price the adversary has already reached during the history of the game, $f(t) = \max_{u \le t} \ln(r_u) - \ln(\frac{M}{D})$.

Defining g: Let g be the maximum log-price the adversary can still achieve before the game ends, $g(t) = \ln(\frac{M}{t}) - \ln(\frac{M}{D})$. In more detail, g corresponds to price M/t

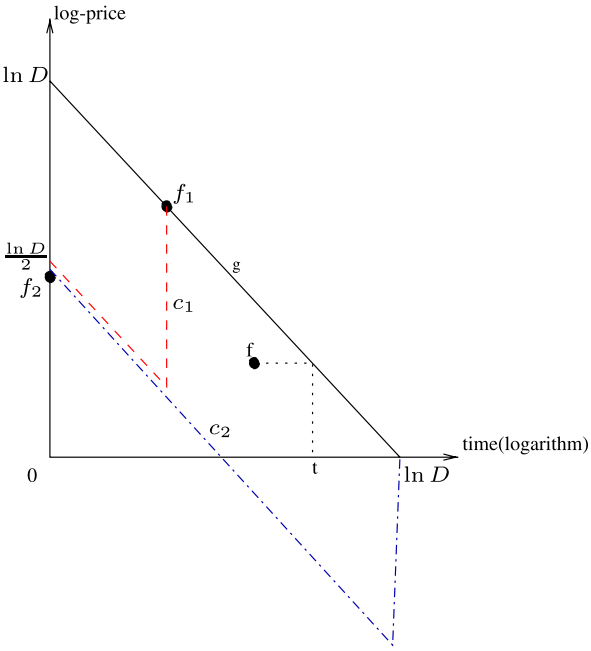

Fig. 2 Illustration for the proof of Theorem 2

at time t, unless equality $f = M/t$ is already reached, in which case we have $g = f$. When $g = f$, the game is over without loss of generality, by the following argument: The adversary herself cannot get higher than g and would therefore decrease the price as quickly as possible, in order to make the player's position worst. Hence the player should stop immediately, exchanging all her remaining money. (The dotted polyline in Fig. 2 illustrates the case $f = g(t)$ in which the player should stop at time t.) We also remark that, on a logarithmic time axis, g decreases at unit speed, starting at point $\ln D$, where $\ln D$ is an arbitrary scaling factor that will make the calculations simpler (cf. Fig. 2).

Defining c: Let c denote the current log-price, as determined by the adversary. Hence an instance of the problem is given by c as a function of time. Note that on a logarithmic time axis, parameter c can decrease at most at unit speed (like g), but c can jump upwards arbitrarily as long as $c \leq g$ (according to the model). Below we always refer to the logarithmic time axis, without explicit mention.

2.1 Optimal Deterministic Algorithms

Unlike the original online search model [7], our first model has more restrictions on the adversary (cf. inequalities (4) and (5)). The restrictions make the adversary in the new model weaker than the adversary in the original model, and intuitively the player in the new model should benefit from this. However, we will prove that this is not the case for deterministic algorithms (cf. Theorem 2).

From inequality (5), we have the upper/lower bounds on the prices r:

$$\frac{M}{D} \leq r \leq M.$$

Inspired by an online search algorithm called *reservation price policy* for the original unrestricted model [7], we have the following deterministic algorithm for the new restricted model:

Deterministic Algorithm RPP: The player accepts the first price that is not smaller than $r^* = \frac{M}{\sqrt{D}}$.

Following the analysis of the reservation price policy in [7], we have Theorem 1. The proof is presented here to make this paper self-contained.

Theorem 1 *The suggested deterministic algorithm is competitive with competitive ratio* $c = \sqrt{D}$.

Proof Let r^* be the threshold for accepting a price and r_{max} be the highest price chosen by the adversary. The player waits for a value $r_t \geq r^*$. If such a price appears in the interval D, the player accepts it and returns it as the result. Otherwise, when waiting until the time D, the player must accept the value $r_{min} = \frac{M}{D}$.

Case 1: If the player chooses a large value as r^*, the adversary will choose $r_{max} < r^*$, causing the player to wait until the time D and accept the value $r_{min} = \frac{M}{D}$. The competitive ratio in this case is $c_1 = \frac{r_{max}}{M/D} < \frac{r^*}{M/D}$.

Case 2: If the player chooses a small value as r^*, the adversary will place r^* at a time $t = 1$, causing the player to accept the value and stop. Right after that, the adversary lifts the price to the maximal value M. The competitive ratio in this case is $c_2 = \frac{M}{r^*}$.

Therefore, the player chooses r^* so as to make $c_1 = c_2$, which results in $r^* = \frac{M}{\sqrt{D}}$ and the competitive ratio $c = c_1 = c_2 = \sqrt{D}$. □

We now prove that no deterministic algorithm can do better for the new model.

Theorem 2 *The optimal deterministic competitive ratio is asymptotically* (*subject to lower-order terms*) $\sqrt{D}$.

Proof We only need to show an adversary strategy that enforces the claimed competitive ratio. Our logarithmic coordinates make the argument rather simple: The adversary starts with $c = \frac{\ln D}{2}$. Then she decreases c at unit speed until the player stops. Immediately after this moment, c jumps to g if $c > 0$ at the stop time (Case 1), otherwise c keeps on decreasing at unit speed (Case 2). Clearly, we have constantly $g - c = \frac{\ln D}{2}$ until the stop time. Let p be the player's value of log-price. In Case (1) we finally get $f = g$, hence $f - p = g - c = \frac{\ln D}{2}$ (cf. the dashed polyline c_1 in Fig. 2). In Case (2), f has still its initial value $\frac{\ln D}{2}$ whereas $p \leq 0$, hence $f - p \geq \frac{\ln D}{2}$ (cf. the line c_2 in Fig. 2). Thus the competitive ratio is at least $e^{\frac{\ln D}{2}} = \sqrt{D}$. □

We have shown that a deterministic player cannot benefit from the constraints on the behavior of price in time (compared to the unrestricted online search problem).

2.2 Competitive Randomized Algorithms

Next we present a randomized algorithm for the new online search model, against the oblivious adversary [1]. It achieves a competitive ratio $c = \frac{\ln D}{1+\ln 2-\frac{2}{\sqrt{D}}}$.

As discussed in the previous section, the new model is a restricted case of online search. In the model, the adversary's payoff is the highest price ever reached. The player's payoff is the price at the moment when she stops. Note that for a player running a randomized strategy, the payoff is the expected price, with respect to the distribution of stops resulting from the strategy and input.

We shall make use of a known simple transformation of (randomized) online search to (deterministic) one-way trading [7]: The player has some budget of money (e.g. in dollars) and she wants to exchange (e.g. from dollars to yen) while the prices may vary over time. Her goal is to maximize her gain. The transformation is given as follows: The budget corresponds to probability 1, and exchanging some fraction of money (in deterministic one-way trading) means to stop the game with exactly that probability (in randomized online search). Note that a deterministic algorithm for online search has to exchange all money at *one* point in time.

