OPTIMAL BUY-AND-HOLD STRATEGIES FOR FINANCIAL MARKETS WITH BOUNDED DAILY RETURNS*

GEN-HUEY CHEN[†], MING-YANG KAO[‡], YUH-DAUH LYUU[§], AND HSING-KUO WONG[§]

Abstract. In the context of investment analysis, we formulate an abstract online computing problem called a *planning game* and develop general tools for solving such a game. We then use the tools to investigate a practical *buy-and-hold trading* problem faced by long-term investors in stocks. We obtain the unique optimal static online algorithm for the problem and determine its exact competitive ratio. We also compare this algorithm with the popular dollar averaging strategy using actual market data.

Key words. buy-and-hold trading problems, the balanced strategy, the dollar averaging strategy, online algorithms, competitive analysis, planning games, minimax theorem, linear programming, zero-sum two-person games.

AMS subject classifications. 5A15, 15A09, 15A23, 05A99, 60C05, 68R05, 90A09, 90A12, 90D10, 90D13

1. Introduction.

In an *online* problem, an *online* algorithm $\mathcal{B}$ is given one input at a time from a sequence of inputs. $\mathcal{B}$ takes an action on each input before seeing any remaining input. In contrast, an *off-line* algorithm sees the entire input sequence before it takes any action. Each action yields a positive *accumulation*. Let E denote the set of all admissible input sequences. Let $\mathcal{C}(\vec{e})$ denote the (expected) total accumulation of an online or offline algorithm $\mathcal{C}$ on $\vec{e} \in E$. Let $\mathcal{A}$ denote the *optimal* offline algorithm, i.e., one that produces the largest total accumulation on each admissible input sequence. In competitive analysis [4, 25, 27], $\mathcal{B}$'s performance is measured by its *competitive ratio*

$$\Upsilon_{\mathcal{B}} = \sup_{\vec{e} \in E} \frac{\mathcal{A}(\vec{e})}{\mathcal{B}(\vec{e})}. \tag{1.1}$$

The *online player* seeks to minimize this ratio by choosing a suitable $\mathcal{B}$ while the *adversary* attempts to maximize it by picking $\vec{e}$ after examining $\mathcal{B}$. This paper assumes that the adversary is *oblivious*, i.e., it fixes the input sequence before $\mathcal{B}$ performs any computation such as generating random bits.

A *planning game* is an abstract online problem where the length of the input sequence is fixed and known a priori to $\mathcal{B}$. This time horizon feature captures many important online problems including those for portfolio rebalancing [7, 8, 21], asset trading [2, 5, 11, 12], secretary selection [1, 6, 13, 16], and bipartite matching [15, 17]. A planning game is *finite* if the numbers of admissible sequences of actions and inputs are both finite; otherwise, it is *infinite*. A finite planning game corresponds to a linear programming problem, where an optimal randomized online algorithm corresponds to an optimal feasible solution. Consequently, we can show that the smallest competitive

*A preliminary version appeared in Proceedings of the 31st Annual ACM Symposium on Theory of Computing, 1999, pages 119–128.

[†]Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, ROC.

[‡]Department of Computer Science, Yale University, New Haven, CT 06520, USA, kao-ming-yang@cs.yale.edu. Research supported in part by NSF Grant CCR-9531028.

[§]Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, ROC. Research supported in part by NSC Grant 87-2416-H-002-031.

Exchange	Circuit Breaker	
	α^{-1}	β
Amsterdam	90%	110%
Bangkok	90%	110%
Paris	95%	110%
Taipei	93%	107%
Tel-Aviv	95%	110%
Tokyo	95%	130%
Vienna	95%	105%

FIG. 1.1. *Circuit breaker rules in various exchanges.*

ratio of any randomized online algorithm for such a game is the reciprocal of the value of the game as a zero-sum two-person game.

In this general optimization framework, we investigate the *buy-and-hold trading problem* defined as follows. An investor starts with some capital, which is normalized to 1 dollar, and trades it for a certain security over n days, which is referred to as the *investment horizon*. To avoid triviality, we assume $n \geq 2$. On each day, the security has only one *exchange rate*, i.e., the number of shares of the security which one unit of capital can buy. Upon seeing the exchange rate, the investor executes one transaction for that day and may trade all or part of the remaining capital. All the capital must be traded by the n-th day, and converting the acquired security back to capital is prohibited. The total *accumulation* of the investor is the number of shares of the security she accumulates at the end of the investment horizon. Note that the competitive ratio between the adversary's and the investor's accumulations is exactly the competitive ratio between the dollar values of the accumulations. This problem is faced by millions of investors who save for retirement purposes on a long-term basis; for instance, a widely popular security for today's investors would be a stock index fund.

We employ the *bounded daily return model*, in which the next day's exchange rate e' depends on the current day's exchange rate e with $e/\beta \leq e' \leq e\alpha$ for some fixed $\alpha, \beta > 1$. The values n, α, and $1/\beta$ are known a priori to the investor. We call α and $1/\beta$ the *daily return bounds*. Figure 1 gives some stock markets which enforce such ratios through circuit breakers. This model can also be regarded as an approximation to the geometric Brownian motion model used extensively in the finance community [3, 9, 14, 18, 20, 24, 26].

A *static* algorithm is an online algorithm for the buy-and-hold trading problem such that for $1 \leq i \leq n$, the (expected) amount of dollars invested by the algorithm on the i-th day is the same for all exchange rate sequences. A *dynamic* algorithm refers to any online algorithm for the problem, which is not necessarily static. The *static* buy-and-hold trading problem refers to the case of the problem where the investor can only use a static algorithm.

We prove that the smallest possible competitive ratio for any randomized or deterministic static algorithm is $\frac{n\alpha\beta-(n-1)(\alpha+\beta)+(n-2)}{\alpha\beta-1}$. We also obtain a deterministic static algorithm with this competitive ratio, called the *balanced strategy*, and prove that it is the only optimal deterministic static algorithm. In comparison, the popular dollar averaging strategy has a strictly greater competitive ratio and thus is not optimal. The balanced strategy is so simple that it can be executed even by those who are not mathematically sophisticated. Starting with one dollar initially, the algorithm invests $\frac{\alpha(\beta-1)}{n\alpha\beta-(n-1)(\alpha+\beta)+(n-2)}$ dollar on the first day, $\frac{(\alpha-1)\beta}{n\alpha\beta-(n-1)(\alpha+\beta)+(n-2)}$ dollar on the last day, and $\frac{(\alpha-1)(\beta-1)}{n\alpha\beta-(n-1)(\alpha+\beta)+(n-2)}$ dollar on each of the other days.

