為替交換問題に対する最適に近い オンラインアルゴリズムの設計と解析

檀浦 詠介* 櫻井 幸一* 岩間 一雄**

* 九州大学大学院システム情報科学研究科 ** 京都大学工学部

文献 [R.El-Yaniv, A.Fiat, R.Karp, and G.Turpin, *Proc. of FOCS*, (1992).] において，単一方向交換問題と双方向交換問題という二つの為替交換問題に対するオンラインアルゴリズムが提案されている．単一方向交換問題とは，所持する通貨（ドル）を別の通貨（円）に交換するというものであり，一方向のみの交換を行なう．El-Yaniv らは，提案した単一方向アルゴリズムが，競合比という評価方法において最適であることを証明している．一方，双方向交換問題とは，二つの通貨の間で交換を行なうことで利益を得ることを目的としている．El-Yaniv らはこの問題に対して競合比の上下限を示しているが，その間には大きな差が存在する．

本稿ではこの双方向問題に対する競合比の上下限を改善する．k を円相場関数の極値の数とし，また円相場の変動範囲の上下限の比が 2 であるならば，従来は $(1.278)^k$ および $(1.131)^k$ であった上下限を，本研究では $(1.189)^k$ and $(1.184)^k$ まで改善している．

On Designing Nearly Optimal Algorithm for On-line Currency Conversion

Eisuke Dannoura*, Kouichi Sakurai* and Kazuo Iwama**

*Department of Computer Science and Communication Engineering Kyushu University

**Department of Information Science Kyoto University

In the paper [R.El-Yaniv, A.Fiat, R.Karp, and G.Turpin, *Proc. of FOCS*, (1992).], on-line algorithms for two versions of conversion problems are invested. One is the unidirectional conversion problem in which the player can convert assets only from dollars to yen, and the other is the bidirectional conversion problems in which the he/she can exchange dollars for yen and yen for dollars. Then, they show that their unidirectional algorithm achieves the optimal competitive ratio. However, their bidirectinal algorithm isn't optimal and there is a relatively large gap between their upper and lower bounds.

In this paper, we show the improved bounds for bidirectional problem. Let k be the number of extrema of exchange rates. Then, when the ratio of the maximum and minimum conversion rates is two, our new upper and lower bounds are $(1.189)^k$ and $(1.184)^k$, respectively. The previously-best bounds are $(1.278)^k$ and $(1.131)k$, respectively.

1 Introduction

One of the most interesting applications of on-line algorithms arises in the field of finance since it is a typical environment where one cannot have secure knowledge of what happens in the future and probably the most important goal is to prepare against the worst case. Our problem in this paper is the *currency conversion problem* between two different currencies, say dollar and yen. Its obvious purpose is to make as much profit as possible by trading one currency for the other and vice versa during some specified period of time.

Our present work was motivated by El-Yaniv, Fiat, Karp, and Turpin [EFKT92], who applied on-line algorithms to currency conversion and used the competitive analysis for the performance. It is shown that a surprisingly small competitive ratio can be achieved under reasonable assumption about the on-line player's knowledge concerning the exchange rates, i.e., only maximum and minimum rates. Their algorithm for one-way currency conversion, which deals with converting assets only from dollars to yen, is optimal or achieves the optimal competitive ratio. Unfortunately however, if we apply their algorithm into the bidirectional conversion problem, then it is no longer optimal and there is a relatively large gap between the upper and lower bounds given in [EFKT92] and no one was able to narrow this gap so far. In the bidirectional version, the player can trades dollars for yen and yen for dollars back and forth, which usually makes much more profit than the unidirectional version (if the player does a good job).

In this paper, we show an improved algorithm for the bidirectional currency conversion problem. Note that the bidirectional algorithm in [EFKT92] divides the whole exchange rate (yen / dollar) run into upward runs and downward runs and then repeats unidirectional algorithms for each run *independently*. Our improvement is based on the following idea: (i) The application of unidirectional version to the bidirectional case should not be independent, namely we should be satisfied by a smaller profit for one run if it is guaranteed that the profit for the next run is large enough to compensate the previous one. (ii) The strategy should be flexible, i.e., it should change some parameter values due to the current exchange rate. We also improve the lower bound given in [EFKT92]. The idea is to introduce a much more sophisticated input than in [EFKT92] and claim that it is still possible to design an optimal bidirectional algorithm for that input. It should be noted that our present argument may seem to remain rather small room of further improvement.

On-line financial decision making has been a popular topic: Cover [Cov91] studied the problem of portfolio selection for investment, and considered a simple on-line strategy that dramatically changes the distribution of its current wealth among the stocks. He compared this on-line strategy with a static off-line portfolio selection strategy. Raghavan [Rag91] analyzed the performance of an on-line investment algorithm under a statistical restriction on request sequences. El-Yaniv and Karp [EK93] analyzed the mortgage problem, and Azar et al. [ABFFLR96] studied the competitive analysis for the generalized problem from the mortgage problem. Chou et al. [CCEKL95] studied the currency-conversion problems without competitive analysis. Dannoura and Sakurai [DS97] analyzed the bidirectional currency conversion problem with some brokerage.