For the new model, it is possible to apply a well-known competitive randomized algorithm EXPO [7]. Applying the EXPO algorithm on the new model achieves a competitive ratio $\ell\frac{2^{\ell-1+1/\ln 2}}{2^{\ell-1+1/\ln 2}-\frac{1}{\ln 2}}$, where $\ell = \log_2 D$. That means for the new model our randomized algorithm is better than the EXPO algorithm by a constant factor $\frac{1+\ln 2}{\ln 2}$ when D becomes large.

We start with some conventions. Without loss of generality, the amount of money of the online player is set to $\ln D$. We imagine that all money, both the exchanged and non-exchanged money, is distributed on the log-price axis. Moreover, the player can temporarily have some of the money in her *pocket*. Formally, the allocation of money on the log-price axis at any time is described by two non-negative real *density functions* A and B, defined as follows. $A(x)$ is the density of already exchanged money in point x of the log-price axis. That is, $A(x)$ represents the accumulated density of the money being exchanged when point $c = x$ has been visited. $B(x)$ is the density of not yet exchanged money in point x of the log-price axis. Unlike A, function B is just an imaginary construct used for accounting purposes, to keep track of the money not yet exchanged. The algorithm's job is to transfer probability mass from B to A without loss. The way this is done by the algorithm depends on the way the adversary moves the price. This will be detailed in the proof below.

The *value* of every piece of *exchanged* money is the price corresponding to its position on the log-price axis. Therefore the total value of exchanged money, which is the integral over the value-by-density product, gives the player's payoff in the game.

Theorem 3 *There is a randomized algorithm for the freshness problem with expected competitive ratio* $\frac{\ln D}{1+\ln 2-\frac{2}{\sqrt{D}}}$ *against an oblivious adversary.*

Proof First we describe how the algorithm works, using the status parameters and density functions introduced earlier.

0. In the beginning, we put the not-yet-exchanged money on interval $[0, \ln D]$ of the log-price axis, with density 1. Formally: $B(x) = 1$ for all $x \in [0, \ln D]$.
1. As g decreases with unit speed, the player takes the money above g away ($B(x) := 0$ for all $x > g$) and puts it in her pocket.
2. Whenever f increases, the player also takes the money below f away ($B(x) := 0$ for all $x < f$) and puts it in her pocket.
3. The player *continuously* locates exchanged money on the log-price axis, observing the following rule: *Whenever you have money in your pocket and c is positive and decreasing, and $A(c) < 2$ holds at the current c, then set $A(c) := 2$, by taking money from your pocket.*
4. If the game is over (because of $f = g$) and not all money is exchanged yet, the player puts the rest r of her money on the current c.

Note that the adversary must set the final c nonnegative due to the lower bound on prices c (cf. inequality (5)). It is obvious from steps 0-2 that we always have $B = 1$ on $[f, g]$, and $B = 0$ outside $[f, g]$.

The idea of the new algorithm is to guarantee some concentration of exchanged money immediately below the final f, not too far from f. Locating much money instantaneously is risky because c may jump upwards, and then this money has little value compared to the adversary's. On the other hand, since c can decrease only with limited speed, the player may completely abstain from exchanging money as long as c is increasing, and wait until c goes down again. This makes the main difference between the new algorithm and the previous algorithms [7]: the former utilizes the lower bound and exchanges money when the price is decreasing, whereas the latter exchanges money when the price is increasing.

After these intuitive thoughts, we start proving the performance guarantee now.

First of all, the player is always able so set $A(c) := 2$ at the current c (as demanded in 3): Note that the player can use the one unit of money from B she obtains per time unit from the part above the falling g, and also the money from B she got directly from the current c when f was going upwards.

In the following, δ_x refers to Dirac's delta function, i.e., the distribution which has infinite density at a single point x, density 0 elsewhere, and integral 1 on any interval that contains x.

By construction, the player produces a density function A that is constantly 2 on certain intervals and constantly 0 outside these intervals, plus a single component $r\delta_c$. In the following, a *full interval* is a maximal interval where $A = 2$ holds, or the final point c where r units of money is located, and a *gap* is a maximal interval between two full intervals, note that we have $A = 0$ in a gap. We make some crucial observations regarding the final situation: (1) We have $A(x) = 2$ for all $x \in (c, f]$, or it holds $c = f$; (2) The gaps have total length at most r.

These claims follow easily from the algorithm: (1) Either c begins decreasing, starting from the last f, and $A(c)$ is set to 2 all the time when $c > 0$ (as we saw above), or the final c equals the final f. (2) Whenever f went upwards, the player has taken from B the money corresponding to the increase of f, and later she has transferred it to A and located it at the same points again. Hence, we have $A = 0$ only on intervals not "visited" again by c, and the money taken from B on these intervals is still in the player's pocket and thus contributes to r.

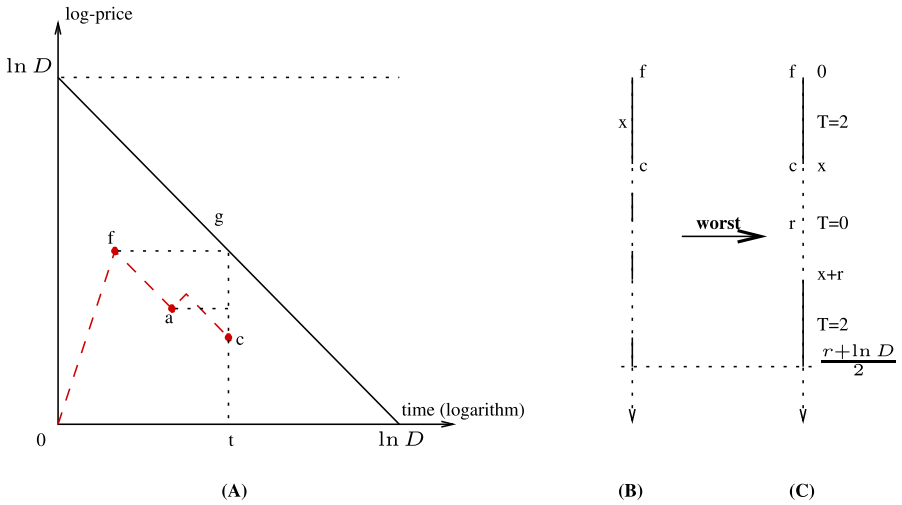

Fig. 3 Illustration for the randomized algorithm

Figure 3A illustrates the player's behavior. The dashed line represents a variation of c in a game; point c is the final value of c when the game ends, i.e. $f = g(t)$. For all values v on the log-price axis between f and a and between a and c, the player sets $A(v) = 2$.

Using (1), (2) we now analyze the payoff the player can guarantee herself. Remember that the value of exchanged money located on the log-price axis decreases exponentially. Let $x = f - c$ (final values). Both r and x depend on the input, i.e., the behavior of c in time. The total amount of money is fixed, it equals $\ln D$. For any fixed r, x, the worst case is now that the gaps sum up to the maximum length r and are as high as possible on the log-price axis, that is, immediately below point c, because in this case all exchanged money outside $[c, f]$ has the least possible value. That is, $[c - r, c]$ is the only gap.

