Optimal Strategies from Random Walks

Jacob Abernethy[*]
Division of Computer Science
UC Berkeley
jake@cs.berkeley.edu

Manfred K. Warmuth[†]
Department of Computer Science
UC Santa Cruz
manfred@cse.ucsc.edu

Joel Yellin
Division of Physical and
Biological Sciences
UC Santa Cruz
yellin@soe.ucsc.edu

Abstract

We analyze a sequential game between a Gambler and a Casino. The Gambler allocates bets from a limited budget over a fixed menu of gambling events that are offered at equal time intervals, and the Casino chooses a binary loss outcome for each of the events. We derive the optimal min-max strategies for both participants. We then prove that the minimum cumulative loss of the Gambler, assuming optimal play by the Casino, is exactly a well-known combinatorial quantity: the expected number of draws needed to complete a multiple set of "cards" in the generalized Coupon Collector's Problem. We show that this quantity and the optimal strategy of the Gambler can be efficiently estimated from a simple random walk.

1 Introduction

This paper analyzes the problem of sequential prediction and decision making from the perspective of a two player game. The game is played by a learner, called here the Gambler, who makes a sequence of betting decisions. The Gambler's opponent is the Casino in which he plays.

Gambler vs. Casino:

1. On each day, the Gambler arrives at the Casino with \$1. The Casino presents n *events* and each event is played once per day. The Gambler chooses a distribution vector $\mathbf{w} \in [0,1]^n$, where $\sum w_i = 1$, and bets the portion w_i of his \$1 budget on event i.

2. On each day the Casino determines the outcome of each event with the objective of winning as much money from the Gambler as possible. In particular, after observing the distribution of the Gambler's bets the Casino decides between a *loss* or a *no loss* for all daily events. These choices are summarized by a *loss vector* $\boldsymbol{\ell} \in \{1,0\}^n$ where $\ell_i = 1$ implies that on event i, the Gambler lost. (For simplicity, we assume the only relevant quantities are losses. By shifting our baseline we can model *wins* as *non-losses*).

3. At the end of each day, the Gambler leaves the Casino having lost $\mathbf{w} \cdot \boldsymbol{\ell} = \sum_i w_i \ell_i$ and the cumulative loss of the gambler is updated as $L \leftarrow L + \mathbf{w} \cdot \boldsymbol{\ell}$. The Gambler also monitors the cumulative performance of each event with a state vector $\mathbf{s} \in \mathbb{N}^n$, where s_i is the current total loss of event i. After incurring loss $\boldsymbol{\ell}$ at the current day, the state vector is updated to $\mathbf{s} \leftarrow \mathbf{s} + \boldsymbol{\ell}$.

4. The Gambler stops playing as soon as he observes that each even has suffered more than k losses, where k is some fixed positive integer known to both. The Casino is aware of this decision and behaves accordingly.

Gambling against a casino may seem an unlikely starting point for a model of sequential decision making – we generally consider the typical environment for learning to be *stochastic* rather than *adversarial*. Yet are these two environments necessarily incompatible? Among the objectives of this paper is to address questions such as: "What will be the Gambler's worst-case cumulative loss?"; and "What is the optimal betting strategy?" These questions, while clearly game-theoretic, are ultimately answered here by considering a *randomized* Casino rather than an adversarial one. From this perspective, randomness may indeed be the Gambler's worst adversary.

Early work on sequential decision making focused on the problem of predicting a binary outcome given advice from a set of n *experts*. In that setting, the goal of the predictor is to combine the predictions of the experts to make his own prediction, with the objective of performing well, in hindsight, compared to the best expert. The performance of both the learner and the experts is measured by a loss function that compares predictions to outcomes. One of the early algorithms, the Weighted Majority algorithm [LW94], utilizes a distribution corresponding to the degree of *trust* in each expert.

It was observed by Freund and Schapire [FS97] that the analysis of the Weighted Majority algorithm can be applied to the so-called *hedge setting*. Rather than predict a binary outcome, the learner now plays some distribution over the experts on every round, a loss value is assigned to each expert independently, and the learner suffers the expected loss according to his chosen distribution. In this case, the learner bears the exact burden of the Gambler - that of "hedging" his bets so as to minimize his cumulative loss. To emphasize that the Gambler/Casino game is useful for settings other than prediction, we use the term "event" rather than "expert".

[*]Supported by DARPA grant FA8750-05-2-0249 and NSF grant DMS-0707060.

[†]Supported by NSF grant CCR 9821087.

A central theme of much of the sequential decision making literature is the use of so-called "exponential weights" to determine the learner's distribution on each round. Use of the exponential weighting scheme in the case of the Casino game results in the following strategy for the Gambler: At a state $\mathbf{s}$, bet

$$w_i = \frac{\beta^{s_i}}{\sum_j \beta^{s_j}} \quad \text{on event } i, \tag{1}$$

where the factor β lies in $[0, 1)$.

From the analysis of the Weighted Majority algorithm it follows that the cumulative loss of the Gambler using the above strategy is bounded by

$$\frac{\ln n + k \ln \frac{1}{\beta}}{1 - \beta}.$$

Under the assumption that the loss of the best event is at most k, the factor β can be tuned [FS97] so that the above bound becomes

$$k + \sqrt{2k \ln n} + \ln n.$$

The exponential weights framework, as well as other online learning techniques, can be motivated using the method of relative entropy regularization [KW99]. While the resulting algorithms are elegant and in some cases can be shown to be asymptotically optimal [CBFHW96], they do not optimally solve the underlying game. Some improvements have been made using, for example, binomial weights that lead to slightly better but still non-optimal solutions [CBFHW96] in a setting where the experts must produce a prediction. While it is formally easy to define the optimal algorithms using minimax expressions, it has generally been assumed that actually computing an efficient solution is quite challenging [CBFH$^+$97]. More recently, however, a minimax result [ALW07] was obtained for the specific game of prediction with absolute loss. The resulting algorithm, Binning, is efficient and optimal in a slightly relaxed setting.