2 Models, Algorithms, and Previous Results

In the *unidirectional conversion problem*, an on-line player converts dollars to yen over some period of time and must convert all the remaining dollars to yen at the end of a trading period. It is prohibited to convert already purchased yen back to dollars, which is the reason why it is called "unidirectional." In contrast, the *bidirectional conversion problem* allows the on-line player to convert back and forth between dollars and yen at the rate of each moment. In this paper, we mainly discuss the so-called *continuous version* where the player can trade at any moment. (However, our algorithm can be easily modified to work for the *discrete version*.)

The measure of the efficiency of on-line algorithms is the so-called *competitive ratio*: Suppose that trading takes place continuously during some time interval $[0,T]$, and let $E(t)$ be an exchange-rate function defined over this interval $[0,T]$. Let $P_X(E)$ denote the amount of yen obtained at the end of the game by an on-line conversion strategy X when the rate changes due to the function $E(t)$. Let OPT denote the optimal off-line strategy where the off-line player can change all its dollars to yen when the (locally) highest exchange rate is reached and can do the opposite trading at the (locally) lowest rate. The competitive ratio of the on-line algorithm X is defined as $\sup_E \frac{P_{OPT}(E)}{P_X(E)}$.

El-Yaniv, Fiat, Karp and Turpin show in [EFKT92] that the on-line player should know in advance upper and lower bounds of the exchange rate to obtain nontrivial on-line conversion algorithms. The present paper also follows this fundamental assumption. Throughout this paper, we use M (resp. m) as an upper (resp. lower) bound on the value of the exchange rate, and a as the initial value of the rate, which the on-line player is also supposed to know in advance.

The basic strategy proposed in [EFKT92] is called the *threat-based strategy*: To attain a given competitive ratio r the on-line player simply defends himself against the threat that the exchange rate may drop to its minimum value m and never come back to higher values. They proved that the threat-based strategy yields the optimal algorithm. More precisely, the threat-based strategy consists of the following three rules.

Rule (1). At the end of the game, spend all the remaining dollars.

Rule (2). Except at the end of the game, purchase only when the current rate is the highest seen so far.

Rule (3). Whenever the exchange rate reaches a new maximum, convert just enough to ensure that a competitive ratio of r would be obtained even if the adversary dropped the exchange rate to m and kept it there throughout the rest of the game.

Let x be the exchange rate ranging from m and M, $D(x)$ and $Y(x)$ be the amount of dollars, and the amount of yen, respectively, which the player should hold when the rate is x. Initially, $x = a$, $D = 1$ and $Y = 0$. The three rules above completely determine one particular on-line algorithm, which is denoted by EFKT in this paper. EFKT has both unidirectional and bidirectional versions. When we discuss the unidirectional version, we can assume that the exchange rate is monotonically increasing. The reason is that both the optimal off-line algorithm and the threat-based algorithm conduct transactions only when the exchange rate reaches a new maximum.

The unidirectional EFKT is given by defining $D(x)$ as follows (recall that a is the initial rate known in advance):

Case 1. $a \in [m, rm]$

$$\begin{cases} x \in [a, rm] & D(x) = 1 \\ x \in [rm, M] & D(x) = 1 - \frac{1}{r} \ln \frac{x-m}{rm-m} \end{cases} \quad (1)$$

Case 2. $a \in [rm, M]$

$$x \in [a, M] \quad D(x) = \frac{a(1-\frac{1}{R})}{a-m} - \frac{1}{R} \ln \frac{x-m}{a-m}. \quad (2)$$

Thus, the algorithm is determined by the initial rate a, the lower bound m of the rate and the competitive ratio r or R, it is often denoted by EFKT(a, m, r) or EFKT'(a, m, R). Namely, if we wish to obtain some competitive ratio, say r_0, all we have to do is just to carry out the trading following the above (1) or (2). However, that is not always successful; i.e., at some moment $m \le x \le M$, $D(x)$ might become negative or we have no dollars to continue the game. We say that EFKT(a, m, r) (or EFKT'(a, m, R)) is *realizable* or simply that the competitive ratio r or R is *realizable*, if $D(M) \ge 0$, namely, if we can continue the game until the moment that the rate goes up to M.

As one can see easily, the realizable competitive ratio becomes minimum when $D(M) = 0$. That means the best possible value r_* of r can be obtained by solving the following equation:

$$r = \ln \frac{\frac{M}{m} - 1}{r - 1}. \quad (3)$$

Also, $R_*(a)$ is the best competitive ratio when the initial exchange rate a is higher than rm, which is a function of a and is usually better (smaller) than r_*:

$$R_*(a) = 1 + \frac{a-m}{a} \ln \frac{M-m}{a-m}. \quad (4)$$

Theorem 2.1 ([EFKT92]) *The unidirectional EFKT is optimal and its competitive ratio is r_* ($= 1.278$ when $M/m = 2$).*

The outline of proving the optimality is as follows: Assume that the on-line player can achieve the competitive ratio smaller than r_*. Then, at the beginning (the exchange rate is a), the player must spend more dollars than above, because otherwise the adversary immediately drops the rate into m (and the competitive ratio is not improved, a contradiction). Similarly, as the exchange rate increases, he/she must spend more dollars than specified by the above algorithm. However, this implies that the player has spent all dollars before the exchange rate reaches M and after that, although the adversary still increases the rate, he has no dollars to spend to achieve the better competitive ratio, a contradiction again.

