Playing the Matching-Shoulders Lob-Pass Game with Logarithmic Regret

Joe Kilian[*†] **Kevin J. Lang**[*‡] **Barak A. Pealmutter**[§¶]

April 26, 1994

Abstract

The best previous algorithm for the matching shoulders lob-pass game, Abe and Takeuchi's (1993) ARTHUR, suffered $O(t^{1/2})$ regret. We prove that this is the best possible performance for any algorithm that works by accurately estimating the opponent's payoff lines. Then we describe an algorithm which beats that bound and meets the information-theoretic lower bound of O(log t) regret by converging to the best lob rate *without* accurately estimating the payoff lines. The noise-tolerant binary search procedure that we develop is of independent interest.

1 Background

The lob-pass problem has its origins in the animal psychology literature (Herrnstein, 1990), but following Abe and Takeuchi (1993) we consider it to be the following simplified tennis game. For each of a sequence of plays we choose to make either a lob or a passing shot. The opponent then stochastically hits or misses our shot, and in the latter case we score a point. The opponent adapts in a way that devalues shots that have been selected frequently. Specifically, the chance of scoring with a given shot is a linear function of the cumulative lob rate, which is the total number of lobs that we have made divided by the total number of shots that we have made. Throughout this paper we represent the cumulative lob rate by the variable r. The payoff functions of two sample opponents are shown in figure 1. Clearly, the odds of scoring with a lob decreases as one moves to the right on the diagrams, and the odds of scoring with a pass decreases as one moves to the left. We should emphasise that the x-axis here does not correspond to our instantaneous lob rate, but to the cumulative lob rate that is calculated by the opponent. This means that the opponent is rather slow in adjusting to changes in our behavior.

The figure of merit is regret: the expected number of points that we could score if we began the game with complete knowledge of the oppenent's payoff functions minus the number of points that we actually do score while simultaneously learning and playing. There is a conflict

[*]NEC Research Institute.
[†]joe@research.nj.nec.com
[‡]kevin@research.nj.nec.com
[§]Siemens Corporate Research.
[¶]bap@learning.scr.siemens.com

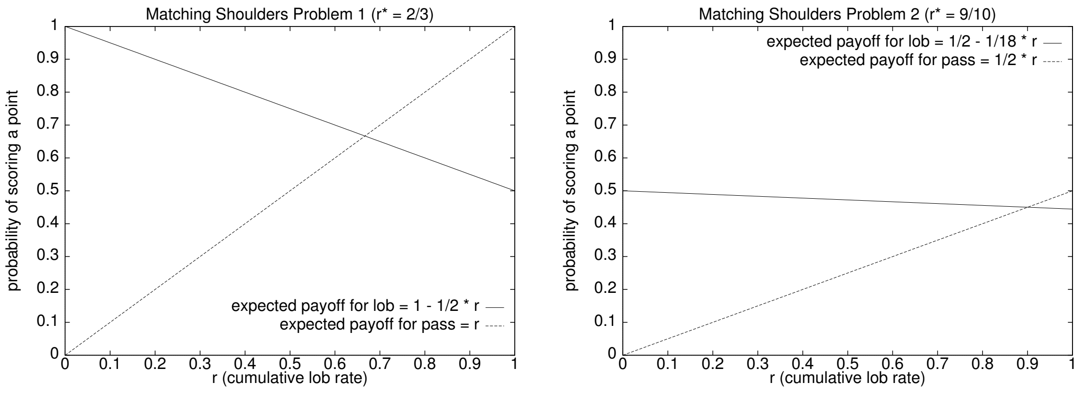


Figure 1: The two sample opponents against which our algorithms were run.

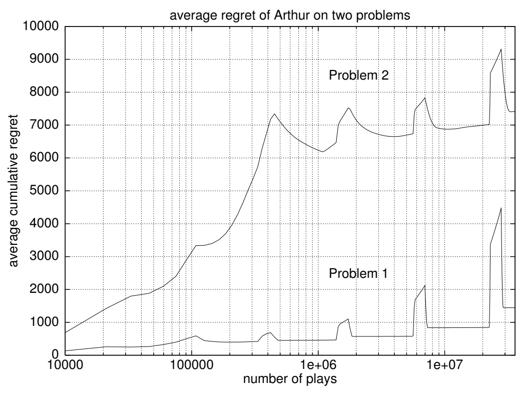
between the goals of exploration (varying the cumulative lob rate to learn more about the opponent's payoff lines) and exploitation (playing at a lob rate that allows us to score many points).