In this paper we show that the minimax solution to the Gambler/Casino prediction game, which is identical to the underlying game of the hedge setting with binary losses, can be obtained efficiently. In addition, the game can be fully analyzed using a simple Markov process: a random walk on an n-dimensional lattice. The value of the game, that is the cumulative loss of an optimal Gambler, can be interpreted as the expected length of such a random walk. The Gambler's optimal play, the portion of his budget he should bet on a given event, can similarly be interpreted as manifesting an assessment of the probability of a specific random outcome of this walk.

The game's stopping criterion, that is when all events have lost at least $k + 1$ times, may seem unusual at first yet fits quite naturally within the experts framework. Indeed, online learning bounds are often tuned with an explicit a priori knowledge of the cumulative loss of the best expert, which here would be k[1]. While perhaps not realistic in practice, k can be estimated and various techniques such as successive doubling can be used to obtain near-optimal bounds [CBFH$^+$97].

The paper is structured as follows. In Section 2 we give a minimax definition of the optimal value of the game considered here. In Section 2.1 we modify the game by restricting the adversary's choices to unit loss vectors. In Section 3, we then turn our attention to a specific Markov process with a number of relevant properties. We apply this randomized approach to the Casino game in Section 4, where we prove our main results. In Section 5 we give recurrences and exact formulas, based on sums over multinomials, for the value of the game and for the optimal probabilities. We set out an efficient method to compute both the optimal strategy of the Gambler and the value of the game. In Section 6 we compare the optimal regret bound to previous results, and in Section 7 we draw a connection between our game and a well studied version of the coupon collector problem. We also briefly summarize what is known about the asymptotics of this problem. We conclude with a discussion of our results and list open problems (Section 8).

2 The Value of the Game

Assume that in each event the Gambler has already suffered some losses specified by state vector $\mathbf{s}$. Define $V(\mathbf{s})$ to be the total money lost by an optimal Gambler playing against an optimal Casino *starting from the state* $\mathbf{s}$. That is, $V(\mathbf{s})$ is the amount of money that the optimal Gambler will lose (against an optimal Casino) from now until the end of the game. Roughly speaking, the value of the game is computed as:

$$V(\mathbf{s}) \stackrel{?}{=} \min_{\text{dist. } \mathbf{w}} \max_{\boldsymbol{\ell} \in \{0,1\}^n} \mathbf{w} \cdot \boldsymbol{\ell} + V(\mathbf{s} + \boldsymbol{\ell})$$

The Gambler chooses $\mathbf{w}$ to minimize the loss while the Casino chooses $\boldsymbol{\ell}$ to maximize the loss, where the loss is computed as the loss $\mathbf{w} \cdot \boldsymbol{\ell}$ on this round plus the worst case loss $V(\mathbf{s}+\boldsymbol{\ell})$ on future rounds. However, we have to be careful, as this recursive definition doesn't address the following issues:

- When is the game over? What is the base case of $V(\cdot)$?
- Is this recursion bounded?
- Do we need to record the losses $\mathbf{s}_i$ that go above k?

We address these issues beginning with some simplifications and notational conventions. First, we assume that the state vector $\mathbf{s}$ lies within the set $\mathcal{S} = \{0, 1, \ldots, k+1\}^n$. Note that it is not necessary to record the losses of events that have already crossed the k threshold. We call such events *dead*. Since the losses of dead events are not restricted, having loss $k + 3$ is the same as loss $k + 100$. We therefore "round" all states $\mathbf{s}$ into the state space $\{0, 1, \ldots, k + 1\}^n$ using the notation $\dot{+}$ which we define below. We use the notation $\lambda(\mathbf{s})$ to record the set of live events; the statement $i \notin \lambda(\mathbf{s})$ is exactly the statement $s_i = k + 1$.

Second, as the game is defined recursively, we must guarantee that this recursion terminates. If the Casino repeatedly chose $\boldsymbol{\ell} = \langle 0, \ldots, 0 \rangle$, for example, the game would make no progress. The same problem occurs if the Casino causes

[1] Strictly speaking, in the expert setting it is assumed that at least one expert has not crossed the k-mistake threshold, while here we stop the Gambler/Casino game when the loss of the *last* expert/event goes beyond this threshold. It is easy to show that this slight modification, made for convenience, increases the worst-case loss of the Gambler by exactly 1.

losses on only dead events. We must therefore place additional restrictions so that the dead state is reached eventually. The simplest way to ensure this is to forbid the Casino from inflicting loss on only dead events. Yet this is not sufficient: with this restriction alone the Gambler would have a guaranteed non-losing strategy by betting solely on dead events. We thus assume that neither can the Casino can inflict losses on dead events nor can the Gambler bet money on them (keeping in mind that all such bets are in any case non-optimal). We must enforce this explicitly in order to have a well-defined game.

We use two notational conventions to describe the above restrictions. First, we write $\mathbf{w} \sim \lambda(\mathbf{s})$ to describe the set $\{\mathbf{w} \in \Delta_n \mid w_i = 0 \; \forall i \notin \lambda(\mathbf{s})\}$ where Δ_n is the n-simplex. We also abuse notation slightly and write $\boldsymbol{\ell} \subset \lambda(\mathbf{s})$ to mean that $\boldsymbol{\ell} \in \{0,1\}^n$ and $\ell_i = 0$ for all $i \notin \lambda(\mathbf{s})$.

We now define the value of the game precisely.

Definition 1 *Define the value $V(\mathbf{s})$ of the game as follows.*

- *At the dead state, $V(\mathbf{d}) := 0$.*
- *For any other $\mathbf{s} \in \mathcal{S}$, we define $V(\mathbf{s})$ recursively as*

$$V(\mathbf{s}) := \min_{\mathbf{w} \sim \lambda(\mathbf{s})} \max_{\mathbf{0} \neq \boldsymbol{\ell} \subset \lambda(\mathbf{s})} \mathbf{w} \cdot \boldsymbol{\ell} + V(\mathbf{s} + \boldsymbol{\ell}). \quad (2)$$

In our notation, we commonly make use of several special states. The state where the game begins is the "initial" state, $\mathbf{s} = \mathbf{0}$. Once all events have lost more than k times the game is over and we refer to this as the *dead* state $\mathbf{d}$. It will also be useful to consider *one-live* states $\mathbf{o}_i$, where all events except i are dead, and the remaining event has exactly k losses. By the game definition, it is easy to check that $V(\mathbf{o}_i) = 1$, since the Gambler must bet all of his money on this event, and the Casino must inflict a corresponding loss, charging the Gambler \$1 and ending the game.