A simple bidirectional strategy is to use the unidirectional strategy independently for each upward phase and downward phase. If the number of phases is k, i.e., if the number of local maxima and local minima is both $k/2$, then the competitive ratio is obviously at most r_*^k.

In [EFKT92], El-Yaniv et al. state that this strategy is not optimal and discussed lower bounds: The idea of proving lower bounds is to assume some restricted input for which we can design an optimal on-line algorithm. As such a restricted input, El-Yaniv et al. gave the one as shown in Fig.1, where the rate increases from m, suddenly drops to m, and repeats this movement. They also showed that the algorithm that uses the unidirectional EFKT for each unit consisting of an increase and a sudden drop is optimal. Since this single unit constitutes two phases (see the next section), the lower bound is $r_*^{k/2}$ or $(\sqrt{r_*})^k$.

Thus, when $M/m = 2$, the best known upper and lower bounds are $(1.278)^k$ and $(1.131)^k$, respectively, which will be improved into $(1.189)^k$ and $(1.184)^k$, in the following sections.

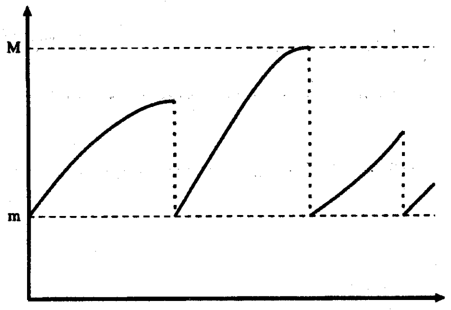

Figure 1: Exchange rates for EFKT lower bound

3 Improving Bidirectional Algorithms

3.1 Basic Idea

Let us consider the two different inputs ($E(t)$'s), as shown in Figs. 2 and 3, both consisting of the upward phase (A–B–C) and the following downward phase (C–D–E). (Note that the single unit of Fig.1 is a special case of this two-phase input where C–D and D–E are infinitely small.) The profit of the off-line algorithm is obviously the same between Fig. 2 and Fig. 3. However, if we use the unidirectional EFKT for each phase, then it is not. Recall that the "threat" against on-line players is a sudden change of the rate into its extreme and so one might think that Fig. 2 is more dangerous for on-line players than

Fig. 3. That is true for the unidirectional case, but is not any longer for the bidirectional case. Here is the reason: Note that the player has the same amount of yen at moment B and that amount of yen can be changed back into dollars at a quite low late around C in Fig. 2. In the case of Fig. 3 the same amount of yen is to be changed (gradually) at much higher rates between C and D, which means the player who follows the rule of the unidirectional EFKT can only get a smaller amount of dollars. In other words, the input like Fig. 3 is harder than the input like Fig. 2 for the on-line player.

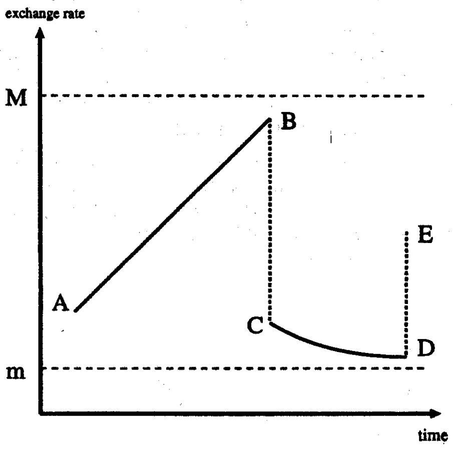

Figure 2: Example of the large dropping

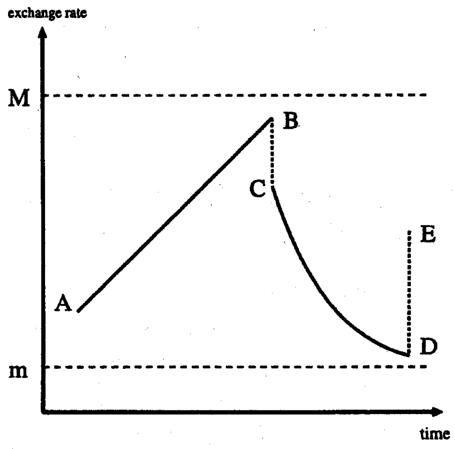

Figure 3: Example of small dropping

Recall that the on-line player gradually changes dollars to yen during the upward phase to prepare the possible sudden drop to the bottom (=m) or to prepare the "threat." On the other hand, the player wish to lower the amount of this gradual changes, since he/she wishes to change as much dollars as possible at higher rates. Now, as we have seen above, the threat is not so dangerous in the case of bidirectional conversion, and so we can actually achieve this goal simply by keeping more dollars until the price goes up. This can be done without changing the basic structure of the unidirectional EFKT but making the amount of sudden drop to be anticipated a little bit smaller than before. Namely, we can apply the unidirectional EFKT with assuming that the price can suddenly drop not to m but to some $m' > m$.