Figure 3C illustrates the worst case corresponding to an instance 3B, where solid lines represent full intervals. In the worst case, the adversary shifts all solid lines except for $[c, f]$ to the lowest possible position so as to minimize the player's payoff. Let the lower solid line be $[x + r, y]$ on the axis going down from f. The total amount of money in the worst case is

$$2x + r + 2(y - (x + r)) = \ln D \quad \Rightarrow \quad y = \frac{r + \ln D}{2}.$$

Hence, a lower bound on the player's payoff, divided by the value at f, is given by

$$\min_{r,x}\left(2\int_0^x e^{-t}dt + re^{-x} + 2\int_{x+r}^{(r+\ln D)/2} e^{-t}dt\right),$$

where we started integration (with $t = 0$) at point f and go down the log-price axis (cf. Fig. 3C). Verify that, in fact, $\int T\,dt = \ln D$. The above expression evaluates to

$$2 + (r - 2 + 2e^{-r})e^{-x} - 2e^{-(r+\ln D)/2} > 2 + (r - 2 + 2e^{-r})e^{-x} - 2/\sqrt{D}.$$

For any fixed x, this is minimized if $2e^{-r} = 1$, that is, $r = \ln 2$. Since now $r - 2 + 2e^{-r} = \ln 2 - 2 + 1 < 0$, the worst case is $x = 0$, which gives $1 + \ln 2 - 2/\sqrt{D}$. The adversary earns $\ln D$ times the value at f. □

2.3 The Lower Bound of Competitive Ratios for Randomized Algorithms

In this section, we present the lower bound of competitive ratios for randomized online search (or deterministic one-way trading) in the first model.

Theorem 4 *The competitive ratio for the first model is* $\Omega(\ln D)$. *More precisely, for every* $\epsilon > 0$ *there exists* D_ϵ *such that for all* $D > D_\epsilon$, *no algorithm for the first model can achieve a competitive ratio better than* $(\ln D)/4 - \epsilon$.

Proof First we consider a modified game where negative log-prices (prices below M/D) are permitted. In the final analysis we will adjust this assumption and consider the original game where log-prices must be nonnegative all the time.

(1) We describe two *primitives* that our adversary will use in her strategy below.

Firstly, the adversary can move not-yet-exchanged money to the player's pocket. Secondly, the adversary can modify A in a special way: She can move pieces of exchanged money on the log-price axis, and multiply their amount (!) by $1/e$ if they are moved one unit upwards (or multiply their amount by e if they are moved one unit downwards). The effect is that the total value of exchanged money, i.e., the player's payoff, remains the same. This manipulation will only be used for keeping function A simple.

According to the one-way trading setting, the player can only place money continuously on the log-price axis at the current point c and thus increase $A(c)$.

Finally, let s be some parameter between 0 and 1 that we fix later.

(2) Now we are prepared to specify the *adversary's strategy*. Let h be another time-varying log-price value with $f \le h \le g$. At the beginning we set $c = f = h = s \ln D$ and $g = \ln D$. As an invariant, we keep $B(x) = 1/(1 - s)$ for all $x \in [h, g]$, and $B(x) = 0$ for all $x \notin [h, g]$ all the time. (Since the total amount of money is $\ln D$, this is consistent in the beginning.) Other not-yet-exchanged money is always moved to the player's pocket. As long as the player keeps some handed-out money in her pocket, the adversary decreases c at unit speed, and increases h at unit speed. Whenever the player runs out of money, the adversary sets immediately $c := h$. If the player even "raises a loan", i.e., takes extra money from $[h, g]$ for immediate exchange, which is of course allowed by the game, the adversary increases h accordingly (going to higher c), so as to keep $B(x) = 1/(1 - s)$ for all $x \in [h, g]$. Note that $f = h$ holds after every such increase. As soon as $h = g$ is reached, all money not yet exchanged is in the player's pocket, hence equality $h = g$ is kept until the game ends with $f = h = g$.

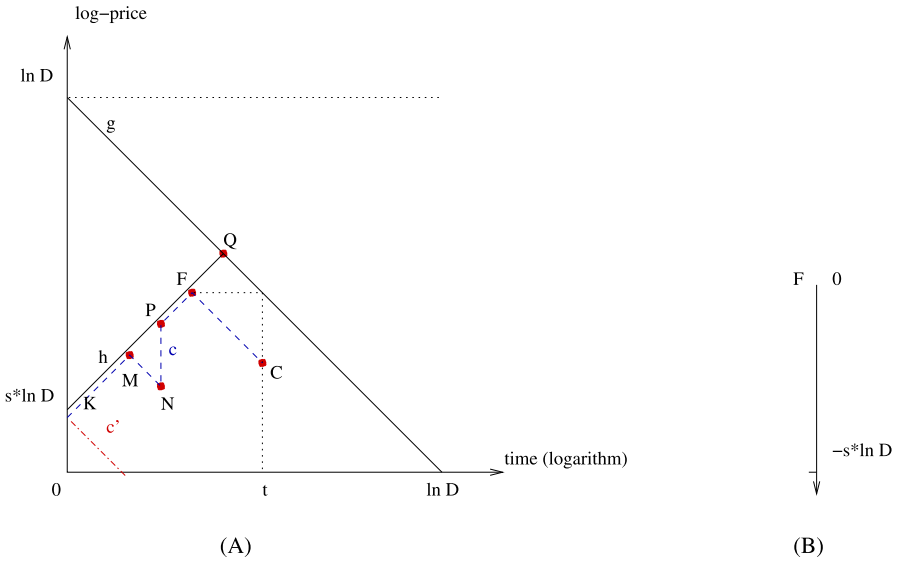

Fig. 4 Illustration for the proof of Theorem 4

Figure 4 illustrates the adversary's strategy. Up to point M the player exchanges all money in her pocket and thus the adversary increases c at unit speed. At point M, the player keeps some handed-out money in her pocket and thus the adversary decreases c at unit speed. At point N, the player spends all money in her pocket and the adversary in response sets immediately $c := h$, i.e. moves c upwards to point P. When $h = g$ at point Q, h, together with g, goes down at unit speed. The game ends when the player has exchanged all money at C, where $f = g = h$.

(3) We have not yet specified how the adversary handles function A. (Remember that the adversary may modify function A describing the exchanged money which the player has put on the log-price axis.) This is not part of the strategy, but of the analysis.

We show the following *invariants*: The adversary can always hold $A(x) = 0$ for all $x > f$, $A(x) \leq 4/(1-s)$ for all $x \in [c, f]$, and even $A(x) \leq 2/(1-s)$ for all $x < f$ when $f = h$.