Previously, El-Yaniv *et al* [10–12] obtained optimal online algorithms for this unidirectional trading problem under the assumption that the daily exchange rates, instead of the daily returns, are between a pair of upper and lower bounds. Al-Binali [2] further studied the same setting in a framework of risk and reward [19]. Our model and that of El-Yaniv *et al* are each formulated for real but different regulations of stock and foreign currency markets. A subtle difference between these models is that their model fixes a upper bound and a lower bound on the daily exchange rates globally for the entire investment horizon, while our model sets new bounds dynamically every day. Interestingly, although this difference might seem minor, they give rise to mathematical results of very distinct flavors using significantly different techniques.

Section 2 discusses how to compute optimal randomized online algorithms for finite planning games. Section 3 uses the general analysis in §2 to derive the balanced strategy and compare it with the dollar averaging strategy. Section 4 concludes the paper with some open problems.

2. General analysis of finite planning games. A finite planning game G can be regarded as a finite zero-sum two-person game $\Gamma_H(m,n)$ defined as follows. For any integer $k>0$, let $Z_k=\{1,2,\ldots,k\}$. The maximizing player is the online player, whose pure strategies are the deterministic online algorithms $\mathcal{B}_i$ of G indexed with $i\in Z_m$. The minimizing player is the adversary of G, whose pure strategies are the input sequences $\vec{\sigma}_j$ of G indexed with $j\in Z_n$. The payoff matrix[1] H of $\Gamma_H(m,n)$ is defined by

$$H(i,j)=\frac{\mathcal{B}_i(\vec{\sigma}_j)}{\mathcal{A}(\vec{\sigma}_j)}>0,\quad i\in Z_m \text{ and } j\in Z_n. \tag{2.1}$$

Let $\Phi(Z_k)$ be the set of all probability density functions defined on Z_k. For $k=n$ or m, each $h\in\Phi(Z_k)$ is regarded as a point in the k-dimensional Euclidean space and represents a mixed strategy that applies the ℓ-th pure strategy indexed by Z_k with probability $h(\ell)$. By von Neumann's minimax theorem [22],

$$\begin{aligned}\max_{f\in\Phi(Z_m)}\min_{g\in\Phi(Z_n)}\sum_{i=1}^{m}\sum_{j=1}^{n}f(i)g(j)H(i,j) &= \min_{g\in\Phi(Z_n)}\max_{f\in\Phi(Z_m)}\sum_{i=1}^{m}\sum_{j=1}^{n}f(i)g(j)H(i,j)\\ &= \max_{f\in\Phi(Z_m)}\min_{j\in Z_n}\sum_{i=1}^{m}f(i)H(i,j) \qquad (2.2)\\ &= \min_{g\in\Phi(Z_n)}\max_{i\in Z_m}\sum_{j=1}^{n}g(j)H(i,j),\end{aligned}$$

which is called the *value* v^* of $\Gamma_H(m,n)$.

[1]In Equation (1.1), we use $\frac{\mathcal{A}(\vec{e})}{\mathcal{B}(\vec{e})}$ instead $\frac{\mathcal{B}(\vec{e})}{\mathcal{A}(\vec{e})}$ so that the competitive ratios of different online algorithms are greater than 1 and therefore are easier to distinguish visually in Figures 3.5 and 3.8. In contrast, in Equation (2.1), we choose $\frac{\mathcal{B}_i(\vec{\sigma}_j)}{\mathcal{A}(\vec{\sigma}_j)}$ instead of $\frac{\mathcal{A}(\vec{\sigma}_j)}{\mathcal{B}_i(\vec{\sigma}_j)}$ in order to simplify the linear algebra involved.

Let r^* be the smallest possible competitive ratio of any randomized online algorithm for G; i.e.,

$$r^* = \min_{f \in \Phi(Z_m)} \max_{j \in Z_n} \frac{\mathcal{A}(\vec{\sigma}_j)}{\sum_{i=1}^{m} f(i)\mathcal{B}_i(\vec{\sigma}_j)}.$$

A randomized online algorithm is *optimal* if its competitive ratio is r^*.

The next theorem relates G and $\Gamma_H(m,n)$.

THEOREM 2.1.

1. $r^* = \frac{1}{v^*}$.
2. *An optimal mixed strategy of the online player of* $\Gamma_H(m,n)$ *induces an optimal randomized online algorithm for G, and vice versa.*

Proof. This theorem follows from Equation (2.2). □

In light of Theorem 2.1, we use G and $\Gamma_H(m,n)$ interchangeably. A main purpose of this paper is to derive the exact value of r^* and an optimal randomized online algorithm for G. To do so by means of Theorem 2.1, the *primal* and *dual* problems of $\Gamma_H(m,n)$ or G are defined as follows:

Primal:		Dual:	
minimize	$x^T u_m$	maximize	$y^T u_n$
subject to	$x^T H \geq u_n^T$	subject to	$Hy \leq u_m$
	$x \geq 0$		$y \geq 0$

where u_k is the column vector of k copies of 1.

For each $j \in Z_n$, let H^j denote the j-th column of H. Moreover, let X and Y be the sets of feasible solutions to the primal and dual problems of G, respectively. Let $\bar{X}$ and $\bar{Y}$ be the sets of optimal feasible solutions to these problems. Let X^* and Y^* be the sets of optimal mixed strategies of the online player and the adversary, respectively.

The next lemma is useful for computing an optimal randomized online algorithm for G and its competitive ratio via linear programming.

LEMMA 2.2.