This value m' clearly depends on the current price, i.e., as the price goes up the value m' should also goes up. However, it appears to be hard to manage m' as a (continuous) function of the current rate. What we do in this paper is just to introduce some different values as m'.

3.2 Upper Bounds

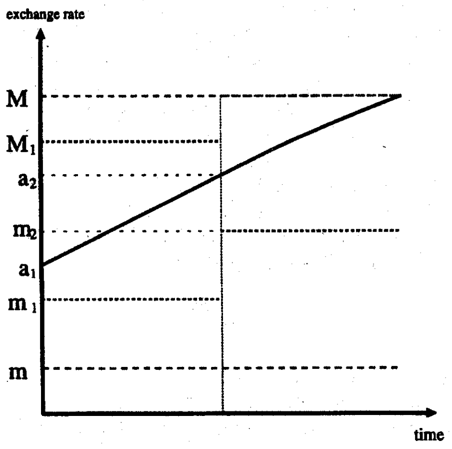

Figure 4: Outline of 2-step algorithm

As mentioned above, our new algorithm changes its strategy when the rate reaches some threshold value, denoted by a_2. We call it the *Two-Step BiDirectional Conversion* (or *2SBD* in short). Precisely speaking, 2SBD denotes the algorithm that works only for the each monotone *phase* as shown in Fig.4. We still call it "Bidirectional" since it is designed by considering the compensation of the competitive ratio between consecutive two phases due to the idea given in Sec.3.1.

Like EFKT(a, m, r), 2SBD is determined by giving the same three parameters a, m, r and furthermore the threshold value a_2; hence it is often denoted by 2SBD(a, m, r, a_2). Note that a and m are given as a part of the input. We can select any values for r and

a_2, but as before, 2SBD(a, m, r, a_2) may not be realizable. The formal definition of this realizability for 2SBD(a, m, r, a_2) will be given later. It will turn out that if 2SBD(a, n, r, a_2) is realizable, then its competitive ratio is at most r^k for k-phase inputs. In the following, we first consider the case that the initial rate is sufficiently small. Once again the following algorithm may not be executable (e.g., we cannot continue since we have no dollars to exchange) if the values given to r and/or a_2 are not appropriate.

Algorithm 2SBD(a, m, r, a_2)

(0) We first compute internal parameters r_1, m_1, r_2 and m_2 from the given values a, m, r and a_2 (details are given later). Make sure that a is "small", i.e., $a \le r_1 m_1$.

(1) While the rate changes from a to a_2, we simulate EFKT(a, m_1, r_1).

(2) When the rate gets to a_2, calculate the amount D' of dollars if we would change all the yen currently held to dollars at this rate (see below for more details). Namely, D' is the total money in dollar the player has at this moment.

(3) While the rate is between a_2 and M, we simulate EFKT(a_2, m_2, r_2) assuming that the initial amount dollars is D'.

(4) Change all the remaining dollars to yen at the end of the phase (i.e., when the rate begins to drop).

Now we explain how to calculate the initial parameters in (0) above: Among others r_1 and m_1 are first determined by equations.

$$r_1 = r, \quad \text{and} \quad m_1 = r_1 m. \tag{5}$$

By this setting, our algorithm can manage at least the following specific input: Recall that we are now assuming that $a \le r_1 m_1 (=$ denoted by $a_1)$. So, there exists the input where the rate starts from a and falls down suddenly to m when it reaches $a_1 = r_1 m_1$. Note that this input is of two phases because the next phase must be again an upward phase (since it already went to m, it cannot go down further). Also one can see that our algorithm that simulates EFKT(a, m_1, r_1) does not do any trading for this input, i.e., the player holds one dollar at the end of the input (more precisely, the player is to change one dollar to m yen when the rate falls to m, i.e., at the end of the first phase, and then immediately changes this m yen back to one dollar at the end of the second phase that coincides with the end of the first phase in this particular input). The optimal strategy can change one dollar to $r_1 m_1$ yen at rate $r_1 m_1$, and then back to $r_1{}^2 m/m = r_1{}^2 = r^2$ dollars when the rate is m. Thus the competitive ratio of our algorithm for this 2-phase input is r^2.