This is vacuously true in the beginning, since no money has been exchanged yet. As long as the player exchanges all her money immediately, we have $c = f = h$, increasing at unit speed, so that the player receives $2/(1-s)$ units of money per time unit. Hence $A(x) \leq 2/(1-s)$ holds at all points x passed by c on the log-price axis (e.g. from point K up to point M in Fig. 4). If the player withholds some money in her pocket, c is going downwards while h is going upwards, both at unit speed. In such periods, the player also receives $2/(1-s)$ units of money per time unit. The player gets the highest possible payoff if she exchanges this money immediately (up to an infinitesimal rest). Any portion exchanged with delay can be moved upwards and reduced by the adversary. This way the adversary can always hold $A(x) \leq 4/(1-s)$ for all $x \in [c, f]$.

For instance, let $A_{KM}(x)$ and $A_{MN}(x)$ be the amounts of money put on each point x of the path going from K up to M and of the path going from M down to N, respectively. We have $A_{KM}(x) \le 2/(1-s)$ and $A_{MN}(x) \le 2/(1-s)$. Therefore, every point x on the log-price axis between M and N has $A(x) = A_{KM}(x) + A_{MN}(x) \le 4/(1-s)$.

Whenever the player decides to exchange all money she currently has in her pocket, we are back to case $c = h = f$. Along with this jump of c, the adversary can move exchanged money (e.g. A_{MN}) upwards to the gap between the previous f (point M) and new $f = h$ (point P), so that $A(x) \le 2/(1-s)$ is recovered at all points $x < f$. (The money fits because the gap between f and h had previously increased at unit speed.)

(4) The final *analysis* step of our proof compares the payoffs of player and adversary. We denote by F the final value of f. The invariants in (3) yield an upper bound on the player's payoff. More precisely, we only use that $A(x) = 0$ for $x > F$, and $A(x) \le 4/(1-s)$ for $x \le F$. For convenience we transform the log-price coordinates by $y := F - x$, that is, $y > 0$ denotes the distance from F, of points x below F (cf. Fig. 4B). Remember that the value of exchanged money decreases exponentially down the log-price axis, hence it decreases now exponentially with y. The adversary (as the optimal off-line player) would place all $\ln D$ units of money at F, that is, at point $y = 0$ in the new coordinates. The ideal case for the player would be that A has maximum density $4/(1-s)$ everywhere below F. Thus, an upper bound on the inverse competitive ratio is given by

$$\frac{4}{(1-s)\ln D}\left(\int_0^\infty e^{-y}dy\right) = \frac{4}{(1-s)\ln D}.$$

However, in the original game, $c \ge 0$ is required at every moment. We let our adversary behave as before, as if negative c were allowed, but actually she stays on $c = 0$ as long as the fictitious c is negative. Hence, all money exchanged at points $c < 0$ must be finally moved to point 0. Since the adversary would exchange all money at the final $F \ge s \ln D$, the best possible competitive ratio would be achieved when $F = s \ln D$ (i.e. the player always keeps some money in her pocket from the beginning, line c' in Fig. 4). Therefore, the ideal case for the player would be that $A(y) = 4/(1-s)$ for every point $0 \le y < s \ln D$ and the remaining $(1 - \frac{4s}{1-s}) \ln D$ units of money are exchanged at $y = s \ln D$ (cf. Fig. 4B). The upper bound on the inverse competitive ratio is now given by

$$\begin{aligned}
&\frac{1}{\ln D}\left(\frac{4}{1-s}\int_0^{s\ln D} e^{-y}dy + e^{-s\ln D}\left(1 - \frac{4s}{1-s}\right)\ln D\right) \\
&\quad = \frac{4}{(1-s)\ln D}\left(1 - \frac{1}{D^s}\right) + \frac{1}{D^s}\frac{1-5s}{1-s} \\
&\quad = \frac{4}{(1-s)\ln D} + O(1/D^s). \qquad (6)
\end{aligned}$$

For any fixed $s > 0$, and D large enough, $1/D^s$ becomes negligible compared to $1/\ln D$. Since this holds for arbitrarily small $s > 0$, the inverse competitive ratio comes arbitrarily close to $\frac{4}{\ln D}$. □

As a remark, we can apply the same analysis and get the same results for more general models with the bounds of prices decreasing like $1/t^S$, where S is any constant:

$$r_t \le r_u \cdot \frac{u^S}{t^S}, \quad \forall t < u,$$
$$\frac{M}{D} \le r_t \le \frac{M}{t^S}, \quad \forall t \in [1, D].$$

Note that the bound still decreases linearly in the logarithmic coordinates, the exponent is just turned into a factor, but scaling factors do not affect the competitive ratios.

3 Optimal Randomized Algorithms for the Second Model

We repeat the second online search model. With a known duration D and known upper/lower bounds of prices r_i: $m \le r_i \le M(i)$, where *the upper bound $M(i)$ is a decreasing function of time i*, the online player is searching for the maximum price. Prices are unfolded on-the-fly over a discrete time interval and when a new price is observed, a new period starts. Given a current price, the player has to decide whether to accept this price or to wait for a better one. The game ends when the player accepts a price, which is also the result.

There is a known simple transformation of (randomized) online search to (deterministic) one-way trading [7]: The budget corresponds to probability 1 and exchanging some fraction of money (in deterministic one-way trading) means to stop the game with exactly that probability (in randomized online search). Therefore, the randomized online search in this section is presented in the form of a deterministic one-way trading. The one-way trading model corresponding to the second model is as follows. With a known duration D and known upper/lower bounds of prices (yen per dollar) r_i: $m \le r_i \le M(i)$, where *the upper bound $M(i)$ is a decreasing function of time i*, the online player or player needs to trade her initial wealth W_0 given in dollar to yen efficiently. Prices are unfolded on-the-fly over a discrete time interval and when a new price is observed, a new period starts. Given a current price, the player has to decide how many of her dollars should be exchanged to yen at the current rate. Without loss of generality, assume that the player's initial wealth is one dollar, $W_0 = 1$.

In this section, we find a new optimal competitive ratio c^* for the second model, which is then used in the threat-based policy [7] to create an optimal algorithm for the second model. The algorithm is computationally simple: the amount of money to exchange in every step follows a simple formula. This makes the algorithm suitable for real applications.

Obviously, we can not achieve an optimal competitive ratio by directly applying the threat-based algorithms of the original models [7], where the upper bound of prices is a constant, to the new model, where the upper bound decreases with time. In the new model the adversary is clearly more restricted and thus the player should benefit from that. Hence, the analysis must adapt to the new model. We decide to

use the same flow for our analysis as that of original models because we think it is the most natural, but the technical details have to be adapted non-trivially at various places.

Particularly, our analysis must exploit the time-decreasing upper bound $M(i)$ to achieve better competitive ratio than that of the original analysis in which the upper bound is a constant, $M(1)$, and known a priori to the player. However, the challenge is that although the price r_i at a time $1 < i \leq D$ is less than $M(i)$, previous prices r_j, where $1 \leq j < i$, can be as high as $M(1)$. Since the prices the player has observed until a time i in the new model have the upper bound $M(1)$ as in the original model, simply adopting the original analysis for the new model cannot achieve any improvement on the competitive ratio.