Below, we include a list of notations for reference:

Notation:

$\mathcal{S} := \{0, \ldots, k+1\}^n$	(the state space)
$\mathbf{0} := \langle 0, 0, \ldots, 0 \rangle = \langle 0^n \rangle$	(the initial state)
$\mathbf{d} := \langle (k+1)^n \rangle$	(the dead state)
$\mathbf{o}_i := \mathbf{d} - \mathbf{e}_i$	(ith one-live state)
$\lambda(\mathbf{s}) := \{i \in [n] : s_i \leq k\}$	(set of live events)
$\mathbf{s} \dot{+} \boldsymbol{\ell} := \langle \min(s_i + \ell_i, k+1) \rangle$	("rounded" addition)
$\lvert\mathbf{s}\rvert := \sum s_i$	(elementwise sum)
$\Delta_n := \{\mathbf{w} \in \mathbf{R}_+^n : \lvert\mathbf{w}\rvert = 1\}$	(the n-simplex)

2.1 The Modified Game

We also consider a modified game that we make easier for the Gambler. In this new game, we restrict the Casino to inflict loss on exactly *one* event in each round, i.e. $\boldsymbol{\ell}$ must be a basis vector $\mathbf{e}_1, \ldots, \mathbf{e}_n$. So for $\boldsymbol{\ell} = \mathbf{e}_i$ we have $\mathbf{w} \cdot \boldsymbol{\ell} = w_i$. We can then precisely define the value $\widehat{V}(\cdot)$ of the modified game:

Definition 2 *Define $\widehat{V}(\mathbf{d}) := V(\mathbf{d}) = 0$. Otherwise*

$$\widehat{V}(\mathbf{s}) := \min_{\mathbf{w} \sim \lambda(\mathbf{s})} \max_{i \in \lambda(\mathbf{s})} w_i + \widehat{V}(\mathbf{s} + \mathbf{e}_i). \quad (3)$$

One of the central results of this paper is that the above game, while seemingly more restricted, is ultimately just as difficult for the Gambler as the original game. It is easy to show that $V(\mathbf{s}) \geq \widehat{V}(\mathbf{s})$, since the Casino has strictly more choices in the original game. We go further and prove as our main result in Theorem 12 that

$$V(\mathbf{s}) = \widehat{V}(\mathbf{s}).$$

Thus both games have the same worst-case outcome.

Both the analysis of the modified game, as well as the proof of the above result, requires a different formulation of the Casino's actions.

3 A Randomized Casino

In Section 2 we presented a game-theoretic analysis of a well-known sequential prediction problem characterized as a game between a Gambler and a Casino. In the present section, we consider a different framework, in which the Casino uses random events. We will show that introducing a randomized strategy of the Casino enables us to specify the optimal strategy of the Gambler.

3.1 A Random Walk on the State Graph

Let us now imagine that our Casino does not fix outcomes deterministically, but instead chooses the outcome of each event using the following random process. Assume we are at state $\mathbf{s}$ and that, on each day, an event i is chosen uniformly at random from $\{1, \ldots, n\}$ and a loss is assigned to event i. In other words, the loss vector $\boldsymbol{\ell}$ is a uniformly sampled unit vector $\mathbf{e}_i$, and after the loss the new state is $\mathbf{s} \dot{+} \mathbf{e}_i$. This process continues until we reach the dead state $\mathbf{d}$.

We can model this behavior as a Markov process on the state space as follows. Consider any sequence of indices $I_1, I_2, \ldots \in [n]$, and let $S_t := \sum_{m=1}^{t} \mathbf{e}_{I_m}$, where $S_0 := \mathbf{0}$. Assuming that we start at state $\mathbf{s}$, this induces a sequence of states

$$\mathbf{s} = \mathbf{s} \dot{+} S_0 \to \mathbf{s} \dot{+} S_1 \to \mathbf{s} \dot{+} S_2 \to \ldots \to \mathbf{s} \dot{+} S_t.$$

Notice that this process has "self-loops"; i.e. it is quite possible that $\mathbf{s} \dot{+} S_t = \mathbf{s} \dot{+} S_{t+1}$. This occurs when $(\mathbf{s} \dot{+} S_t)_{I_{t+1}}$ is already at $k+1$.

If we imagine the state space $\mathcal{S}$ as an n-dimensional lattice, which we will call the *state lattice*, then the Markov process above can be interpreted as a random walk on this lattice. The walker starts at the initial state $\mathbf{0}$, and on every time interval a positively directed single step is taken along an axis drawn uniformly at random. If the walker has already reached the $k+1$ boundary in this dimension, he remains in place. The walk stops once the dead state $\mathbf{d}$ is reached. We will show that the value V is $1/n$ times the expected total number of random draws that achieves this position. Thus V is the expected walk/path length from $\mathbf{s}$ to $\mathbf{d}$.

3.2 Survival Probabilities

We now define a *survival probability* at a state $\mathbf{s}$. We will show in the next section that such probabilities are the basis for the Gambler's optimal strategy.

Definition 3 *Assume we are at state $\mathbf{s}$, and let the random state $\mathbf{s} \dot{+} S_t$ be the result of the above random walk after t*

steps. Define the ith survival probability $\widehat{p}_i(\mathbf{s})$ to be the probability that

$$\exists t: \ \mathbf{s}\dot{+}S_t = \mathbf{o}_i.$$

Equivalently,

$$\widehat{p}_i(\mathbf{s}) \ = \ Pr(\lambda(\mathbf{s}\dot{+}S_t) = \{i\} \textit{ for some } t).$$

We call these survival probabilities since $\widehat{p}_i(\mathbf{s})$ is the probability that, if the losses were assigned randomly to the events in sequence, the ith event would be the last non-dead event.