Following Abe and Takeuchi (1993) we concentrate on the "matching shoulders" version of the game, in which the expected payoff for a lob at $r = 0$ equals the expected payoff for a pass at r=1. This version of the game has the special property that the optimal cumulative lob rate r^* is located at the intersection of the two payoff lines.

The ARTHUR algorithm of Abe and Takeuchi (1993) converges to r^* by constructing a sequence of increasingly accurate models of the opponent. It alternates between distinct exploration and exploitation phases. In the exploration phase it plays at two widely separated lob rates in order to obtain the leverage needed to accurately estimate the two payoff lines. In the exploitation phase it plays at the lob rate specified by the intersection of the two lines. Figure 2(a) show the average regret of ARTHUR against the two sample opponents. The jumps in regret occur during the exploration phase of the algorithm. Abe and Takeuchi (1993) proved that the regret of ARTHUR grows no faster than $O(t^{1/2})$ when r^* is bounded away from 0 and 1 and the lines' slopes are bounded away from 0.

2 A Lower Bound

We derive a new lower bound of $O(t^{1/2})$ expected regret for the class of lob-pass algorithms whose line estimates gain accuracy at the same rate as their estimates of r^*. ARTHUR belongs to this class, so it is doing as well as it can. Our proof (details omitted) makes explicit the exploration *vs.* exploitation tradeoff by establishing a circular set of inequalities between the variance of the estimates of the lines' parameters, the variance of the estimate of r^*, the statistical leverage, and the expected regret. These relations are inconsistent with an expected regret of less than $O(t^{1/2})$.



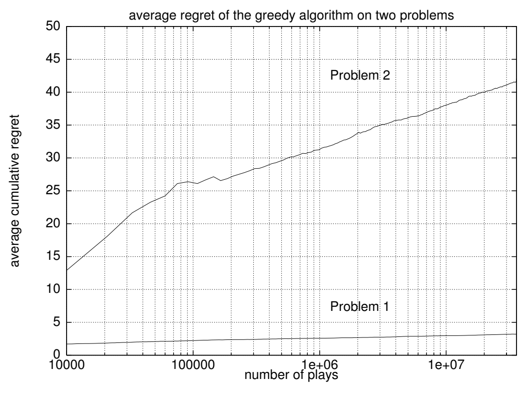


Figure 2: Regret of ARTHUR *vs.* our greedy algorithm.

The main result of this paper is that a completely greedy strategy (all exploitation and no exploration) can play the matching shoulders lob-pass game with $O(\log t)$ regret. The $O(t^{1/2})$ lower bound is beaten by converging to r^* without bothering to obtain accurate estimates of the payoff lines.

3 An Algorithm with Logarithmic Regret

In the matching shoulders lob-pass game one can determine whether the optimal lob rate r^* lies to the right or left of a given lob rate r by measuring whether the average payoff is higher for lobs or for passes at that lob rate. This observation converts the problem of playing the lob-pass game to the problem of finding a target point on the real line by querying a noisy comparison oracle. In lemma 6.1 we solve the latter problem by describing a noise-tolerant binary search procedure whose guesses converge to the target point at an exponential rate. In the present application we simulate the comparison oracle by playing the lob-pass game long enough to give a confident opinion about which action has the higher payoff at the requested lob rate. Successive queries take exponentially longer to answer but incur a roughly constant amount of regret, so regret accumulates logarithmically with time.

The amount of regret per query would be exactly constant if there were a sequential statistical procedure that could take two completely unknown coins q_1 and q_2, flip them $O(1/(q_1 - q_2)^2)$ times, and determine with fixed confidence p whether $q_1 > q_2$. Suppose that our oracle simulation employed this fictitious statistical procedure. Let d be the distance between a query point r and the target value r^*. By geometry, the difference between the two payoffs at r is $O(d)$, so the procedure could reliably tell which payoff was higher after making $O(1/d^2)$ plays. Since the expected regret per play is $O(d^2)$, the total regret incurred while processing the query is $O(1)$.