In order to exploit the time-decreasing upper bound, our analysis utilizes the intrinsic feature of the threat-based policy: the policy considers the current price only if it is the highest seen so far. If the length k of *increasing* sequences of prices was known a priori to the player, $M(k)$ could be used as the actual upper bound of the corresponding *original* sequences of prices. Although the length k is controlled by the adversary and thus it is unknown to the player, the player only needs to know the maximum competitive ratio c^* achievable by the adversary to follow the threat-based policy (explanation is given below). Therefore, unlike the original analysis, our analysis first finds the maximum achievable competitive ratio $c^{(k)}$ for a known fixed k and then computes $c^* = \max_{1 \leq k \leq D} c^{(k)}$.

Let $k \leq D$ be the length of an *increasing* sequence of prices $m \leq p_1 < p_2 < \cdots < p_k \leq M(K)$, where K is the index of p_k in the original sequence, $k \leq K$. Since $M(i)$ is a decreasing function, $M(K) \leq M(k)$. This follows $m \leq p_1 < p_2 < \cdots < p_k \leq M(k)$.

For instance, if we have a sequence R of prices $\{1, 2, 4, 3, 7, 5, 6\}$ with $D = 7$, then the corresponding increasing sequence P of the prices is $\{1, 2, 4, 7\}$ with $k = 4$. Note that $R[5] = 7$ is included in the increasing sequence as $P[4]$ and $R[4] = 3$ is ignored since $R[3] > R[4]$. We have $P[4] \leq M(5) < M(4)$, since $P[4]$ corresponds to $R[5]$ (i.e. time/step 5) in the original sequence R and $M(i)$ is a decreasing function.

We will prove that the optimal competitive ratio c^* is

$$c^* = \max_{k=2\ldots D} \left\{ c \;\middle|\; c = k\left(1 - \left(\frac{c-1}{\frac{M(k)}{m} - 1}\right)^{1/k}\right)\right\}. \tag{7}$$

For each D given, we find the ratio c^* that satisfies (7) by simply computing c for each $k = 2, 3, \ldots, D$ and then choosing the maximum c as c^*. With the competitive ratio c^* found, the player follows the threat-based policy as in [7]:

- Consider exchanging dollar to yen at the current price only if it is the highest seen so far;
- When exchanging dollar, exchange *just enough* dollar at the current price to ensure the competitive ratio c^* even if the adversary then drops the rate to the minimum and keeps it there until the end.

The amount of dollar s_i that should be exchanged at the current price r_i is:

$$s_1 = \frac{1}{c} \cdot \frac{r_1 - mc}{r_1 - m} \quad \text{and} \quad s_i = \frac{1}{c} \cdot \frac{r_i - r_{i-1}}{r_i - m}, \quad \forall i \geq 2, \tag{8}$$

where $c = c^*$, $r_1 \geq mc^*$ and $r_{i-1} = \max_{1 \leq j \leq i-1} r_j$. If none of prices given is larger than mc^* until the end of game, the player can achieve the competitive ratio c^* by just exchanging all her dollars at the minimal price m at the end of game.

Since the threat-based algorithms is influenced only by the increasing sequence of given prices (cf. Rule 1), henceforth we consider threat-based algorithms on the increasing sequence $P = p_1, p_2, \ldots, p_k$, where $p_1 < p_2 < \cdots < p_k$. This implies

$$s_1 = \frac{1}{c} \cdot \frac{p_1 - mc}{p_1 - m} \quad \text{and} \quad s_i = \frac{1}{c} \cdot \frac{p_i - p_{i-1}}{p_i - m}, \quad \forall i \geq 2. \tag{9}$$

As we know, the competitive ratio c used in formula (9) is the target competitive ratio that the player tries to achieve. Obviously, the ratio cannot be an arbitrary small number. For instance, if the player chooses $c = 1$, she will exchange all her dollars at the first price r_1 since $s_1 = 1$. Then she will run out of dollars to exchange when the adversary issues a higher price r_2 in the next step and thus the player fails to achieve the competitive ratio $c = 1$. Therefore, the player following the threat-based policy achieves a competitive ratio only if the chosen ratio is large enough.

The following lemmas are inspired by the analysis of the original threat-based policy in [7].

Definition 1 Given a sequence R of prices, a threat-based algorithm A_c as defined by formula (9) with a ratio c, is *c-proper* with respect to R if

- the sum of daily exchanged dollars s_i computed by A_c is not larger than 1, the initial wealth (i.e. $\sum_{1 \leq i \leq k} s_i \leq 1$) and
- the resulting ratio of optimal offline return to online return $A_c(R)$ is not larger than c (i.e. $\frac{OPT(R)}{A_c(R)} \leq c$).

Lemma 1 *The threat-based algorithm following formula (9) with $c = c'$ is guaranteed to achieve the competitive ratio c' as long as there are enough dollars to exchange until the end of the game.*

Proof Let D_i and Y_i be the number of dollars and yen after the exchange at a step i of an increasing sequence of prices $P = p_1, p_2, \ldots, p_k$. We will prove that at any step i the algorithm always achieves a competitive ratio c' even if the adversary drops the rate to minimum in the next step and keeps it there until the end of a game, i.e.

$$\frac{p_i}{Y_i + mD_i} \leq c', \forall 1 \leq i \leq k. \tag{10}$$

We prove this lemma by induction. For the case $i = 1$, we have

$$\frac{p_1}{Y_1 + mD_1} = \frac{p_1}{p_1 s_1 + m(1 - s_1)} = c'.$$

Therefore, inequality (10) is correct for $i = 1$. Assume that the inequality is correct for $i = k - 1$, i.e.,

$$\frac{p_{k-1}}{Y_{k-1} + mD_{k-1}} \leq c'. \tag{11}$$

We will prove that the inequality is also correct for $i = k$, i.e.,

$$\frac{p_k}{Y_k + mD_k} \le c'. \tag{12}$$

Indeed, as long as there are enough dollars to exchange until the end, we have

$$\begin{aligned}\frac{p_k}{Y_k + mD_k} &= \frac{p_k}{(Y_{k-1} + s_k p_k) + m(D_{k-1} - s_k)} \\ &= \frac{p_k}{(Y_{k-1} + mD_{k-1}) + s_k(p_k - m)} \\ &\le \frac{p_k}{\frac{p_{k-1}}{c'} + \frac{p_k - p_{k-1}}{c'}} = c' \text{ (inequality (11) and formula (9)).}\end{aligned}$$ □

Lemma 2 *If A_c is c-proper with respect to an price sequence R, then for any $c' \ge c$, $A_{c'}$ is c'-proper with respect to R.*

Proof Let s_i and s'_i are amount of dollars converted on day/step i by A_c and $A_{c'}$, respectively. Following formula (9), we have

$$\begin{aligned}s_1 - s'_1 &= \frac{p_1}{p_1 - m}\left(\frac{1}{c} - \frac{1}{c'}\right) \ge 0, \\ s_i - s'_i &= \frac{p_i - p_{i-1}}{p_i - m}\left(\frac{1}{c} - \frac{1}{c'}\right) \ge 0, \quad \forall i \ge 2.\end{aligned}$$

Therefore, $\sum_i s'_i \le \sum_i s_i \le 1$ since A_c is c-proper. That means $A_{c'}$ satisfies the first condition of *c'-proper*. Moreover, from Lemma 1 it follows that $A_{c'}$ achieves a competitive ratio c' with respect to R, satisfying the second condition of *c'-proper*. □

Lemma 2 implies the following corollary.