Lemma 4 *For any $\mathbf{s} \neq \mathbf{d}$, the vector*

$$\widehat{p}(\mathbf{s}) := \langle \widehat{p}_i(\mathbf{s}) \rangle_{i=1}^n$$

defines a distribution on $\{1, \ldots, n\}$.

Proof: The quantity $\sum_i \widehat{p}_i(\mathbf{s})$ is the probability that eventually there is exactly one live event. This probability is exactly 1, given that the current state is not the dead state $\mathbf{d}$. ■

We list some examples of survival probabilities:

- When $\mathbf{s} = \mathbf{0}$ (or any other symmetric state), we have
$$\widehat{p}_i(\mathbf{s}) = \frac{1}{n}, \ \forall i$$
bexause there is a uniform chance of survival.
- When i is a dead event, i.e. $s_i = k + 1$, then
$$\widehat{p}_i(\mathbf{s}) = 0$$
because no dead event can be the last remaining live event.
- If there is only one remaining live event, i.e. $\lambda(\mathbf{s}) = \{i\}$, then
$$\widehat{p}_i(\mathbf{s}) = 1.$$

Computing $\widehat{p}_i(\mathbf{s})$ for more general $\mathbf{s}$ requires a recursion, and we leave this discussion for Section 5.

3.3 Expected Path Lengths

Another important quantity we consider is the *length of a random path*, i.e. the number of steps in the random walk on the state lattice required until the dead state $\mathbf{d}$ is reached.

Definition 5 *For a sequence $S_0, S_1, \ldots$, let*

$$T(\mathbf{s}) := \min\{t \geq 0 : \mathbf{s}\dot{+}S_t = \mathbf{d}\}.$$

That is, $T(\mathbf{s})$ is the length *of the random path starting at $\mathbf{s}$ and just entering $\mathbf{d}$. Furthermore, let*

$$\tau(\mathbf{s}) := \mathbb{E}\, T(\mathbf{s})$$

be the expected path length.

We note that paths may be infinitely long due to self-loops, yet such paths occur with probability 0. A key fact is that the expected path length $\tau(\mathbf{s})$ can be rewritten using indicator variables:

$$T(\mathbf{s}) = \sum_{t=0}^{\infty} \mathbf{1}[\mathbf{s}\dot{+}S_t \neq \mathbf{d}], \tag{4}$$

i.e. $T(\mathbf{s})$ is the number of initial segments (including the empty segment) of a random path starting at $\mathbf{s}$ that has not reached the dead state $\mathbf{d}$.

We now prove a relationship between expected path length $\tau(\mathbf{s})$ and survival probabilities $\widehat{p}_i(\mathbf{s})$:

Lemma 6 *For any state $\mathbf{s}$ and event i,*

$$\widehat{p}_i(\mathbf{s}) = \frac{1}{n}(\tau(\mathbf{s}) - \tau(\mathbf{s}\dot{+}\mathbf{e}_i)).$$

Proof: When $i \notin \lambda(\mathbf{s})$, then $\mathbf{s} = \mathbf{s}\dot{+}\mathbf{e}_i$ and it is trivially true that

$$\widehat{p}_i(\mathbf{s}) = 0 = \frac{1}{n}(\tau(\mathbf{s}) - \tau(\mathbf{s}\dot{+}\mathbf{e}_i)).$$

The interesting case is when $i \in \lambda(\mathbf{s})$. Indeed, Using (4), we have

$$\begin{aligned}
&\tau(\mathbf{s}) - \tau(\mathbf{s} + \mathbf{e}_i) \\
&\quad = \ \mathbb{E}\, T(\mathbf{s}) - \mathbb{E}\, T(\mathbf{s} + \mathbf{e}_i) \\
&\quad = \ \mathbb{E}\left[\sum_{t=0}^{\infty} \mathbf{1}[\mathbf{s}\dot{+}S_t \neq \mathbf{d}] - \mathbf{1}[(\mathbf{s} + \mathbf{e}_i)\dot{+}S_t \neq \mathbf{d}]\right].
\end{aligned}$$

Since the dead state $\mathbf{d}$ is an absorbing state we have that for any path S, if $\mathbf{s}\dot{+}S = \mathbf{d}$, then $\mathbf{s} + \mathbf{e}_i\dot{+}S = \mathbf{d}$ as well. Equivalently, if $(\mathbf{s} + \mathbf{e}_i)\dot{+}S \neq \mathbf{d}$, then $\mathbf{s}\dot{+}S \neq \mathbf{d}$. Thus in the difference between the expectations, we only need be concerned with sequences S_t that are accounted for in the first expectation but not in the second. Therefore the above difference becomes

$$= \ \mathbb{E}\left[\sum_{t=0}^{\infty} \mathbf{1}[(\mathbf{s}\dot{+}S_t \neq \mathbf{d}) \wedge ((\mathbf{s} + \mathbf{e}_i)\dot{+}S_t) = \mathbf{d}]\right].$$

We claim that any sequence S_t that satisfies the conjunction must have the property that $(S_t)_i = k - s_i$. This is true because $(\mathbf{s} + \mathbf{e}_i)\dot{+}S_t = \mathbf{d}$ and therefore $(S_t)_i \geq k + 1 - s_i$. Also $(S_t)_j \geq k + 1 - s_j$, for $j \neq i$. This implies that $\mathbf{s}\dot{+}S_t = \mathbf{o}_i$ and the above difference becomes

$$\mathbb{E}\left[\textstyle\sum_{t=0}^{\infty} \mathbf{1}[\mathbf{s}\dot{+}S_t = \mathbf{o}_i]\right].$$