We next determine r_2 and m_2 due to the following idea: Suppose that the rate has reached the threshold value a_2, i.e., it has not fallen before that. At this moment, if we consider that we finish EFKT(a, m_1, r_1), then we can calculate the whole amount Y of money in yen by changing all the remaining dollars to yen at this rate a_2. This Y is better than the similar amount Y' EFKT(a, m_1, r_1) can make for the input which suffers from some sudden drop before it reaches a_2. Note that our competitive ratio r_1 assumes this worse value Y'. In other words, the competitive ratio, say r', of our algorithm for the better result Y is better than r_1. It is not hard to see that we can write

$$r_1' = \frac{1}{r} + \left(1 - \frac{m_1}{a_2}\right)\left(1 - \frac{1}{r}\ln\frac{a_2/m_1 - 1}{r - 1}\right). \tag{6}$$

Now define $r_2 = r/r'$. Then one can see that we can achieve the competitive ratio r for the input that continues to go up after reaching a_2 (and may drop to at worst m_2). To determine m_2, we can use equation (4), i.e., m_2 is determined so as to satisfy

$$r_2 = 1 + \frac{a_2 - m_2}{a_2}\ln\frac{M - m_2}{a_2 - m_2}. \tag{7}$$

The similar algorithm for the case that the initial rate a is higher than a_1 is denoted by 2SBD'. There are two different cases, $a \in [a_1, a_2]$ and $a \in [a_2, M]$. In the latter case, we simply omit the first step of 2SBD and then for the second step, we use the optimal EFKT whose initial value is a and whose final rate ($=$ the rate at which the amount of dollars becomes 0) is M. In the former case, we simulate EFKT'$(a, m_1, R(a))$ instead of EFKT(a_1, m_1, r_1) in the first step. Here, EFKT'$(a, m_1, R(a))$is the optimal EFKT' whose initial rate is a and whose final rate is the same as the final rate of EFKT(a, m_1, r). Using the fundamental equations of EFKT', one can get

$$R(a) = \begin{cases} 1 + \frac{a - m_1}{a}\ln\frac{M_1 - m_1}{a - m_1} & a \in [a_1, a_2] \\ 1 + \frac{a - m_2}{a}\ln\frac{M - m_2}{a - m_2} & a \in [a_2, M] \end{cases} \tag{8}$$

where $M_1 = m\{1 + (r - 1) * e^r\}$.

Thus 2SBD(a, m, r, a_2) has been completely specified. What remains to do is to obtain the best realizable r and the threshold value a_2, which make 2SBD(a, m, r, a_2) realizable, from the values a, m and M. Note that this optimization process is not a part of the on-line algorithm since m and M are given in advance and without less of generality we can set $a = m$ (recall that if $a \le r_1 m_1$, then the algorithm does not depend on the value a). So, it is enough to show that some particular values actually satisfies conditions for the realizability. Now let us define that 2SBD(a, m, r, a_2) is *realizable* if it passes the following three tests:

(i) 1st-Step-Test(a, m_1, r_1, a_2): The algorithm passes this test if it can continue EFKT(a_1, m_1, r_1) until the rate reaches a_2, i.e., there remain some dollars (≥ 0) at rate a_2.

(ii) 2nd-Step-Test(a_2, m_2, r_2): The algorithm passes this test if it can continue EFKT(a_2, m_2, r_2) until the rate reaches M.

(iii) InterPhase-Test: The algorithm passes this test if its competitive ratio is at most r^2 for the following specific 2-phase input $I(w)$: The rate goes up to a_2, then right after we start the simulation of EFKT(a_2, m_2, r_2), it falls suddenly to some value $w < m_2$ and then goes down to m. It is not hard to see that the competitive ratio for the first phase (until w) is

$$\frac{r(a_2 - m_2)}{a_2 + (\frac{r}{r'} - 1)w - \frac{rm_2}{r'}}, \tag{9}$$

which might be worse than r. For the second phase, i.e., from w to m, we can apply 2SBD', i.e., the algorithm for the input where initial rate is higher than a_1. Although details are omitted, its competitive ratio is not worse than the following values

$$R(a') = \begin{cases} 1 + \frac{a'-m_1}{a'} \ln \frac{M_1 - m_1}{a' - m_1} & a' \in [a_1, a_2], \\ 1 + \frac{a'-m_2}{a'} \ln \frac{M - m_2}{a' - m_2} & a' \in [a_2, M]. \end{cases} \tag{10}$$

Since the algorithm used in the second phase above is not 2SBD' itself but its opposite-direction version, its competitive ratio can be written as $R(mM/w)$. As a result, the following condition should be met to pass InterPhase-Test:

$$\frac{r(a_2 - m_2)}{a_2 + (\frac{r}{r'} - 1)w - \frac{rm_2}{r'}} R(mM/w) \leq r^2. \tag{11}$$

All we have to do now is to seek values r and a_2 that passes the three tests above. The following informal procedure appears to work quite well:

(1) Set an arbitrary (but reasonable) value to r.

(2) Compute r_1 and m_1.

(3) Set some value, say $\sqrt{a_1 M_1}$, to a_2 and compute r_2 and m_2.

(4) Execute the three tests. If all the tests are passed then r is realizable and try an even smaller value. Otherwise, first try to adjust the value of a_2. It might help to know that if we make a_2 smaller then it becomes easier to pass both 1st-Step-Test and InterPhase-Test, but becomes harder to pass the 2nd-Step-Test. If this adjustment of a_2 fails, then we have to make r larger and repeat from (2).

Thus we finally obtain our bidirectional conversion algorithm which repeats realizable 2SBD(a, m, r, a_2) and its opposite-direction version.