Corollary 1 *If c^* is the maximum competitive ratio that is achievable by the adversary when the player follows the improved threat-based policy, A_{c^*} is c^*-proper regardless of the actual sequence of prices created by the adversary.*

Indeed, for each sequence R of prices, there exists the smallest competitive ratio c so that A_c is c-proper with respect to R. Since c^* is the *maximum* competitive ratio that is achievable by the adversary, $c^* \ge c$. From Lemma 2 it follows that A_{c^*} is c^*-proper with respect to R.

The main idea of the following analysis is to find the maximum competitive ratio c^* that is achievable by the adversary when the player follows the threat-based policy. The competitive ratio will then become the competitive ratio of the threat-based policy for the new one-way trading model and will be known by the player since it is computed using only known information: the duration D, the lower bound m and the upper bound function $M(i)$ (cf. equation (7)).

Henceforth, we assume that if k is fixed, k is known to the online player. Since it is trivial for the player to achieve the competitive ratio 1 when $k = 1$, without loss of generality we assume $k > 1$.

Lemma 3 *For fixed $k > 1$, the maximum competitive ratio that the adversary can achieve is*

$$c^{(k)} = \frac{kp^*}{km + (p^* - m)} \tag{13}$$

where p^* is the unique root in $[m, M(k)]$ of function

$$f(p) = (p-m)^{\frac{k}{k-1}} + mk(p-m)^{\frac{1}{k-1}} - m(k-1)(M(k)-m)^{\frac{1}{k-1}} = 0.$$

The maximum is achieved when

$$p_1 = p^* \quad \text{and} \quad \frac{p_i - p_{i-1}}{p_i - m} = 1 - \left(\frac{p^* - m}{M(k) - m}\right)^{1/(k-1)}, \quad \forall i \in [2, k]. \tag{14}$$

Proof Since the player spends his dollars only on the increasing sequence $p_1, p_2, \ldots, p_k$, we have $\sum_{i=1}^{k} s_i = 1$. Replacing s_i using formula (9), we obtain

$$\frac{1}{c}\frac{p_1 - cm}{p_1 - m} + \frac{1}{c}\sum_{i=2}^{k}\frac{p_i - p_{i-1}}{p_i - m} = 1$$

which results in a formula for c

$$c = 1 + \frac{p_1 - m}{p_1} \cdot \sum_{i=2}^{k}\frac{p_i - p_{i-1}}{p_i - m}. \tag{15}$$

On the other hand,

$$\begin{aligned}
\sum_{i=2}^{k}\frac{p_i - p_{i-1}}{p_i - m} &= \sum_{i=2}^{k}\left(1 - \frac{p_{i-1} - m}{p_i - m}\right) = k - 1 - \sum_{i=2}^{k}\frac{p_{i-1} - m}{p_i - m} \\
&\le k - 1 - (k-1)\left(\prod_{i=2}^{k}\frac{p_{i-1} - m}{p_i - m}\right)^{1/(k-1)} \\
&= (k-1)\left(1 - \left(\frac{p_1 - m}{p_k - m}\right)^{1/(k-1)}\right).
\end{aligned}$$

Equality occurs if and only if $\frac{p_{i-1}-m}{p_i - m} = (\frac{p_1 - m}{p_k - m})^{1/(k-1)}$ $\forall i \in [2, k]$.

Applying the inequality on (15) follows

$$c \le 1 + \frac{p_1 - m}{p_1} \cdot (k-1)\left(1 - \left(\frac{p_1 - m}{p_k - m}\right)^{1/(k-1)}\right).$$

Since the right side increases with p_k and $p_k \le M(k)$, we have

$$c \le 1 + \frac{p_1 - m}{p_1} \cdot (k-1)\left(1 - \left(\frac{p_1 - m}{M(k) - m}\right)^{1/(k-1)}\right). \tag{16}$$

Let $u = (p_1 - m)^{1/(k-1)} \ge 0$ and $v = (M(k) - m)^{1/(k-1)} > 0$, the right side becomes

$$c^{(k)}(p_1) = 1 + (k-1)\left(\frac{u^{k-1}v - u^k}{p_1 v}\right).$$

The derivative of $c^{(k)}(p_1)$ can be written as follows

$$\frac{dc^{(k)}(p_1)}{dp_1} = -\frac{u^k + mku - m(k-1)v}{p_1^2 v}.$$

Let $f(u) = u^k + mku - m(k-1)v$. Since (i) $f(u)$ increases with $u \ge 0$ and (ii) $f(0) = -m(k-1)v < 0$ as well as $f(v) = v^k + mv > 0$ for all $v > 0, k > 1$, equation $f(u) = 0$ has a unique positive root u^*. Moreover,

$$\frac{d^2c^{(k)}(u^*)}{dp_1} = -\frac{k((u^*)^{k-1} + m)}{p_1^2 v(k-1)(u^*)^{k-2}} < 0.$$

Therefore, $c^{(k)}(p_1)$ achieves its maximum at u^* or at $p_1 = p^* = (u^*)^{k-1} + m$.

From $f(u^*) = (u^*)^k + mku^* - m(k-1)v = 0$, we have

$$\frac{u^*}{v} = \frac{m(k-1)}{(u^*)^{k-1} + mk}, \quad \text{or}$$

$$\left(\frac{p^* - m}{M(k) - m}\right)^{1/(k-1)} = \frac{m(k-1)}{p^* + m(k-1)}.$$

Replacing p_1 by p^* in the right side of inequality (16) follows

$$\begin{aligned} c^{(k)}(p^*) &= 1 + \frac{p^* - m}{p^*} \cdot (k-1)\left(1 - \left(\frac{p^* - m}{M(k) - m}\right)^{1/(k-1)}\right) \\ &= 1 + \frac{p^* - m}{p^*} \cdot (k-1)\left(1 - \frac{m(k-1)}{p^* + m(k-1)}\right) \\ &= \frac{kp^*}{km + (p^* - m)}. \end{aligned}$$

Inequality (16) becomes:

$$c \le \frac{kp^*}{km + (p^* - m)}.$$

The equality occurs when

$$p_1 = p^* \quad \text{and} \quad \frac{p_i - p_{i-1}}{p_i - m} = 1 - \left(\frac{p^* - m}{M(k) - m}\right)^{1/(k-1)}, \quad \forall i \in [2, k]$$

where $p^* = (u^*)^{k-1} + m$ and u^* is the unique positive root of function $f(u) = u^k + mku - m(k-1)v = 0$. □

Lemma 4 *For fixed $k > 1$, $c^{(k)}$ is the unique root, c, of*

$$c = k \cdot \left(1 - \left(\frac{c-1}{\frac{M(k)}{m} - 1}\right)^{1/k}\right). \tag{17}$$

Proof The worst k-step sequence $P' = p'_1, \ldots, p'_k$ created by the adversary has the following properties from Lemma 3.

$$p'_1 = p^* \quad \text{and} \quad \frac{p'_i - p'_{i-1}}{p'_i - m} = 1 - \left(\frac{p^* - m}{M(k) - m}\right)^{1/(k-1)}, \quad \forall i \in [2, k]$$

where $c^{(k)} = \frac{kp^*}{km + (p^* - m)}$.