The last term is exactly $\widehat{p}_i(\mathbf{s})$, the probability that $\mathbf{s}\dot{+}S_t$ eventually arrives at $\mathbf{o}_i$, times the expected number of iterations spent in state $\mathbf{o}_i$ before arriving at $\mathbf{d}$. To leave $\mathbf{o}_i$, the random walk must make a step in the ith direction, and thus the expected "waiting time" at $\mathbf{o}_i$ is can be computed as

$$\sum_{q=1}^{\infty} q \ \underbrace{(1 - \frac{1}{n})^{q-1}}_{\text{prob. of } q-1 \text{ loops}} \ \underbrace{\frac{1}{n}}_{\text{prob. of leaving}} = n.$$

■

The last lemma implies an important fact about the state lattice. Interpret the state lattice as a directed graph with directed edges at all pairs $(\mathbf{s}, \mathbf{s} + \mathbf{e}_i)$ for each $i \in \lambda(\mathbf{s})$. Also associate the edge $(\mathbf{s}, \mathbf{s} + \mathbf{e}_i)$ with the survival probability $\widehat{p}_i(\mathbf{s})$. Consider starting at state $\mathbf{s}$ and walking through this directed graph:

$$\mathbf{s} \to \mathbf{s} + \mathbf{e}_{i_1} \to \mathbf{s} + \mathbf{e}_{i_1} + \mathbf{e}_{i_2} \to \ldots$$

Corollary 7 *Consider any two states $\mathbf{s}, \mathbf{s}'$. For any path from $\mathbf{s}$ to $\mathbf{s}'$ through the directed state graph, the sum of all edge weights $\widehat{p}_i(\cdot)$ along this path is* independent *of the choice of path.*

Proof: Assume the path $\mathbf{s} = \mathbf{s}^1, \mathbf{s}^2, \ldots, \mathbf{s}^T, \mathbf{s}^{T+1} = \mathbf{s}'$ defined by a sequence of moves is $i_1, i_2, \ldots, i_T$, where $\mathbf{s}^{t+1} = \mathbf{s}^t + \mathbf{e}_{i_t}$. By Lemma 6 the total weight sum is

$$\sum_{t=1}^{T} \widehat{p}_{i_t}(\mathbf{s}^t) = \sum_{t=1}^{T} \frac{1}{n}(\tau(\mathbf{s}^t) - \tau(\mathbf{s}^{t+1})) = \frac{1}{n}(\tau(\mathbf{s}) - \tau(\mathbf{s}')),$$

which is independent of the choice of path. ■

Note that in the definition of the directed state graph above and in the corollary we ignore loops, which occur when $\mathbf{s} = \mathbf{s}\dot{+}\mathbf{e}_i$ (or equivalently $i \notin \lambda(\mathbf{s})$). Such loops out of state $\mathbf{s}$ are immaterial because they correspond to dead events, and $i \notin \lambda(\mathbf{s})$ iff $\widehat{p}_i(\mathbf{s}) = 0$.

4 The Optimal Strategy

We now have the all the tools to express $\widehat{V}(\mathbf{s})$ in terms of the expected path length $\tau(\mathbf{s})$, prove that $V(\mathbf{s}) = \widehat{V}(\mathbf{s})$, and show that the optimal betting strategy for the gambler is $\widehat{p}(\mathbf{s})$.

We prove two major theorems in this section. We provide the mathematically precise argument for each but, as formality often obscures the true intuition, we also provide an "English Version" so that the reader sees a rough sketch. Our mathematical proofs require induction on the state space $\mathcal{S}$, so we need a "measure of progress" for state vectors $\mathbf{s}$. For any $\mathbf{s} \in \mathcal{S}$, define $m(\mathbf{s}) := n(k+1) - |\mathbf{s}|$, the number of steps required before reaching the dead state. Clearly $m(\mathbf{s}) = 0$ if and only if $\mathbf{s} = \mathbf{d}$.

Theorem 8 *For all states* $\mathbf{s}$*,*

$$\widehat{V}(\mathbf{s}) = \frac{1}{n}\tau(\mathbf{s}).$$

Proof: (*English Version*) Assume that the Gambler always plays according to the distribution vector $\widehat{p}(\mathbf{s})$. Then we may think of the Casino's choices as a walk around the state graph and, as we discussed at the end of Section 3, a collection of the "weights" $\widehat{p}_i(\cdot)$ along the way, ending at $\mathbf{d}$. But as we proved in Corollary 7 for the weights $\widehat{p}(.)$, it doesn't matter what path is taken: the Casino will always receive $\frac{1}{n}(\tau(\mathbf{s}) - \tau(\mathbf{d})) = \frac{1}{n}\tau(\mathbf{s})$ on any path from $\mathbf{s}$ that just ended in $\mathbf{d}$.

If the Gambler ever chooses a distribution $\mathbf{w}$ different from $\widehat{p}(\mathbf{s})$ at some state $\mathbf{s}$, then the Casino can simply let $\boldsymbol{\ell} = \mathbf{e}_j$ for any j for which $w_j > \widehat{p}_j(\mathbf{s})$, and on this round the casino will force loss *greater* than $\widehat{p}_j(\mathbf{s})$. This means that on some path starting from $\mathbf{s}$, the Casino will accrue total weight/loss larger than $\frac{1}{n}\tau(\mathbf{s})$, and therefore that the distribution $\mathbf{w}$ at $\mathbf{s}$ was non-optimal for the Gambler. We conclude that for the Gambler $\widehat{p}(.)$ is the only optimal assignment of distributions to states. ■

Proof: (*Formal Version*) We induct on $m(\mathbf{s})$. First we check the base case $\mathbf{s} = \mathbf{d}$. In this case, the expected path length is exactly 0 since we have already reached the dead state. Thus $\frac{\tau(\mathbf{s})}{n} = 0 = \widehat{V}(\mathbf{d})$ as desired.