1. *For all nonzero* $x \in X$ *and* $y \in Y$, $\frac{x}{x^T u_m}$ *and* $\frac{y}{y^T u_n}$ *are mixed strategies for the online player and the adversary, respectively.*
2. $\min_{x \in X} x^T u_m = r^* = \max_{y \in Y} y^T u_n$.
3. $X^* = \frac{1}{r^*} \cdot \bar{X} \neq \emptyset$, *and* $Y^* = \frac{1}{r^*} \cdot \bar{Y} \neq \emptyset$.
4. *For each nonzero* $x \in X$, *if* $j \in Z_n$ *satisfies* $x^T H^j = \min_{\ell \in Z_n} x^T H^\ell$, *then* $\vec{\sigma}_j$ *is a worst-case input sequence for the online player's mixed strategy* $\frac{x}{x^T u_m}$.

Proof. This lemma follows from Theorem 2.1 and basics of linear programming [22]. □

The next fact is useful for analyzing the uniqueness of an optimal randomized online algorithm for G.

FACT 2.3 (see [23]). *For any* $x \in \bar{X}$ *and* $y \in \bar{Y}$, *x and y are extreme points of the convex polyhedra* $\bar{X}$ *and* $\bar{Y}$ *if and only if there is a square submatrix* $H' = (h_{ij})_{i \in I, j \in J}$ *of H for some* $I \subseteq Z_m$ *and* $J \subseteq Z_n$ *with the following properties:*

1. H' *is nonsingular.*
2. $\sum_{i \in I} h_{ij} x_i = 1$ *for all* $j \in J$.
3. $\sum_{j \in J} h_{ij} y_j = 1$ *for all* $i \in I$.
4. *For all* $i \notin I$, $x_i = 0$.
5. *For all* $j \notin J$, $y_j = 0$.

The next theorem combines Lemma 2.2 and Fact 2.3 for the case $m = n$.

THEOREM 2.4. *Assume that $m = n$ and H^{-1} exists. Let $x = (u_n^T H^{-1})^T$ and $y = H^{-1}u_n$. Further assume $x \geq 0$ and $y \geq 0$. Let $b = \frac{x}{x^T u_n}$.*

1. *Then, x and y are optimal feasible solutions to the primal and dual problems of G, respectively.*
2. *$\Upsilon_{\mathcal{B}} = r^* = x^T u_n$, where $\mathcal{B}$ is the randomized online algorithm corresponding to the online player's mixed strategy b; in other words, $\mathcal{B}$ is optimal for G.*
3. *For all $j = 1, \ldots, n$, $\frac{\mathcal{A}(\vec{\sigma}_j)}{\mathcal{B}(\vec{\sigma}_j)} = \Upsilon_{\mathcal{B}}$; i.e., $\mathcal{B}$ has the same performance relative to the adversary's on every input sequence.*
4. *If every component of x and y is strictly greater than 0, then x and y are the only optimal feasible solutions to the primal and dual problems of G, and consequently, $\mathcal{B}$ is the only optimal randomized online algorithm.*

Proof.

Statement 1. By direct verification, $x \in X$ and $y \in Y$. Then, since $x^T u_n = (y^T u_n)^T = y^T u_n$, by Lemma 2.2(2) $x \in \bar{X}$ and $y \in \bar{Y}$.

Statement 2. Note that $x^T u_n = r^*$ by Statement 1 and Lemma 2.2(2). Then, by Statement 1 and Lemma 2.2(3), b is an optimal mixed strategy of the online player. Thus, this statement follows from Theorem 2.1.

Statement 3. As pointed out in Statement 2, $x^T u_n = r^*$. By direct evaluation and Statement 2, $b^T H = \frac{1}{r^*} u_n = \frac{1}{\Upsilon_{\mathcal{B}}} u_n$. Then, this statement follows from the fact that by definition, the j-th component of $b^T H$ equals $\frac{\mathcal{B}(\vec{\sigma}_j)}{\mathcal{A}(\vec{\sigma}_j)}$.

Statement 4. To prove the uniqueness of $\mathcal{B}$, by Theorem 2.1(2) and Lemma 2.2(3), it suffices to show that $\bar{X}$ has a unique element. By basics of linear programming [22], $\bar{X}$ has only a finite number of extreme points, and any element in X is a finite convex combination of these extreme points. Thus, it suffices to show that x is the only extreme point of $\bar{X}$ as follows. Since H^{-1} exists, x and y are extreme points of $\bar{X}$ and $\bar{Y}$ by Fact 2.3 with $I = J = Z_n$. On the other hand, let z be any extreme point of $\bar{X}$. Since y is an extreme point of $\bar{Y}$, there is a square submatrix $H' = (h_{ij})_{i \in I, j \in J}$ of H such that z and y satisfy the five conditions in Fact 2.3. Since $y_j > 0$ for $j \in Z_n$, $J = Z_n$ by Condition 5. Since H' is square, $I = Z_n$ and $H' = H$. Then, by Condition 2, $z^T H = u_n^T$. Since $x^T H = u_n^T$, we have $z = x$ as desired. □

3. Optimal static algorithms.

This section applies the general tools in §2 to the static buy-and-hold trading problem to derive the smallest possible competitive ratio for static algorithms.

3.1. Notations.

As specified in §1, the investor in the buy-and-hold trading problem is given α, β, and n prior to an n-day investment horizon.

For $i \in Z_n$, let e_i be the given security's exchange rate on the i-th day of the investment horizon. Let e_0 be the exchange rate on the 0-th day, i.e., the day right before the investment horizon. Without loss of generality, we normalize e_0 to 1 to simplify the discussion. An *admissible* exchange rate sequence is any $\vec{e} = \langle e_1, e_2, \ldots, e_n \rangle$ where $e_i \in [e_{i-1}\beta^{-1}, e_{i-1}\alpha]$.