Unfortunately, any sequential test that is correct with fixed confidence p for arbitrarily small

$|q_1 - q_2|$ requires more than $O(1/(q_1 - q_2)^2)$ examples.[1] Thus, we adopt a more complicated search <algorithm that supplements each query with a nominal lower bound l on $|q_1 - q_2|$, and which is able to tolerate the additional mistakes that occur when the bound is violated. In lemma 6.2 we show that such a search procedure exists, and that both d and l converge exponentially fast (to the target point and to 0 respectively). With this search procedure the oracle's statistical problem becomes trivial; it can determine which line is higher with the required confidence by computing the average payoffs from a batch of $O(l^{-2})$ plays.

The regret incurred while processing a query is now $O(d^2/l^2)$. The quantities d and l are probabilistically linked by the search algorithm so that $d = O(l)$ is the most likely state and higher values of d/l are progressively less likely. We are interested in the shape of this tail because it determines the variance of the regret-per-query distribution. It turns out that an oracle that responds correctly with fixed confidence p whenever $|q_1 - q_2| > l$ does not provide a sharp enough tail for our purposes. Fortunately the truth rate of our oracle simulation increases very rapidly with increasing $|q_1 - q_2|/l$. In lemma 6.3 we prove that this shrinks the tail of the d/l distribution enough to yield a regret-per-query distribution with bounded variance, and also that some tricky dependency issues can resolved. For all practical purposes, this lemma brings us back to constant regret per query. Here then, is our main result:

Theorem 3.1 *On any instance of the lob pass game where r^* is bounded away from 0 and 1 and the payoff slopes are bounded away from 0, the expected cumulative regret of the above algorithm at time t is $O(\log t)$ with probability approaching 1.*

Proof Sketch. The bounds on the payoff slopes and on the position of r^* are used to construct linear mappings between the quantities x_n and u_n manipulated by the search algorithm and the quantities r_n and l_n employed by the oracle simulation. By lemma 6.3, after j queries the total regret is less than $c_1 j$ with probability $1 - e^{-c_2 j}$. The elapsed time t after j queries is at least the number of plays required to process the last query. By lemma 6.2, this is at least $O(e^{c_3 j})$ with probability $1 - e^{-\sqrt{j}/2}$. Therefore, $j \leq O(\log t)$ and the cumulative regret is less than $O(\log t)$ with probability $1 - e^{-\sqrt{\log t}/2}$. □

We note that this algorithm meets the information theoretic lower bound on regret. This bound can be informally derived as follows. We are trying to estimate r^*. By elementary statistics the standard deviation of the estimate cannot be less than $O(1/\sqrt{i})$ at the ith play. Because instantaneous regret grows with the square of distance from r^*, it must be at least $O(1/i)$. Cumulative regret is therefore at least $O(\log i)$.

4 A Practical Algorithm

Now we turn to a much more efficient (but harder to analyze) algorithm for the matching shoulders lob-pass game. This algorithm maintains least squares estimates of the two payoff

[1] A sequential test that is almost good enough is the following: apply the batch t-test for significantly different means to a sequence of fresh batches of coin flips. The size of batch i is $O(2^i)$, and the required confidence level is $1 - O(2^{-i})$. This procedure discriminates between arbitrarily close coins with about $O((q_1 - q_2)^{-2} \log\log(q_1 - q_2)^{-2})$ flips. Plugging this test into our binary-search algorithm gives a lob-pass player with regret $\log t \log\log t$.