Theorem 3.1 *Suppose that k is even and the k-phase input has the same initial and final rate. Then if 2SBD(a, m, r, a_2) is realizable, then the whole algorithm achieves r^k as its competitive ratio for the k-phase input.*

(Proof sketch) By the construction, it is obvious that r^k is achieved if the rate does not fall (in upward phases) below m_1 or m_2. If it does, then the worst case is that the rate falls right after the first step or right after the second step (proof is omitted). However, we have considered this worst case when r_1 is determined by introducing InterPhase-Test. □

Remark. If the final rate of the k-phase input is lower than its initial rate, then we can get a better competitive ratio. Otherwise, the ratio can become only as bad as r^{k+1}.

2SBD can be extended to kSBD whose parameters can be determined by the following procedure.

(1) Set some value to r as before, which determines r_1, m_1 and a_1.

(2) Set some value to a_2 and make sure that EFKT(a_1, m_1, r_1) is realizable at a_2. Now r_2 is determined.

(3) Set some value $m_2 \in [m_1, a_2]$.

(4) Repeat (2) and (3) for a_i, r_i and m_i for $i = 3, 4, \cdots$. If a_k and r_k are set, then the final m_k is determined.

(5) Execute InterPhase-Test at the beginning of each step.

Table 1 shows an example of realizable competitive ratios and related parameter values for $k = 2, 4$, and 8 when $M/m = 2$

Theorem 3.2 *When $M/m = 2$, the best realizable competitive ratio for 8SBD is less than* 1.1894

4 Lower Bounds

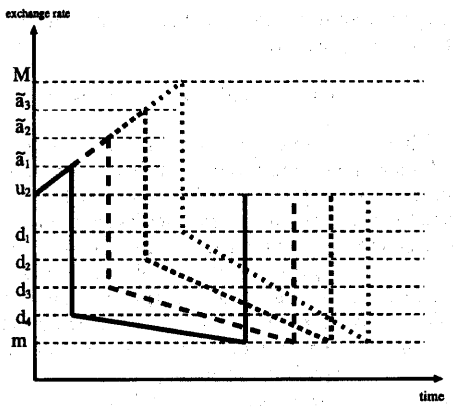

Figure 5: Exchange rates for our lower bound

The idea of proving lower bounds is to find some restricted input for which we can design an optimal online-conversion algorithm. In this paper, we use the input I_0, that looks like Fig.5. Our input, I_0 can take one of the four difficult types of rate changes: Type 1: The rate goes up from u_2, suddenly goes down to d_4 before it reaches $\tilde{a}_1$, goes down toward m and finally *goes suddenly* up to u_2. Type 2, 3 and 4 are similar, but they suddenly go down to d_3, d_2 and d_1 before getting to $\tilde{a}_2, \tilde{a}_3$ and M, respectively. For the reason mentioned later, we design I_0 so that it will satisfy the following conditions: Firstly the values of $\tilde{a}_1, \tilde{a}_2$ and $\tilde{a}_3$ (we shall again call them threshold val-

Table 1: The realizable parameter values for $k = 2, 4$, and 8 when $M/m = 2$

k	$(m_2, \cdots, m_k)$	$(a_2, \cdots, a_k)$
2	(125.2)	(155.6)
4	$(121.6, 125.4, 130.6)$	$(147.8, 156.0, 168.2)$
8	$(121.5, 123.3, 125.3, 127.5, 132.9, 133.0, 136.5)$	$(145.8, 154.2, 164.0, 168.1, 173.4, 176.5, 181.5)$

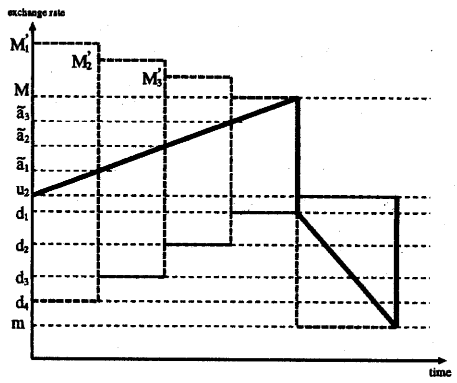

Figure 6: Optimal Algorithm for the restricted inputs

ues) have to satisfy the following equations:

$$\frac{\tilde a_3}{d_1 - d_2} - r^{-1}(d_1, m, u_2) \ln \frac{M - d_1}{\tilde a_3 - d_1} = 0, \qquad (12)$$

$$\frac{\tilde a_2}{d_2 - d_3} - r^{-1}(d_2, m, u_2) \ln \frac{\tilde a_3 - d_2}{\tilde a_2 - d_2} = \frac{\tilde a_3}{d_1 - d_2} \qquad (13)$$

$$\frac{\tilde a_1}{d_3 - d_4} - r^{-1}(d_3, m, u_2) \ln \frac{\tilde a_2 - d_3}{\tilde a_1 - d_3} = \frac{\tilde a_2}{d_2 - d_3} \qquad (14)$$