As in §1, let E denote the set of all admissible exchange rate sequences. Let $\mathcal{A}$ denote the optimal offline trading algorithm. Let $\mathcal{B}$ be the investor's online trading algorithm. After the adversary examines $\mathcal{B}$ but before the investor starts executing $\mathcal{B}$, the adversary picks and fixes some $\vec{e} \in E$. On the i-th day for $i \in Z_n$, upon seeing e_i, $\mathcal{B}$ decides the amount of remaining capital to be traded for shares of the security without knowing any future exchange rate, i.e., e_j with $j > i$. Note that $\mathcal{A}(\vec{e}) = \max_{1 \leq i \leq n} e_i$,

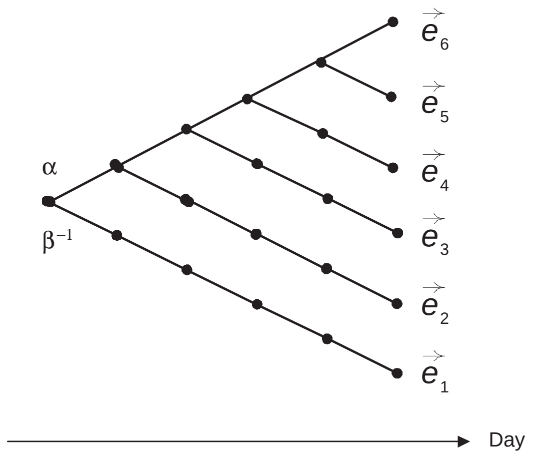

FIG. 3.1. *The downturns.*

and $\mathcal{B}(\vec{e}) = \sum_{i=1}^{n} a_i e_i$, where a_i is the (expected) amount of dollars invested by $\mathcal{B}$ on the i-th day and depends only on the current and past exchange rate $e_1, e_2, \ldots, e_i$.

For $i \in Z_n$, the algorithm $\mathcal{S}_i$ which trades the entire initial capital of one dollar on the i-th day is called the *trade-once algorithm on the i-th day.* Note that $\mathcal{S}_i$ is static and $\mathcal{S}_i(\vec{e}) = e_i$.

Let $\mathcal{S}$ be a randomized static algorithm. Let s_i be the expected amount of dollars invested by $\mathcal{S}$ on the i-th day. Note that $s_i \geq 0$ for all i and $\sum_{i=1}^{n} s_i = 1$. Thus, let $\mathcal{S}'$ be the deterministic static algorithm that invests s_i on the i-the day. Also, since the amounts $s_1, \ldots, s_n$ define a probability density function in $\Phi(Z_n)$, let $\mathcal{S}''$ be the randomized static algorithm that applies $\mathcal{S}_i$ with probability s_i.

LEMMA 3.1. *$\mathcal{S}$, $\mathcal{S}'$, and $\mathcal{S}''$ are equivalent in the sense that for all $\vec{e} \in E$, $\mathcal{S}(\vec{e}) = \mathcal{S}'(\vec{e}) = \mathcal{S}''(\vec{e})$.*

Proof. Straightforward. ◻

By Lemma 3.1, we identify $\mathcal{S}, \mathcal{S}'$, and $\mathcal{S}''$. Also, let r_{s}^* be the smallest competitive ratio for the static algorithms; then by Lemma 3.1,

$$r_{\mathrm{s}}^* = \inf_{f \in \Phi(Z_n)} \sup_{\vec{e} \in E} \frac{\mathcal{A}(\vec{e})}{\sum_{i=1}^{n} f(i)\, \mathcal{S}_i(\vec{e})}. \tag{3.1}$$

3.2. Reduction to finite games. The static buy-and-hold trading problem is an infinite planning game because the adversary has an infinite number of pure strategies, while by Lemma 3.1 the online player has n pure strategies $\mathcal{S}_i$. In order to use the tools in §2, we need to reduce the game to a finite one by eliminating the adversary's dominated pure strategies, i.e., non-worst-case exchange rate sequences, so that the remaining exchange rate sequences are finite in number.

For $j = 1, \ldots, n$, let

$$\vec{e}_j = \langle \overbrace{\alpha, \alpha^2, \ldots, \alpha^j}^{j}, \overbrace{\alpha^j \beta^{-1}, \alpha^j \beta^{-2}, \ldots, \alpha^j \beta^{j-n}}^{n-j} \rangle.$$

We call these n exchange rate sequences the *downturns*; see Figure 3.1 for an illustration.

LEMMA 3.2.

1. *Given a static algorithm $\mathcal{S}$, each $\vec{e} \in E$ is dominated by downturn $\vec{e}_j$, i.e., $\frac{\mathcal{A}(\vec{e})}{\mathcal{S}(\vec{e})} \le \frac{\mathcal{A}(\vec{e}_j)}{\mathcal{S}(\vec{e}_j)}$, where $e_j = \max_{i=1}^n e_i$.*
2. *The smallest competitive ratio for the static algorithms is*

$$r_s^* = \inf_{f \in \Phi(Z_n)} \max_{1 \le j \le n} \frac{\mathcal{A}(\vec{e}_j)}{\sum_{i=1}^n f(i)\, \mathcal{S}_i(\vec{e}_j)}.$$

3. *The static buy-and-hold trading problem can be regarded as a finite zero-sum two-person game $\Gamma_K(n,n)$ with the payoff matrix K defined by $K(i,j) = \alpha^{i-j}$ if $i \le j$ or β^{j-i} if $i > j$, i.e.,*

$$K = \begin{pmatrix} 1 & \alpha^{-1} & \alpha^{-2} & \cdots & \alpha^{1-n} \\ \beta^{-1} & 1 & \alpha^{-1} & \cdots & \alpha^{2-n} \\ \beta^{-2} & \beta^{-1} & 1 & \cdots & \alpha^{3-n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta^{1-n} & \beta^{2-n} & \beta^{3-n} & \cdots & 1 \end{pmatrix}.$$

Proof. Statement 1 follows from the fact that $\frac{e_j}{e_i} \le \alpha^{j-i}$ if $i \le j$ and $\frac{e_j}{e_i} \le \beta^{i-j}$ otherwise. Statement 2 follows from Equation (3.1) and Statement 1. For Statement 3, we let $\mathcal{S}_i$ be the online player's i-th pure strategy; and let $\vec{e}_j$ be the the adversary's j-th pure strategy. As in §2, the payoff matrix K is defined by $K(i,j) = \frac{\mathcal{S}_i(\vec{e}_j)}{\mathcal{A}(\vec{e}_j)}$. The statement then follows from the facts that $\mathcal{A}(\vec{e}_j) = \alpha^j$ and that $\mathcal{S}_i(\vec{e}_j) = \alpha^i$ if $i \le j$ or $\alpha^j \beta^{j-i}$ otherwise. ☐

In light of Lemma 3.2, an optimal mixed strategy of the online player of $\Gamma_K(n,n)$ corresponds to an optimal static algorithm. Thus, we next solve $\Gamma_K(n,n)$ to derive an optimal static algorithm.