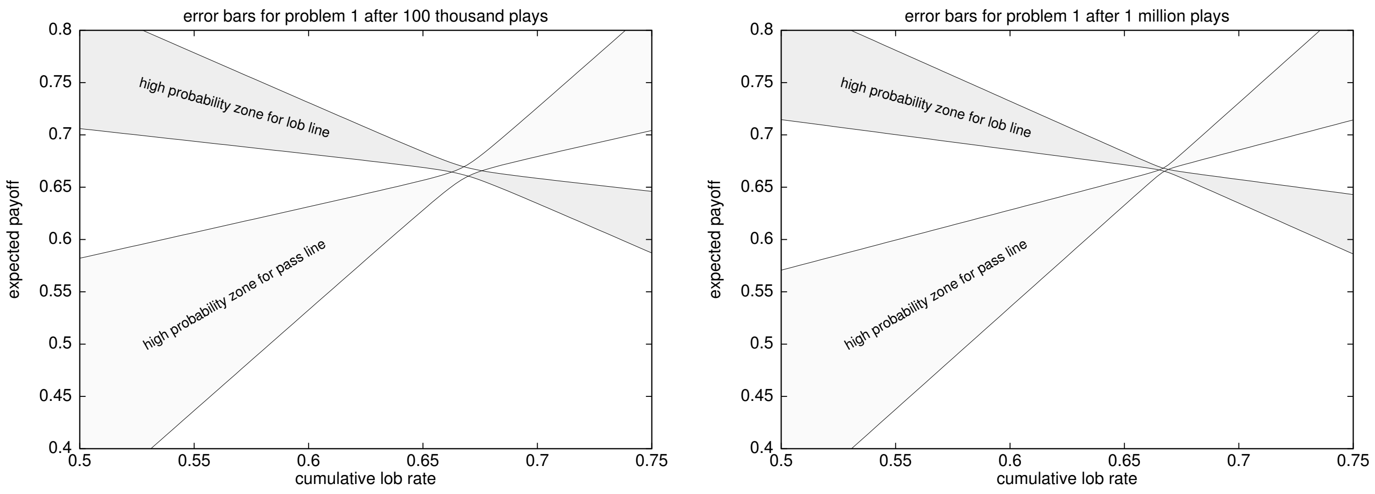


Figure 3: Plausible ($\sigma = 1$) locations for payoff lines after 100,000 and 1,000,000 plays. The intersection can be known with arbitrarily greater accuracy than the slopes.

lines, and it always selects the play that appears to offer the better expected payoff at the current cumulative lob rate.[2] Figure 2(b) shows the average regret of this algorithm on several hundred games against two different opponents. It is clear that the constant factors are very good; against the first opponent our average regret is only 3 after 10 million plays. Asymptotically, regret appears to grow like $\log t$. But the algorithm is estimating the payoff lines! Why isn't it limited by the $O(t^{1/2})$ lower bound of section 2? The answer is that the algorithm converges to r^* so fast that it obtains very little leverage, and hence on any reasonable time scale it remains unsure about the orientation of the lines. However, the variances and covariances of its parameter estimates cancel out in such a way that the algorithm knows where the two lines cross despite its uncertainty about the lines themselves.

The situation is illustrated by figure 3(a), in which we have plotted confidence windows (at one standard deviation) for the two payoff lines after 100 thousand plays against the first opponent. The zone of high probability for each payoff line is bounded by a pair of hyperbolas, and is narrowest in the vicinity of $r = 2/3$, where most of the play has occurred. The most probable location of r^* is in the diamond-shaped intersection of the two confidence windows. Note that this diamond is already quite small, yet the flared-out shapes of the individual confidence windows show that the data is insufficient to accurately estimate the slope of either line.

Figure 3(b) shows what happens to the confidence windows when the algorithm is allowed to make an additional 900 thousand plays. Because these new actions all occur at lob rates near r^*,

[2]Details: Our least squares code incorporates the matching shoulders constraint. We disallow plays that would cause the cumulative lob rate to approach 0 (or 1) faster than $4/\sqrt{t+64}$ to insure that both plays are tried infinitely often.

the new data doesn't provide enough leverage to noticeably improve the flared-out shapes of the two confidence windows. However, the two hyperbolas defining each confidence window have approached each other by shifting vertically.[3] These vertical shifts don't require any leverage, yet they are able to significantly reduce the size of the diamond-shaped high-confidence zone for r^*.