Now assume that $m(\mathbf{s}) > 0$. Then

$$\begin{aligned}
\widehat{V}(\mathbf{s}) &= \min_{\mathbf{w}\sim\lambda(\mathbf{s})} \max_{i\in\lambda(\mathbf{s})} w_i + \widehat{V}(\mathbf{s}+\mathbf{e}_i) \\
\text{(induc.)} \quad &= \min_{\mathbf{w}\sim\lambda(\mathbf{s})} \max_{i\in\lambda(\mathbf{s})} w_i + \frac{1}{n}\tau(\mathbf{s}+\mathbf{e}_i) \\
&\leq \max_{i\in\lambda(\mathbf{s})} \widehat{p}_i(\mathbf{s}) + \frac{1}{n}\tau(\mathbf{s}+\mathbf{e}_i) \\
\text{(Lem. 6)} \quad &= \max_i \frac{1}{n}(\tau(\mathbf{s}) - \tau(\mathbf{s}+\mathbf{e}_i)) + \frac{1}{n}\tau(\mathbf{s}+\mathbf{e}_i) \\
&= \frac{1}{n}\tau(\mathbf{s}).
\end{aligned}$$

We prove $\widehat{V}(\mathbf{s}) \geq \frac{1}{n}\tau(\mathbf{s})$ by a similar induction. Assume that the Gambler chooses the optimal distribution $\mathbf{w}^*$ which may indeed be different from $\widehat{p}(\mathbf{s})$. For any $i \notin \lambda(\mathbf{s})$, $\widehat{p}_i(\mathbf{s})$ is defined as zero. For the optimal strategy $w_i^* = 0$ as well because otherwise the Casino can incurr unbounded loss by playing $\mathbf{e}_i$ repeatedly. Since $\mathbf{w}^*$ and $\widehat{p}(\mathbf{s})$ are different distributions on the live events $\lambda(\mathbf{s})$, there must exist some $j \in \lambda(\mathbf{s})$ for which $w_j^* > \widehat{p}_j(\mathbf{s})$. We now have

$$\begin{aligned}
\widehat{V}(\mathbf{s}) &= \max_{i\in\lambda(\mathbf{s})} w_i^* + \widehat{V}(\mathbf{s}+\mathbf{e}_i) \\
\text{(induc.)} \quad &= \max_{i\in\lambda(\mathbf{s})} w_i^* + \frac{1}{n}\tau(\mathbf{s}+\mathbf{e}_i) \\
&\geq w_j^* + \frac{1}{n}\tau(\mathbf{s}+\mathbf{e}_i) \\
&> \widehat{p}_j(\mathbf{s}) + \frac{1}{n}\tau(\mathbf{s}+\mathbf{e}_i) \\
\text{(Lem.6)} \quad &= \frac{1}{n}(\tau(\mathbf{s}) - \tau(\mathbf{s}+\mathbf{e}_i)) + \frac{1}{n}\tau(\mathbf{s}+\mathbf{e}_i) \\
&= \frac{1}{n}\tau(\mathbf{s}).
\end{aligned}$$

■

Corollary 9 *For any* $\mathbf{s} \neq \mathbf{d}$*,* $\widehat{p}(\mathbf{s})$ *is the unique optimal probability vector for the learner for the game related to* $\widehat{V}$*.*

Proof: See end of last proof. ■

Corollary 10 *For all* $\mathbf{s}$ *and all* $i \in [n]$*,*

$$\widehat{p}_i(\mathbf{s}) = \widehat{V}(\mathbf{s}) - \widehat{V}(\mathbf{s}\dot{+}\mathbf{e}_i)$$

Proof: This follows from the previous theorem and Lemma 6. ■

We need one more lemma before we can prove our main result.

Lemma 11 *For any state* $\mathbf{s}$ *and distinct events* $i, j \in \lambda(\mathbf{s})$*, we have*

$$\widehat{p}_i(\mathbf{s}) < \widehat{p}_i(\mathbf{s}+\mathbf{e}_j).$$

This fact is intuitive: if losses are randomly assigned then the probability that the ith event will survive last *strictly increases* when another event suffers a loss. We prove this precisely below.

Proof: To show that $\widehat{p}_i(\mathbf{s}) \leq \widehat{p}_i(\mathbf{s}+\mathbf{e}_j)$ is straightforward. Any sequence $S_0, S_1, S_2, \ldots$ that brings $\mathbf{s}$ to the one-live state $\mathbf{o}_i$ also brings $\mathbf{s}+\mathbf{e}_j$ to $\mathbf{o}_i$. Indeed, if $\mathbf{s}\dot{+}S_t = \mathbf{o}_i$ for some t then certainly $(\mathbf{s}+\mathbf{e}_j)\dot{+}S_t = \mathbf{o}_i$ as well.

To show that this inequality is strict, we need only find one random sequence for which $\mathbf{s}+\mathbf{e}_j$ is brought to $\mathbf{o}_i$ but not $\mathbf{s}$. Take any sequence $S_0, S_1, \ldots$ such that $\mathbf{s}\dot{+}S_t = \mathbf{d} - \mathbf{e}_i - \mathbf{e}_j$ (where the only events remaining are i and j) and where $S_{t+1} = S_t + \mathbf{e}_i$. Then $(\mathbf{s}+\mathbf{e}_j)\dot{+}S_t = \mathbf{o}_i$ but $\mathbf{s}\dot{+}S_{t+1} = \mathbf{s}\dot{+}(S_t+\mathbf{e}_i) = \mathbf{o}_j$. ■