3.3. Deriving an optimal static algorithm.

LEMMA 3.3. *For $n \ge 2$, $\det(K) = (1 - \alpha^{-1}\beta^{-1})^{n-1} > 0$.*

Proof. We use K_n to emphasize the dimension n of K. Let A_{ij} be the submatrix of K_n obtained by deleting row i and column j. To expand $\det(K_n)$ along the first row of K_n, observe that $A_{11} = K_{n-1}$. Furthermore, the first column of A_{12} equals β^{-1} times that of A_{11}, while the other columns of A_{12} equal the corresponding ones of A_{11}; thus $\det(A_{12}) = \beta^{-1}\det(A_{11})$. For $j = 3, \ldots, n$, $\det(A_{1j}) = 0$ because in A_{1j}, the first column equals β^{-1} times the second column. Hence, $\det(K_n) = \det(A_{11}) - \alpha^{-1}\det(A_{12}) = \det(K_{n-1}) - \alpha^{-1}\beta^{-1}\det(K_{n-1}) = (1-\alpha^{-1}\beta^{-1})\det(K_{n-1})$. The lemma immediately follows by induction on n. ☐

Let b^* be the column vector of n components defined by

$$b_i^* = \begin{cases} \frac{\alpha(\beta-1)}{n\alpha\beta-(n-1)(\alpha+\beta)+(n-2)} & i = 1; \\ \frac{(\alpha-1)(\beta-1)}{n\alpha\beta-(n-1)(\alpha+\beta)+(n-2)} & 1 < i < n; \\ \frac{(\alpha-1)\beta}{n\alpha\beta-(n-1)(\alpha+\beta)+(n-2)} & i = n. \end{cases} \tag{3.2}$$

Since $b^* > 0$ and ${b^*}^T u_n = 1$, b^* represents a mixed strategy of the online player. So let BAL denote the static algorithm which applies $\mathcal{S}_i$ with probability b_i^*; note that by Lemma 3.1, BAL is equivalent to the deterministic static algorithm which invests b_i^* dollars on the i-th day. Let c^* be the vector obtained by swapping the first and n-th components of b^*. Similarly, $c^* > 0$ and ${c^*}^T u_n = 1$, and we intend c^* to represent a mixed strategy of the adversary.

The next theorem analyzes BAL. In light of Statement 3 of the theorem, we call BAL the *balanced strategy.*

THEOREM 3.4. *Let* $r = \frac{n\alpha\beta-(n-1)(\alpha+\beta)+(n-2)}{\alpha\beta-1}$.

1. BAL *is an optimal static algorithm, and* $\Upsilon_{\mathrm{BAL}} = r^*_{\mathrm{s}} = r$.
2. BAL *is the only optimal static algorithm subject to the equivalence stated in Lemma 3.1.*
3. *For* $j = 1, \ldots, n$, $\frac{\mathcal{A}(\vec{e}_j)}{\mathrm{BAL}(\vec{e}_j)} = \Upsilon_{\mathrm{BAL}}$*; in other words,* BAL *has the same performance relative to the adversary's on every downturn.*

Proof. Let $\bar{b} = rb^*$, and $\bar{c} = rc^*$. By Lemma 3.3, K^{-1} exists. Below we prove $\bar{b}^T = u_n^T K^{-1}$ and $\bar{c} = K^{-1}u_n$. Then, Statement 1 follows from Theorem 2.4(2) and the fact that $\bar{b} \geq 0$, $\bar{c} \geq 0$, and $\bar{b}^T u_n = r$. Statement 2 follows from Theorem 2.4(4) and the fact that every component of $\bar{b}$ and $\bar{c}$ is greater than 0. Statement 3 follows from Theorem 2.4(3)

To prove $\bar{b}^T = u_n^T K^{-1}$ and $\bar{c} = K^{-1}u_n$, observe that $\bar{c}$ can be obtained by swapping α and β in $\bar{b}$, and the j-th column K^j of K can be obtained from the j-th row of K by the same operation. Therefore, $\bar{b}^T K = u_n^T$ if and only if $K\bar{c} = u_n$, and we only need to establish $\bar{b}^T K = u_n^T$. Since $\bar{b}^T K^1 = 1$ if and only if $\bar{b}^T K^n = 1$, we only show $\bar{b}^T K^j = 1$ for $1 \leq j < n$ as follows:

$$\begin{aligned}
\bar{b}^T K^j &= \sum_{i=1}^{n} \bar{b}_i K(i,j) \\
&= \frac{1}{\alpha\beta-1}\Bigg[\alpha(\beta-1)\alpha^{1-j} + \sum_{1<i\leq j}(\alpha-1)(\beta-1)\alpha^{i-j} + \\
&\qquad \sum_{j<i<n}(\alpha-1)(\beta-1)\beta^{j-i} + (\alpha-1)\beta\beta^{j-n}\Bigg] \\
&= \frac{1}{\alpha\beta-1}\left[\alpha^{2-j}(\beta-1) + (\alpha-\alpha^{2-j})(\beta-1) + \right. \\
&\qquad \left.(\alpha-1)(1-\beta^{j-n+1}) + (\alpha-1)\beta^{j-n+1}\right] \\
&= \frac{1}{\alpha\beta-1}(\alpha\beta-1) \\
&= 1.
\end{aligned}$$

□

3.4. Comparison with the dollar averaging strategy. The *dollar averaging strategy* (DA) is the static algorithm which invests an equal amount of capital, i.e., $1/n$ dollars, on each trading day. Thus, by Lemma 3.1, DA is the uniformly mixed strategy for the online player in the game $\Gamma_K(n,n)$. By Theorem 3.4, DA is not an optimal static algorithm, and $\Upsilon_{\mathrm{BAL}} < \Upsilon_{\mathrm{DA}}$. The next lemma gives a closed-form formula of Υ_{DA}. Figure 3.2 plots the relationship between Υ_{DA} and Υ_{BAL} for $2 \leq n \leq 100$.