To the extent that the curved sides of this diamond-shaped region can be approximated by straight lines[4] we can explain the apparent logarithmic regret of the algorithm as follows. Obtaining k samples near r^* reduces the vertical dimension of each confidence window at r^* to $O(1/\sqrt{k})$. The geometry of parallelograms turns this into a horizontal uncertainty of $O(1/\sqrt{k})$ in the position of r^*. Squaring to get instantaneous regret and then integrating gives a cumulative regret of $O(\log k)$.

5 A Note on the General Case

An algorithm that obtained less than $O(t^{1/2})$ expected regret in the *non*-matching shoulders case could be used to estimate the slope of a line with accuracy exceeding the statistical limitation imposed by the available leverage. This establishes a lower bound on the expected regret. By playing toward the current estimate of r^* plus a cyclic perturbation of magnitude $O(t^{-1/4})$ for leverage, one can achieve average regret that grows like $O(t^{1/2})$, meeting this lower bound.[5] The perturbation causes cyclic swings in regret with amplitude $O(t^{3/4})$. Because these swings dominate the long-term growth rate, periodically the regret is $O(t^{3/4})$. On the other hand, by arranging to be at the bottom of a swing at the end of a finite game, a player could stop with a *negative* regret of size $O(t^{3/4})$.[6] By adopting a cyclic perturbation of magnitude $O(t^{-1/3})$ one can reduce the upper bound to $O(t^{2/3})$, but then the average regret increases to $O(t^{2/3})$ and the swings into negative regret disappear.

6 Appendix: Binary Search

There is an existing literature on noise-tolerant binary search (see Borgstrom and Kosaraju (1993) for a recent paper), but it has focused on finite-length searches. Our results provide for rapid convergence over an infinite sequence of queries.

First, we consider the basic scenario in which we seek to approximate an unknown target point x_{target} on the real line, given a noisy oracle that states whether a query x_n lies to the right or left of the x_{target}. On each query, the oracle gives the correct answer with independent probability p.

[3]Also, the hyperbolas have become pointier.

[4]This approximation improves with time.

[5]An appropriately modified ARTHUR can also meet this lower bound.

[6]That is, the score would be $O(t^{3/4})$ higher than would be achieved by playing at r^* for the entire game.

Lemma 6.1 *For any constant p bounded away from $1/2$, there is a search algorithm* A *and a constant c such that for sufficiently large n*

$$\Pr\left(|x_n - x_{target}| > e^{-cn}\right) < 1 - e^{-\sqrt{n}/2}.$$

Proof Sketch. Due to the independence of the correctness of the answers, we can use a standard majority vote strategy to boost the correctness probability to any constant bounded away from 1. A maintains a hypothesis interval $[x_l, x_u]$, denoting its estimated lower and upper bounds on x_{target}. Let $w = |x_u - x_l|$ (the hypothesis width) and $\delta = \max(|x_l - x_{target}|, |x_u - x_{target}|)$ (the distance from the far endpoint to the target point). At each iteration, A divides $[x_l, x_u]$ into 6 subintervals and queries the 7 endpoints. Depending on the pattern of responses, A either expands the hypothesis interval by a factor of 2 in the indicated direction (i.e. changing $[x_l, x_u]$ to $[x_l, x_u + w]$ or $[x_l - w, x_u]$, or contracts to the indicated subinterval. Every correct action reduces δ^2/w by at least a factor of $3/2$, while the worst mistake increases it by a factor of 6. With a sufficiently accurate oracle, such as with $p > .972$, the average change in $\log \delta^2/w$ will be negative. Because all changes are of bounded size, a martingale argument shows that $\log \delta^2/w$ decreases at a linear rate and hence δ^2/w is exponentially small with probability approaching 1. Finally, $w/4 \leq \delta/2 \leq \delta^2/w$. $\square$

Our application actually requires a couple of deviations from this simple oracle model. The new oracle takes a query (x_n, u_n), where $u_n > 0$, and computes $d_n = |x_n - x_{target}|$. If $u_n > d_n$ then the oracle answers correctly with probability $p = 1 - e^{-\alpha d^2/u^2}$. If $u_n \leq d_n$ the oracle's behavior is unconstrained. Our application places two competing requirements on u_n. The simplest is that with high probability, u_n decrease exponentially with n. We first sketch an algorithm that achieves the upper bound on u_n.