Since $s'_i = \frac{1}{c} \cdot \frac{p'_i - p'_{i-1}}{p'_i - m}, \forall i \in [2, k]$, we have $s'_2 = \cdots = s'_k$. On the other hand,

$$\begin{aligned} s'_1 &= \frac{1}{c} \cdot \frac{p'_1 - cm}{p'_1 - m} \\ &= \frac{1}{c^{(k)}} \cdot \frac{p^* - c^{(k)}m}{p^* - m} \\ &= \frac{p^* + m(k-1))}{kp^*} \cdot \frac{p^*(p^* - m)}{(p^* - m)(p^* + m(k-1))} \\ &= \frac{1}{k}. \end{aligned}$$

Since $\sum_{i=1}^{k} s'_i = 1$, we have

$$s'_1 = s'_2 = \cdots = s'_k = 1/k. \tag{18}$$

From formula (9), we have:

$$\begin{aligned} & s'_i = \frac{1}{c^{(k)}} \cdot \frac{p'_i - p'_{i-1}}{p'_i - m}, \forall i \geq 2 \\ \Rightarrow \quad & \frac{1}{k} = \frac{1}{c^{(k)}} \cdot \left(1 - \left(\frac{p'_1 - m}{M(k) - m}\right)^{1/(k-1)}\right) \quad \text{(Lemma 3)} \\ \Rightarrow \quad & c^{(k)} = k\left(1 - \left(\frac{p'_1 - m}{M(k) - m}\right)^{1/(k-1)}\right). \end{aligned} \tag{19}$$

From formula (9), we also have:

$$\frac{1}{k} = s'_1 = \frac{1}{c^{(k)}} \cdot \frac{p'_1 - c^{(k)}m}{p'_1 - m}$$

$$\Rightarrow \quad p_1' c^{(k)} - m c^{(k)} = p_1' k - k m c^{(k)}$$

$$\Rightarrow \quad p_1' = \frac{m c^{(k)} (k-1)}{k - c^{(k)}}. \tag{20}$$

Replacing p_1' in $c^{(k)}$ (see (19)) by the right side of (20) follows

$$c^{(k)} = k\left(1 - \left(\frac{mk(c^{(k)}-1)}{(k-c^{(k)})(M(k)-m)}\right)^{1/(k-1)}\right). \tag{21}$$

Expanding (21) follows

$$\frac{k - c^{(k)}}{k} = \left(\frac{mk(c^{(k)}-1)}{(k-c^{(k)})(M(k)-m)}\right)^{1/(k-1)}$$

$$\Rightarrow \quad \frac{k - c^{(k)}}{k}\left(\frac{mk(c^{(k)}-1)}{(k-c^{(k)})(M(k)-m)}\right) = \left(\frac{mk(c^{(k)}-1)}{(k-c^{(k)})(M(k)-m)}\right)^{k/(k-1)}$$

$$\Rightarrow \quad \left(\frac{c^{(k)}-1}{\frac{M(k)}{m}-1}\right)^{1/k} = \left(\frac{mk(c^{(k)}-1)}{(k-c^{(k)})(M(k)-m)}\right)^{1/(k-1)}. \tag{22}$$

Replacing the right side by the left side of (22) in (21), we obtain:

$$c^{(k)} = k \cdot \left(1 - \left(\frac{c^{(k)}-1}{\frac{M(k)}{m}-1}\right)^{1/k}\right).$$

Let

$$f(c) = c + k\left(\frac{c-1}{\frac{M(k)}{m}-1}\right)^{1/k} - k.$$

Since i) $f(c)$ is an increasing function with $c \geq 1$ and ii) $f(1) = 1 - k < 0$ as well as $\lim_{c \to +\infty} f(c) > 0$ due to $k > 1$ and $M(k) > m$, equation $f(c) = 0$ has a unique root $c \geq 1$. □

Up to this point, we have proved that for a fixed k, the maximum competitive ratio with respect to k that the adversary can achieve is the unique root of (17). Therefore, the maximum competitive ratio achievable by the adversary for the whole game with a known duration D, where k is any value in range $[1, D]$, is the maximum root of (17), where $k = 2, \ldots, D$.

Corollary 2 *The maximum competitive ratio c^* achievable by the adversary for the whole game with a known duration D is*

$$c^* = \max_{k=2\ldots D}\left\{c \;\middle|\; c = k\left(1 - \left(\frac{c-1}{\frac{M(k)}{m}-1}\right)^{1/k}\right)\right\}. \tag{23}$$

For each D given, we find the ratio c^* that satisfies (23) by simply computing c for each $k = 2, \ldots, D$ and then choose the maximum c as c^*. Since A_{c^*} is c^*-proper regardless of the actual sequence of prices created by the adversary (cf. Corollary 1), the player that uses the algorithm A_{c^*} is guaranteed to achieve the competitive ratio c^*.

Theorem 5 *The maximum competitive ratio c^* achievable by the adversary is the lowest possible competitive ratio for the one-way trading game with a time-decreasing upper bound.*

Proof Let *ALG* be any (deterministic) algorithm. Let k^* is the k that corresponds to c^* in (23). Let the worst sequence of prices be $R' = p'_1, p'_2, \ldots, p'_{k^*}, m_{k^*+1}, \ldots, m_D$, where p'_i are computed as those in the worst k-step sequence P' in the proof of Lemma 4. This also implies $p'_i < p'_{i+1}, i = 1, \ldots, k^* - 1$.

The adversary behaves as follows. At the first step, the adversary presents the price p'_1 to *ALG*. If *ALG* spends less than $1/k^*$ dollars at this rate, i.e. $s_1 < 1/k^*$, the adversary drops the price to the minimum m and keeps it there until the end of game. Since p'_1 was chosen so that $1/k^*$ is the minimum amount that needs to be converted at p'_1 so as to ensure the competitive ratio c^* even if the adversary subsequently drops the price to the minimum and keeps it there until the end, *ALG* with $s_1 < 1/k^*$ cannot achieve the competitive ratio c^*.