LEMMA 3.5. $\Upsilon_{\mathrm{DA}} = \max\left\{\frac{n(1-\alpha^{-1})}{1-\alpha^{-n}}, \frac{n(1-\beta^{-1})}{1-\beta^{-n}}\right\}$.

Proof. Let $B_j = \sum_{i=1}^{n} K(i,j)$. By Lemma 3.2, $\Upsilon_{\mathrm{DA}} = \max_{1\leq j\leq n} \frac{n}{B_j}$. By algebra, $B_{j+1} - B_j$ is a decreasing function of j. Thus, B_j is a function of j whose minimum occurs at one end of the domain $\{1, \ldots, n\}$. The lemma follows from this concavity.
□

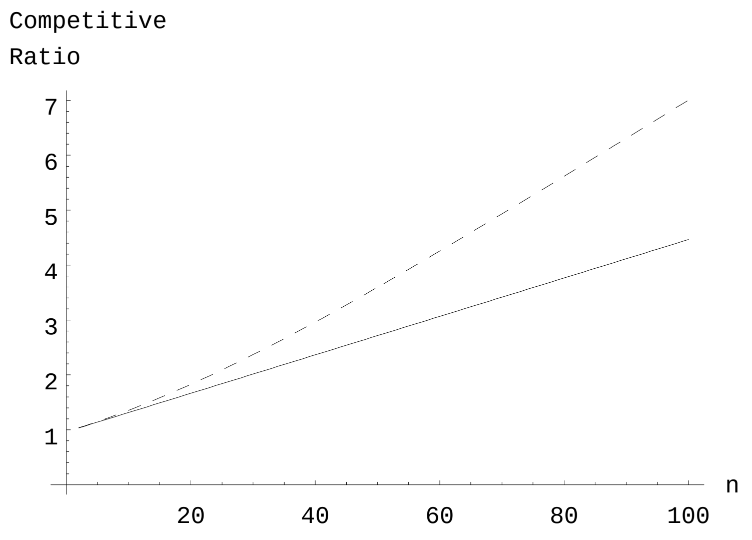

FIG. 3.2. *The dashed and solid dotted lines denote* Υ_{DA} *and* Υ_{BAL}*, respectively, with* $\alpha = 1/0.93$ *and* $\beta = 1.07$.

We have also experimented with BAL and DA using Taiwan's market data. As shown in Figure 1, the Taipei Stock Exchange (TSE) adopts $\alpha = 1/0.93$ and $\beta = 1.07$. We select the Taiwan Semiconductor Manufacturing Company (TSMC) and Acer Computer Company (Acer) for experimental analysis. TSMC is the largest foundry of wafer manufacturing in the world and is listed on both TSE and the New York Stock Exchange (NYSE) under the symbol TSM. Acer is the world's third largest PC manufacturer as well as the fifth largest mobile PC manufacturer.

Figure 3.3 shows the daily closing prices of TSMC in 1997. All stock prices are quoted in the New Taiwan dollar (NT dollar). One investment plan is executed each month. Each plan buys shares of TSMC with an initial capital of one NT dollar as in §3; however, the exchange rate of a day is the reciprocal of that day's share price without an initial normalization to one. A *monthly* accumulation is the total number of shares acquired over a month. For ease of comparison, a monthly accumulation is expressed in NT dollar by converting the acquired shares into NT dollars at the price of the last trading day of each month. Figure 3.4 shows the monthly accumulations of BAL and DA on TSMC for each month of 1997. Notice that BAL and DA are money-making except in September, October, and December. Figure 3.5 shows the *realized* competitive ratios of BAL and DA, which are the performance ratios as defined in Equation (1.1) but with $\vec{e}$ set to the actual exchange rate sequences. Note that for all twelve months, these ratios are less than 1.35. For visual clarity, we join the monthly accumulations and competitive ratios by line segments, and use the solid and dotted lines to denote the graphs of BAL and DA, respectively. Observe that overall, BAL outperforms DA.

Figure 3.6 shows the daily closing prices of Acer in 1997. Figures 3.7 and 3.8 show the monthly accumulations and realized competitive ratios of BAL and DA, respectively. The experimental results for Acer lead to similar conclusions to those for TSMC.