where r^{-1} are given later. Due to the second conditions, $\tilde a_1, \tilde a_2$ and $\tilde a_3$ are now functions of d_1, d_2, d_3 and d_4. The second condition is that $u_2 = mM/d_2$. Now combined with the first condition, I_0 is completely determined by giving the values of d_1, d_2, d_3 and d_4. Thus, I_0 is often expressed by $I_0(d_1, d_2, d_3, d_4)$. The last condition is rather technical, i.e., we will select the values d_1, d_2, d_3 and d_4 so as to satisfy that

$$m < d_4 < d_3 < d_2 < \sqrt{mM} < d_1 < M. \qquad (15)$$

Fig.6 illustrates the idea of our optimal algorithm for the input I_0. The algorithm is denoted by $A(r'_1, r'_2, r'_3, r'_4)$ (or simply by A if nothing wrong happens) where the parameter values r'_i will be determined later. (1) While the rate is going up from u_2 to $\tilde a_1$, $A(r'_1, r'_2, r'_3, r'_4)$ simulates EFKT(u_2, d_4, r'_1), (2) from $\tilde a_1$ to $\tilde a_2$, it simulates EFKT$(\tilde a_1, d_3, r'_2)$, (3) from $\tilde a_2$ to $\tilde a_3$, it simulates EFKT$(\tilde a_2, d_2, r'_3)$, and (4) from $\tilde a_3$ to M, it simulates EFKT$(\tilde a_3, d_1, r'_4)$. (5) After I_0 suddenly goes down to d_i ($i = 1, 2, 3$ or 4), A simulates OPT-EFKT$^{-1}(d_i, m, u_2)$ which is the optimal EFKT such that its initial rate is d_i and the upper and lower bounds of the rate are m and u_2, respectively.

Recall that the initial rate is high, EFKT requires us to change a certain amount of money at the very beginning of the game. This means that our current algorithm A might also requires us to change some amount of money when it changes, for example, from EFKT$(\tilde a_1, d_3, r'_2)$ to EFKT$(\tilde a_2, d_2, r'_3)$ at the rate $\tilde a_2$ discontinuously. The condition, 12 - 14 we introduced before present this discontinuous behaviors, i.e., the amount of dollars one should hold at the end of EFKT$(\tilde a_1, d_3, r'_2)$ and at the beginning of EFKT$(\tilde a_2, d_2, r'_3)$ are the same under those conditions.

Now let us determine the values of r'_1, r'_2, r'_3 and r'_4 so that A$(r'_1, r'_2, r'_3 . r'_4)$ will be optimal. It should be noted first that A's behavior after the rate has fallen to d_i is obviously optimal. So, let $r^{-1}(d_i)$ be the competitive ratio of OPT-EFKT$^{-1}(d_i, m, u_2)$. Then as was mentioned in Sec.3.1,

$$r^{-1}(d_4) \le r^{-1}(d_3) \le r^{-1}(d_2) \le r^{-1}(d_1) \qquad (16)$$

The basic idea of determining the values r'_1, r'_2, r'_3 and r'_4 is to determine them so that the competitive ratio of A will be the same for the type 1, type 2, type 3 and type 4 inputs. To compensate the relation 16 above, we have to achieve increasingly better competitive ratio as the input changes from type 1, type 2, type 3 to type 4.

First of all it is not hard to decide r'_4. Since the final rate of EFKT$(\tilde a_3)$ (= the rate at which the amount of dollars becomes 0) must be M, r'_4 can be obtained from the fundamental equations of EFKT. Namely, $r'_4 = \frac{\tilde a_3 - d_1}{\tilde a_3} \ln \frac{M - d_1}{\tilde a_3 - d_1}$. Then we decide r'_3 from the condition that A achieves the same ratio for the type 3 and type 4 inputs. Note that the trading is exactly the same for type 3 and type 4 until the rate goes up to $\tilde a_2$. So we can think this $\tilde a_2$ is the starting point and for simplicity we assume that the player has 1 dollar at this moment. (More precisely, we should say that if the player changes all the yen to dollars at this rate $\tilde a_2$, then he/she would hold 1 dollar. Namely, we assume that $\tilde a_2 D(\tilde a_2) + Y(\tilde a_2) = \tilde a_2$.)

Now starting from this $\tilde a_2$, A simulates EFKT$(\tilde a_2, d_2, r'_3)$. If the rate falls to d_2 before it reaches $\tilde a_3$, then it is type3 and the competitive ratio is obviously $r'_3 r^{-1}(d_2)$. Otherwise, if the rate goes to

$\tilde{a}_3$, then the amount of the dollars at the moment can be calculated by the following fundamental equation

$$D(x) = \frac{\tilde{a}_2(1 - 1/r'_3)}{\tilde{a}_2 - d_2} - \frac{1}{r'_3} \ln \frac{x - d_2}{\tilde{a}_2 - d_2} \qquad (17)$$

for EFKT($\tilde{a}_2, d_2, r'_3$) and the total amount of money D' in dollar at that moment can be written as

$$D' = D(\tilde{a}_3) + Y(\tilde{a}_3)/\tilde{a}_3 = (1 - \frac{d_2}{\tilde{a}_3})D(\tilde{a}_3) + \frac{1}{r'_3} \ (< 1). \qquad (18)$$

Therefore, the competitive ratio for the type 4 input is $(\frac{1}{D'})r'_4 r^{-1}(d_1)$ (i.e., from $\tilde{a}_2$ to $\tilde{a}_3$, the on-line player reduces his/her total money from 1 to D' dollars). Now since this ratio is equal to the ratio for type 3,

$$r'_3 D' r^{-1}(d_2) = r'_4 r^{-1}(d_1), \qquad (19)$$

from which we can get r'_3. We can use the same procedure to get r'_1 and r'_2.