Lemma 6.2 *Let α be sufficiently large. Then there is a algorithm* B *such that for sufficiently large n, $|x_n - x_{target}| < O(e^{-cn})$ and $u_n < O(e^{-cn})$ with probability* $1 - e^{-\sqrt{n}/2}$.

Proof Sketch. We modify A as follows: Before each iteration, B replaces the hypothesis interval $[x_l, x_u]$ by $[x_l - s, x_r + \frac{w}{5} - s]$, where s a chosen uniformly from the real interval $[0, \frac{w}{5}]$. Finally, B sets $u_n = w/2000$. By a simple probability argument, the probability that a given query will come within u of x_{target} is some small constant, regardless of the values of x_l, x_u and x_{target}. The proof of convergence of x_n and u_n is then completely analogous to that of Lemma 6.1. $\square$

So far, there has been no reason not to set u_n as small as possible. However, our application imposes a cost of $O(d_n^2/u_n^2)$ on each query, and requires a linear bound on the total cost of a sequence of queries with high probability. Here is where we make use of the oracle's improved performance when d_n^2/u_n^2 becomes large.

Lemma 6.3 *Let α be a sufficiently large constant. Then there exists a constant c' such that for algorithm* B,

$$\Pr\left[\sum_{j=1}^{n}(d_j^2/u_j^2) > cn + c'n\right] < 1 - e^{-c^2 n}.$$

Proof Sketch. By making c' sufficiently large, we can ignore all cases where $d_i^2/u_i^2 \leq 100$. When $d_i^2/u_i^2 > 100$ then with constant factors we can approximate d_i^2/u_i^2 by δ_i^2/w_i^2 (since $u_i = w/2000$). Suppose that an iteration of B begins with query (x_i, u_i). By the same argument as in the proof of Lemma 6.1, we can show that this iteration of B decreases δ_i/w_i by some constant factor except for unlucky events in which case it increases δ_i/w_i by at most some constant factor. We use here the fact that x_{target} is far outside the interval $[x_l, x_u]$ when δ_i/w_i is large. These unlucky events occur with probability $c_1 e^{-\alpha_1(\delta_i^2/w_i^2)})$ for some constant α_1 that is proportional to α and some normalization constant c_1.

Part of the difficulty in analyzing the sum of δ_i^2/w_i^2 is that the terms are highly correlated. To get around this problem, we first make a charging argument that allows us to only consider the contributions of bad events. We then model the cost of the bad events in a way which is at least as bad as what actually occurs, but in which there is complete independence in the modeled steps of the algorithm.

Suppose that an iteration of B begins with query (x_i, u_i), and a bad event occurs. Due to the geometric rise and fall of δ_i^2/w_i^2, we can simply charge δ_i^2/w_i^2 to our total cost and be within a constant factor of the actual cost. Thus, we need only worry about the bad events.

To eliminate correlations, we consider the "worst-case" model M which at each step assigns a cost randomly according to $\Pr[\text{cost} > q] = c_2 e^{-\alpha_2 q}$ for some sufficiently small positive α_2 and normalization constant c_2. At each iteration of the algorithm, there is a cost $q_i = \delta_i^2/w_i^2$ that will be incurred with probability $c_1 e^{-\alpha_1(\delta_i^2/w_i^2)})$. Regardless of the value of q_i, M will assign at least as much cost at least as often (up to constant factors). Let the expectation of the cost assigned by M be c^*, and note that the variance of M is constant. Thus, up to constant multiplicative factors, the probability of incurring a total cost greater than $cn + c^*n$ is less than the probability that the sum of n values independently chosen from M. At this point, the lemma follows from a constructive version of the Central Limit Theorem applied to M. □

References

Abe, N. and Takeuchi, J.-i. (1993). The 'Lob-Pass' Problem and an On-Line Learning Model of Rational Choice. In *Sixth Annual ACM Workshop on Computational Learning Theory*, pp. 422–428 Santa Cruz, CA.

Borgstrom, R. S. and Kosaraju, S. R. (1993). Comparison-Based Search in the Presence of Errors. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pp. 130–136.

Herrnstein, R. (1990). Rational Choice Theory. *American Psychologist*, *45*(3), 356–367.