Theorem 12 *For all states* $\mathbf{s}$*,*

$$V(\mathbf{s}) = \widehat{V}(\mathbf{s}) = \frac{1}{n}\tau(\mathbf{s}).$$

Proof: (*English Version*) Imagine a gambler who plays the distribution $\widehat{p}(\mathbf{s})$ at every state $\mathbf{s}$. We already know that the Casino can use its modified game strategy and simply play unit vectors $\boldsymbol{\ell} = \mathbf{e}_i$ on each round to force $\frac{1}{n}\tau(\mathbf{s})$ loss. Yet since $\boldsymbol{\ell}$ is unrestricted, can it obtain more? The answer is No: consider what happens if the Casino decides to choose $\boldsymbol{\ell}$ larger than a unit vector, e.g. let $\boldsymbol{\ell} = \mathbf{e}_i + \mathbf{e}_j$ for simplicity. Then on this round it obtains $\widehat{p}_i(\mathbf{s}) + \widehat{p}_j(\mathbf{s})$, but it can do better! We proved in Lemma 11 that survival probabilities strictly increase and therefore $\widehat{p}_i(\mathbf{s}) < \widehat{p}_i(\mathbf{s}+\mathbf{e}_j)$. Thus, a more patient Casino could choose $\boldsymbol{\ell} = \mathbf{e}_j$ on this round, obtain $\widehat{p}_j(\mathbf{s})$, and then choose $\boldsymbol{\ell} = \mathbf{e}_i$ on the next round to obtain $\widehat{p}_i(\mathbf{s}+\mathbf{e}_j)$. As $\widehat{p}_j(\mathbf{s}) + \widehat{p}_i(\mathbf{s}+\mathbf{e}_j) > \widehat{p}_j(\mathbf{s}) + \widehat{p}_i(\mathbf{s})$, the Casino only does worse by playing non-unit vectors. Indeed, this suggests that the Gambler has a strategy by which the Casino can inflict only as much loss as in the modified game, and thus the value $V(\mathbf{s})$ is no different from $\widehat{V}(\mathbf{s})$. ■

Proof: (*Formal Version*) Certainly $V(\mathbf{s}) \geq \widehat{V}(\mathbf{s})$, since the Casino is given strictly fewer choices in the modified game. Thus we are left to show that $V(\mathbf{s}) \leq \widehat{V}(\mathbf{s})$. We proceed via induction on $m(\mathbf{s})$. By definition, $V(\mathbf{s}) = \widehat{V}(\mathbf{s})$ for the case $\mathbf{s} = \mathbf{d}$. Now assume that, for all successive states $\mathbf{s}'$ where $m(\mathbf{s}') < m(\mathbf{s})$, $V(\mathbf{s}') = \widehat{V}(\mathbf{s}')$. We proceed by directly analyzing the recursive definition (2). Assume that the Gambler has chosen the (possibly non-optimal) distribution $\mathbf{w} = \widehat{p}(\mathbf{s})$ to distribute his wealth on the live events $\lambda(\mathbf{s})$, and let $\boldsymbol{\ell}^* \in \{0,1\}^n$ be an optimal choice of the Casino (which can depend on the Gambler's choice). By definition (1) of $V(\mathbf{s})$, the chosen loss vector can't be $\mathbf{0}$ and all events with loss one must be in $\lambda(\mathbf{s})$. More precisely,

$$\begin{aligned}
V(\mathbf{s}) &= \min_{\mathbf{w}\sim\lambda(\mathbf{s})} \max_{\mathbf{0}\neq\boldsymbol{\ell}\subset\lambda(\mathbf{s})} \mathbf{w}\cdot\boldsymbol{\ell} + V(\mathbf{s}+\boldsymbol{\ell}) \\
\text{(ind.)} &= \min_{\mathbf{w}\sim\lambda(\mathbf{s})} \max_{\mathbf{0}\neq\boldsymbol{\ell}\subset\lambda(\mathbf{s})} \mathbf{w}\cdot\boldsymbol{\ell} + \widehat{V}(\mathbf{s}+\boldsymbol{\ell}) \\
&\leq \max_{\mathbf{0}\neq\boldsymbol{\ell}\subset\lambda(\mathbf{s})} \widehat{p}(\mathbf{s})\cdot\boldsymbol{\ell} + \widehat{V}(\mathbf{s}+\boldsymbol{\ell}) \\
&= \widehat{p}(\mathbf{s})\cdot\boldsymbol{\ell}^* + \widehat{V}(\mathbf{s}+\boldsymbol{\ell}^*)
\end{aligned}$$

If $\boldsymbol{\ell}^*$ is any unit vector $\mathbf{e}_i$, s.t. $i \in \lambda(\mathbf{s})$, then

$$\begin{aligned}
V(\mathbf{s}) &\leq \widehat{p}(\mathbf{s})\cdot\mathbf{e}_i + \widehat{V}(\mathbf{s}+\mathbf{e}_i) \\
&= \widehat{p}_i(\mathbf{s}) + \widehat{V}(\mathbf{s}+\mathbf{e}_i) \;=\; \widehat{V}(\mathbf{s})
\end{aligned}$$

and in this case, $V(\mathbf{s}) = \widehat{V}(\mathbf{s})$ and we are done. We now prove by contradiction that $\boldsymbol{\ell}^*$ can have no more than one non-zero coordinate. Assume indeed that $|\boldsymbol{\ell}^*| > 1$, i.e. it admits a decomposition $\boldsymbol{\ell}^* = \mathbf{e}_i + \bar{\boldsymbol{\ell}}$ for some i and bit vector $\bar{\boldsymbol{\ell}} \neq \mathbf{0}$ with $\bar{\ell}_i = 0$. Applying Lemma 11 repeatedly, we have that $\widehat{p}_i(\mathbf{s}) < \widehat{p}_i(\mathbf{s}+\bar{\boldsymbol{\ell}})$ and therefore

$$\begin{aligned}
&\widehat{p}(\mathbf{s})\cdot\boldsymbol{\ell}^* + \widehat{V}(\mathbf{s}+\boldsymbol{\ell}^*) \\
&\quad= \widehat{p}_i(\mathbf{s}) + \widehat{p}(\mathbf{s})\cdot\bar{\boldsymbol{\ell}} + \widehat{V}(\mathbf{s}+\boldsymbol{\ell}^*) \\
\text{(Lem. 11)} &\quad< \widehat{p}_i(\mathbf{s}+\bar{\boldsymbol{\ell}}) + \widehat{p}(\mathbf{s})\cdot\bar{\boldsymbol{\ell}} + \widehat{V}(\mathbf{s}+\boldsymbol{\ell}^*) \\
\text{(Cor. 10)} &\quad= \widehat{V}(\mathbf{s}+\bar{\boldsymbol{\ell}}) - \widehat{V}(\mathbf{s}+\boldsymbol{\ell}^*) + \widehat{p}(\mathbf{s})\cdot\bar{\boldsymbol{\ell}} + \widehat{V}(\mathbf{s}+\boldsymbol{\ell}^*) \\
&\quad= \widehat{p}(\mathbf{s})\cdot\bar{\boldsymbol{\ell}} + \widehat{V}(\mathbf{s}+\bar{\boldsymbol{\ell}}).
\end{aligned}$$