Theorem 4.1 $A(r'_1, r'_2, r'_3, r'_4)$ *is optimal for* $I_0(d_1, d_2, d_3, d_4)$.

(Proof sketch) Note that our input I_0 can be divided into some "ranges" $[u_2, \tilde{a}_1], [\tilde{a}_1, \tilde{a}_2], \cdots, [d_1, m], [d_2, m]$, and soon. It is clearly possible to see which range the current rate is in, for example, if the rate gradually goes up over $\tilde{a}_2$, then the current rate is in range $[\tilde{a}_2, \tilde{a}_3]$. If the rate suddenly drops to d_2 and then further goes down gradually, then the current rate is in range $[d_2, m]$.

Now suppose that some algorithm B uses a strategy that is different from EFKT in some range. Then it is easy to show that B can be improved by using EFKT in that range. Next suppose that B shows a worse competitive ratio for, say, type 2 than type 3. Then we can modify the EFKT used in the range $[\tilde{a}_1, \tilde{a}_2]$ so that it will spend a little more dollars and $[\tilde{a}_2, \tilde{a}_3]$ a little less dollars. This implies that the ratio for type 2 is improved and the ratio for type 3 is worsened, which again improves the whole ratio.

So, we can assume that any optimal algorithm always uses EFKT as its subalgorithms and achieves the same competitive ratio for the four types. Then we can prove that the player must hold some dollars $D(\geq 0)$ at the maximum rate M. (Otherwise, the ratio for type 4 must be smaller than others.) If $D > 0$, then we can again improve the ratio for all the types. We can therefore conclude that $D = 0$. Now the algorithm is determined uniquely which must be $A(r'_1, r'_2, r'_3, r'_4)$ as described above. □

Now we enter the optimization process by adjusting d_1, d_2, d_3 and d_4 to get the worst ratio. In this process, one should be careful since for some values of d_1, d_2, d_3, d_4, some of $\tilde{a}_1, \tilde{a}_2, \tilde{a}_3$ may be smaller than u_2, If $\tilde{a}_1$ is smaller than u_2, then the type 1 input no longer exists and the above arguments should be done for the remaining three different types of inputs. The design process of our optimal algorithm is similar. In this particular 2-phase inputs, the worst ratio is 1.173^2 when $d_1 = 152.9$, $d_2 = 131.5$, $d_3 = 116.2$, $d_4 = 105.8$.

We can again extend this approach to k-phase inputs. First of all, we design the optimal algorithm not only for I_0 but also for I_{01}, I_{02}, I_{03} and I_{04}, where I_{01} begins with not u_2 but $u_1 = mM/d_1$. I_{02} is the same as I_0 (begins with u_2). I_{03} and I_{04} begins with u_3 and u_4, respectively. Now we add one more previous phase and get again four different types of inputs, according to where it changes the phase, at u_1, u_2, u_3 or u_4 after the first phase. Then we can use exactly the same technique to design the optimal algorithm for this 3-phase input. Since this input includes 4×4 different types of inputs in total, the competitive ratio per single phase of the optimal algorithm becomes worse. Our computer experiment shows:

Theorem 4.2 *When $M/m = 2$, there exists 50-phase input for which the optimal algorithm cannot achieve the competitive ratio that is better than* $(1.184)^{50}$

References

[ABFFLR96] Azar, Y., Bartal, Y., Feuerstein, E., Fiat, A., Leonardi, S. and Rosen, A., "On capital investment," *In Proc. of the 23rd ICALP,* 429-441,xii+680, 1996.

[Cov91] Cover, T.M. "Universal Portforios," *J. of Math. Finance* 1(1) pp. 1-29, January 1991.

[CCEKL95] Chou, A., Cooperstock, J., El-Yaniv, R., Klugerman, M. and Leighton, T., "The Statistical Adversary Allows Optimal Money-Making Trading Strategies," *In Proc. of SODA'95,* 1995.

[DS97] Dannoura, E. and Sakurai, K., "On-line versus off-line in money-making strategies with BROKERAGE," *In Proc. of ISAAC'97,* 1997.

[EFKT92] El-Yaniv, R., Fiat, A., Karp, R. and Turpin, G., "Competitive Analysis of Financial Games," *In Proc. of the 33rd FOCS,* pp.327-333, 1992.

[EK93] El-Yaniv, R. and Karp. R., "The Mortgage Problem," *In Proc. of 2nd Israel Symposium on Theory and Computing Systems,* 1993.

[Rag91] Raghavan, P., "A statistical adversary for on-line algorithms," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* 7,79-83, 1991.