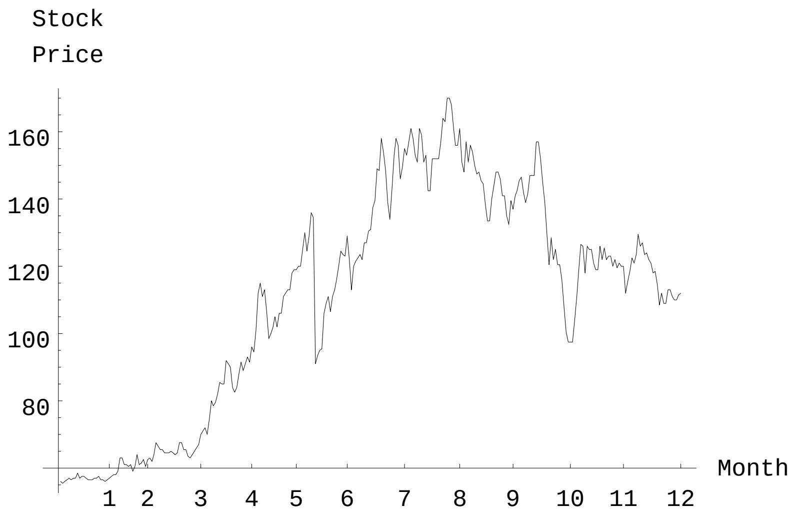

FIG. 3.3. *TSMC's daily closing stock prices in 1997.*

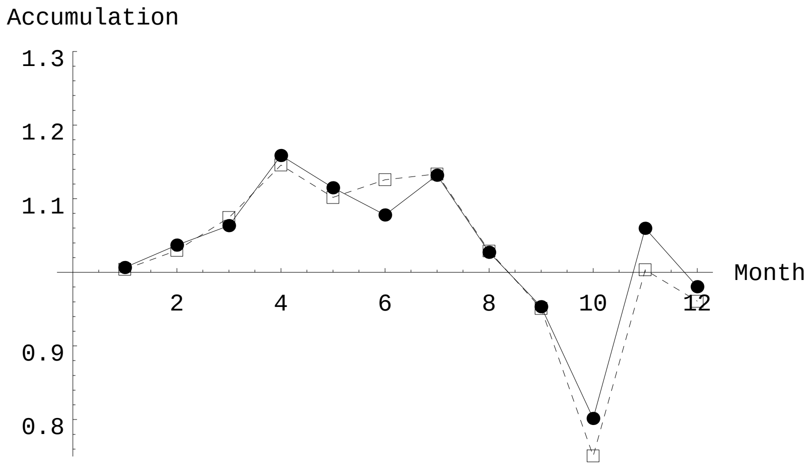

FIG. 3.4. *Accumulations of* BAL *and* DA *on TSMC.*

4. Open problems. We have presented the balanced strategy BAL and proved its unique optimality among the static algorithms. Furthermore, each of its exact competitive ratio and daily investment amounts has a closed-form expression which takes $O(1)$ time to evaluate. In light of these results, an immediate open problem is whether there are similar results for dynamic online trading algorithms. There are two orthogonal directions for further research as follows.

One direction is to change the assumption that the time horizon is fixed and known a priori to $\mathcal{B}$. For instance, it would be meaningful to consider the scenario that there is a cash stream instead of a one-time capital at the beginning of the investment horizon. For this scenario, an investor might need to guess when the cash

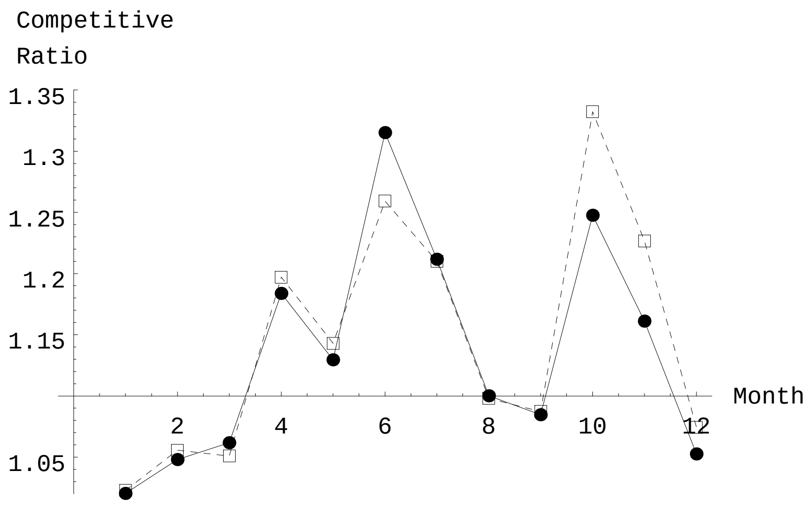

FIG. 3.5. *Realized competitive ratios of* BAL *and* DA *on TSMC.*

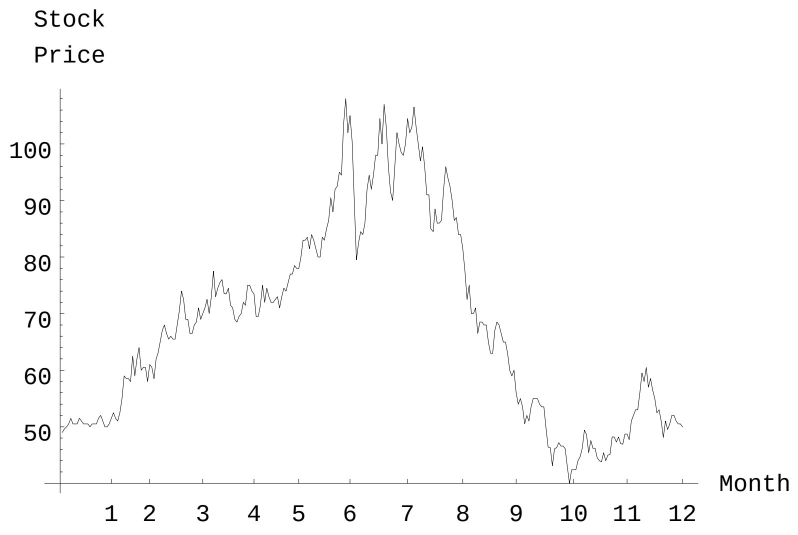

FIG. 3.6. *Acer's daily closing stock prices in 1997.*

stream will end.

The other direction is to replace α and β with a known probability distribution of the ratio $\frac{e'}{e}$. This would be an example of the standard approach in finance of considering the average-case performance under an assumed probabilistic model. While the worst-case approach in computer science is unnecessarily pessimistic, the average-case approach in finance is overly dependent on the chosen model. In general. it would be of interest to combine these two approaches to formulate more informative computational problems than either approach could.

Acknowledgments. We wish to thank the anonymous referees for very thoughtful comments. Some of the comments have resulted in open problems in §4.