If *ALG* spends more than $1/k^*$ dollars at p'_1, the adversary presents p'_2 to *ALG* in the next step. At each step $i = 2, \ldots, k^*$, if amount of dollars *ALG* has exchanged so far is smaller than i/k^*, the adversary stops (i.e., drops the price to the minimum and keeps it there until the end of game). Otherwise, she presents the next price p'_{i+1} to *ALG*. Clearly the adversary will stop at a step $j \le k^*$ since the player's initial wealth is one dollar. We have

$$\sum_{i=1}^{j-1} s_i \ge \frac{j-1}{k^*},$$

$$\sum_{i=1}^{j} s_i < \frac{j}{k^*}$$

which follows $s_j < \frac{j}{k^*} - \frac{j-1}{k^*} = \frac{1}{k^*}$. Since p'_j was chosen so that $\frac{1}{k^*}$ is the minimum amount that needs to be exchanged at p'_j in order to ensure the competitive ratio c^*, *ALG* with $s_j < \frac{1}{k^*}$ cannot achieve the competitive ratio c^*.

That means *ALG* achieves the competitive ratio c^* only if *ALG* keeps $s_1 = s_2 = \cdots = s_{k^*} = \frac{1}{k^*}$, otherwise *ALG* will end up with a worse/higher competitive ratio. □

Theorem 5 implies the following corollary

Corollary 3 *The threat-based algorithm A_{c^*} is an optimal competitive algorithm for the one-way trading problem with a time-decreasing upper bound.*

Table 1 Numerical comparison of competitive ratios between different algorithms

	Value of D						
	4	8	16	32	64	128	⋯
Lower bound $\mathcal{C}$	1.15	1.23	1.31	1.39	1.48	1.57	⋯
Corresponding s	0.1823	0.1816	0.1810	0.1803	0.1795	0.1788	⋯
Original TBP c	1.48	1.85	2.28	2.75	3.25	3.77	⋯
Improved TBP c^*	1.17	1.33	1.52	1.73	1.99	2.25	⋯
Corresponding k	2	2	3	3	4	5	⋯

Table 1 shows the competitive ratios c^* and their corresponding k for each value of D (rows Improved TBP c^* and Corresponding k), which result from applying the improved threat-based policy to the freshness problem, that is, $M(t) = M/t$ and $m = M/D$. The competitive ratios c^* of the improved TBP turn out to be much better/smaller than those of the original threat-based policy (row Original TBP c) and close to the lower bounds (row Lower bound $\mathcal{C}$).

4 Conclusions

We have extended the set of practical problems that can be transformed to online search by presenting new online search models. Unlike the previous models, the new models allow the lower/upper bounds of prices to vary with time. The practicality of the new models has been demonstrated by their use to deal with the freshness problem of concurrent data objects [4].

For the first model we have presented an optimal deterministic algorithm with competitive ratio $\sqrt{D}$, where D is the known duration of the game, and a nearly-optimal randomized algorithm with competitive ratio $\frac{\ln D}{1+\ln 2-\frac{2}{D}}$. We have also proved that the lower bound of competitive ratios of randomized algorithms is asymptotically $\frac{\ln D}{4}$. A new technique that exploits the features of the new models has been used in the randomized algorithm and the proof of the lower bound.

Secondly, we have presented an optimal competitive algorithm against a stronger adversary, where the upper bound of prices decreases arbitrarily with time and the lower bound is constant. The second model is inspired by the fact that some applications do not utilize the decay speed of the lower bound of prices in the first model. Although the analysis flow in the new model looks similar to that in the previous models, the technical details have to be adapted non-trivially at various places. Obviously, we cannot achieve an optimal competitive ratio by directly applying the threat-based algorithms of the original models [7], where the upper bound of prices is constant, to the new model, where the upper bound decreases arbitrarily with time. In the latter the adversary is clearly more restricted and thus the player should benefit from that. Our numerical experiments on the freshness problem [4] have shown that the new algorithm obtains much better/smaller competitive ratios than the original threat-based algorithms do.

Acknowledgements The authors wish to thank the anonymous reviewers for their helpful and thorough comments on the earlier version of this paper.

References

1. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
2. Chen, G.H., Kao, M.Y., Lyuu, Y.D., Wong, H.K.: Optimal buy-and-hold strategies for financial markets with bounded daily returns. In: Proc. of ACM Symp. on Theory of Computing (STOC), pp. 119–128 (1999)
3. Cho, J., Garcia-Molina, H.: Synchronizing a database to improve freshness. In: SIGMOD '00: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 117–128 (2000)
4. Damaschke, P., Ha, P.H., Tsigas, P.: Competitive freshness algorithms for wait-free objects. In: Proceedings of the European Conference on Parallel Computing (Euro-Par). Lecture Notes in Computer Science, vol. 4128, pp. 811–820. Springer, Berlin (2006)
5. El-Yaniv, R.: Competitive solutions for online financial problems. ACM Comput. Surv. **30**(1), 28–69 (1998)
6. El-Yaniv, R., Fiat, A., Karp, R., Turpin, G.: Competitive analysis of financial games. In: Proc. of the 33rd Symp. on Foundations of Computer Science, pp. 327–333 (1992)
7. El-Yaniv, R., Fiat, A., Karp, R.M., Turpin, G.: Optimal search and one-way trading online algorithms. Algorithmica **30**(1), 101–139 (2001)
8. Ha, P.H., Papatriantafilou, M., Tsigas, P.: Efficient self-tuning spin-locks using competitive analysis. J. Syst. Softw. **80**(7), 1077–1090 (2007)
9. Ha, P.H., Papatriantafilou, M., Tsigas, P.: Self-tuning reactive diffracting trees. J. Parallel Distrib. Comput. **67**(6), 674–694 (2007)
10. Ha, P.H., Tsigas, P.: Reactive multi-word synchronization for multiprocessors. J. Instr. Lev. Parallelism **6**(Special issue) (2004). http://www.jilp.org/vol6/v6paper3.pdf
11. Herlihy, M.: Wait-free synchronization. ACM Trans. Program. Syst. **11**(1), 124–149 (1991)
12. Herlihy, M., Wing, J.: Linearizability: a correctness condition for concurrent objects. ACM Trans. Program. Lang. Syst. **12**(3), 463–492 (1990)
13. Kang, K.D., Son, S.H., Stankovic, J.A.: Managing deadline miss ratio and sensor data freshness in real-time databases. IEEE Trans. Knowl. Data Eng. **16**(10), 1200–1216 (2004)
14. Labrinidis, A., Roussopoulos, N.: Exploring the tradeoff between performance and data freshness in database-driven web servers. VLDB J. **13**(3), 240–255 (2004)
15. Li, W.S., Po, O., Hsiung, W.P., Candan, K.S., Agrawal, D.: Engineering and hosting adaptive freshness-sensitive web applications on data centers. In: WWW '03: Proceedings of the 12th International Conference on World Wide Web, pp. 587–598 (2003)
16. Ling, Y., Chen, W.: Measuring cache freshness by additive age. SIGOPS Oper. Syst. Rev. **38**(3), 12–17 (2004)
17. Pacitti, E., Simon, E.: Update propagation strategies to improve freshness in lazy master replicated databases. VLDB J. **8**(3–4), 305–318 (2000)