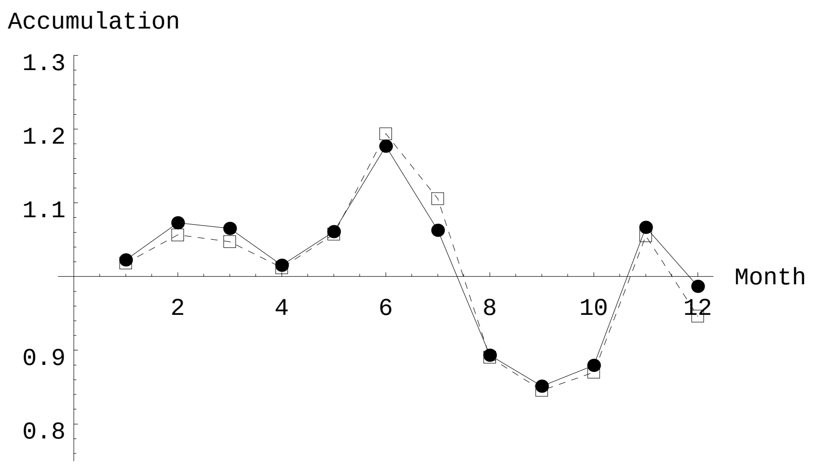

FIG. 3.7. *Accumulations of* BAL *and* DA *on Acer.*

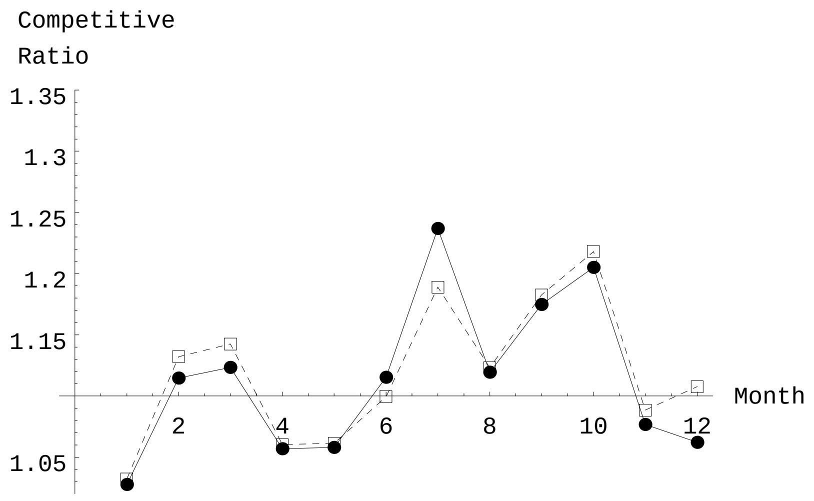

FIG. 3.8. *Realized competitive ratios of* BAL *and* DA *on Acer.*

REFERENCES

[1] M. AJTAI, N. MEGIDDO, AND O. WAARTS, *Improved algorithms and analysis for secretary problems and generalizations*, in Proceedings of the 36th Annual Symposium on Foundations of Computer Science, 1995, pp. 473–482.

[2] S. AL-BINALI, *The competitive analysis of risk taking with application to online trading*, in Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, 1997, pp. 336–344.

[3] M. BAXTER AND A. RENNIE, *Financial Calculus: an Introduction to Derivative Pricing*, Cambridge University Press, Cambridge, United Kingdom, 1996.

[4] A. BORODIN AND R. EL-YANIV, *Online Computation and Competitive Analysis*, Cambridge University Press, Cambridge, United Kingdom, 1998.

[5] A. CHOU, J. COOPERSTOCK, R. EL-YANIV, M. KLUGERMAN, AND T. LEIGHTON, *The statisti-*

cal adversary allows optimal money-making trading strategies, in Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms, 1995, pp. 467–476.

[6] Y. S. Chow, S. Moriguti, H. Robbins, and S. M. Samuels, *Optimal selection based on relative rank (the secretary problem)*, Israel Journal of Mathematics, 2 (1964), pp. 81–90.

[7] T. Cover and E. Ordentlich, *Universal portfolios with side information*, IEEE Transactions on Information Theory, 42 (1996).

[8] T. M. Cover, *Universal portfolio*, Mathematical Finance, 1 (1991), pp. 1–29.

[9] D. Duffie, *Dynamic Asset Pricing Theory*, Princeton University Press, Princeton, NJ, 1996.

[10] R. El-Yaniv, *Competitive solutions for online financial problems*, ACM Computing Surveys, 30 (1998), pp. 28–69.

[11] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin, *Competitive analysis of financial games*, in Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science, 1992, pp. 327–333.

[12] ———, *Optimal search and one-way trading online algorithms*. Manuscript, 1997.

[13] P. Freeman, *The secretary problem and its extensions*, International Statistical Review, 51 (1983), pp. 189–206.

[14] J. Hull, *Options, Futures, and Other Derivatives*, Prentice-Hall, Upper Saddle River, NJ, 3rd ed., 1997.

[15] M. Y. Kao and S. R. Tate, *Online matching with blocked input*, Information Processing Letters, 38 (1991), pp. 113–116.

[16] ———, *On-line difference maximization*, in Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms, 1997, pp. 175–182.

[17] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, *An optimal algorithm for on-line bipartite matching*, in Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, 1990, pp. 352–358.

[18] Y. D. Lyuu, *Financial Engineering and Computation: Principles, Mathematics, Algorithms*, To be published, 1999.

[19] K. R. MacCrimmon and D. A. Wehrung, *Taking Risks: The Management of Uncertainty*, Free Press, New York, NY, 1986.

[20] S. N. Neftci, *An Introduction to the Mathematics of Financial Derivatives*, Academic Press, New York, NY, 1996.

[21] E. Ordentlich and T. Cover, *On-line portfolio selection*, in Proceedings of the 9th Conference on Computational Learning Theory, 1996, pp. 310–313.

[22] L. A. Petrosjan and N. A. Zenkevich, *Game Theory*, World Scientific, Singapore, 1996.

[23] T. Raghavan, *Zero-sum two-person games*, in Handbook of Game Theory, R. Aumann and S. Hart, eds., vol. 2, Elsevier Science Publishers B. V., 1994, pp. 735–759.

[24] W. F. Sharpe, G. J. Alexander, and J. V. Bailey, *Investments*, Prentice-Hall, Upper Saddle River, NJ, 5th ed., 1995.

[25] D. Sleator and R. E. Tarjan, *Amortized efficiency of list update and paging rules*, Communications of the ACM, 28 (1985), pp. 202–208.

[26] P. Wilmott, S. Howison, and J. Dewynne, *The Mathematics of Financial Derivatives*, Cambridge University Press, Cambridge, United Kingdom, 1995.

[27] A. C. C. Yao, *New algorithms for bin packing*, Journal of the ACM, 27 (1980), pp. 207–227.