But the statement $\widehat{p}(\mathbf{s})\cdot\boldsymbol{\ell}^* + \widehat{V}(\mathbf{s}+\boldsymbol{\ell}^*) < \widehat{p}(\mathbf{s})\cdot\bar{\boldsymbol{\ell}} + \widehat{V}(\mathbf{s}+\bar{\boldsymbol{\ell}})$ implies $\boldsymbol{\ell}^*$ is a non-optimal choice for the Casino and this contradicts our assumption that $\boldsymbol{\ell}^*$ was optimum. ■

Corollary 13 *For any* $\mathbf{s} \neq \mathbf{d}$*, if the learner plays with the optimum probability vector* $\widehat{p}(\mathbf{s})$*, then the only optimal responses of the adversary in the recurrence* (2) *for* V *is to choose a unit vector of a live event.*

Proof: Proved at the end of the last theorem. ■

5 Recurrences, Combinatorics and Randomized Algorithms

The quantities $V(\mathbf{s})$, $\tau(\mathbf{s})$ and $\widehat{p}_i(\mathbf{s})$ have a number of interesting properties that we lay out in this section.

5.1 Some Recurrences

The expected path length, $\tau(\mathbf{s})$ satisfies a very natural recursion. When $\mathbf{s} = \mathbf{d}$, then the path length is deterministically 0 and therefore $\tau(\mathbf{d}) = 0$. Otherwise, we see that the expected path length is

$$\tau(\mathbf{s}) = 1 + \frac{\sum_{i=1}^n \tau(\mathbf{s}\dot{+}\mathbf{e}_i)}{n}. \tag{5}$$

That is, the expected path length is 1, for the current step in the path, plus the expected path length of the next random state. Since the next state is chosen randomly from the set $\{\mathbf{s}\dot{+}\mathbf{e}_i : i = 1, \ldots, n\}$, the probability of any given state is $\frac{1}{n}$, hence the normalization factor.

Of course, our original quantity of interest is $V(\mathbf{s})$, and as we showed in Theorem 12 $V(\mathbf{s}) = \frac{1}{n}\tau(\mathbf{s})$. This immediately gives us a recursion for V:

$$\begin{aligned}
V(\mathbf{s}) &= \frac{1}{n}\left(1 + \frac{1}{n}\sum_{i=1}^n \tau(\mathbf{s}\dot{+}\mathbf{e}_i)\right) \\
&= \frac{1 + \sum_{i=1}^n V(\mathbf{s}\dot{+}\mathbf{e}_i)}{n}.
\end{aligned}$$

This recurrence, while true for the function $V(\cdot)$, is ambiguous because $V(\mathbf{s})$ can occur on both sides of the equation. Indeed, whenever $i \notin \lambda(\mathbf{s})$, $V(\mathbf{s}\dot{+}\mathbf{e}_i) = V(\mathbf{s})$. However, we can rearrange all $V(\mathbf{s})$ terms to obtain the following well-defined recursion:

$$V(\mathbf{s}) = \frac{1 + \sum_{i\in\lambda(\mathbf{s})} V(\mathbf{s}+\mathbf{e}_i)}{|\lambda(\mathbf{s})|}. \tag{6}$$

We can find a similar recurrence for $\widehat{p}_i(\cdot)$. For the one-live states $\mathbf{o}_i$ we have $\widehat{p}_j(\mathbf{o}_i) = 1$ if $i = j$ and 0 otherwise. If $|\lambda(\mathbf{s})| > 1$, then

$$\widehat{p}_i(\mathbf{s}) = \frac{\sum_{j=1}^{n} \widehat{p}_i(\mathbf{s}\dot{+}\mathbf{e}_j)}{n}.$$

As $\widehat{p}_i(\mathbf{s})$ is the probability of ending at state $\mathbf{o}_i$ after executing the Markov chain, this formula is obtained by conditioning on one step of the Markov process. That is, the probability of ending at state $\mathbf{o}_i$ is

$$\sum_j Pr(j \text{ chosen}) Pr(\text{random process takes } \mathbf{s}\dot{+}\mathbf{e}_j \text{ to } \mathbf{o}_i).$$

This recurrence suffers from the same problem as did our initial recurrence for $V(\cdot)$: $\widehat{p}(\mathbf{s})$ can occur on both sides of the equality. We again solve this problem by rearranging terms and obtain

$$\widehat{p}_i(\mathbf{s}) = \frac{\sum_{j \in \lambda(\mathbf{s})} \widehat{p}_i(\mathbf{s} + \mathbf{e}_j)}{|\lambda(\mathbf{s})|}.$$

5.2 Combinatorial Sums

A further analysis gives us exact expressions for both $\widehat{p}_i(\mathbf{s})$ and $V(\mathbf{s})$ in terms of infinite sums of multinomials.

Proposition 5.1 *For any state* $\mathbf{s} \in \mathcal{S}$,

$$\widehat{p}_i(\mathbf{s}) = \sum_{\mathbf{r}:\mathbf{s}\dot{+}\mathbf{r}=\mathbf{o}_i} \binom{|\mathbf{r}|}{r_1, r_2, \ldots, r_n} \left(\frac{1}{n}\right)^{|\mathbf{r}|+1}.$$

Proof: By definition, $\widehat{p}_i(\mathbf{s})$ is the probability that $\mathbf{s}$ reaches the one-live state $\mathbf{o}_i$ eventually. To compute this probability, we consider at what point the Markov process *exits* the state $\mathbf{o}_i$ and into $\mathbf{d}$. Recall the random variable S_t defined in Section 3. Take any $\mathbf{r}$ for which $\mathbf{s}\dot{+}\mathbf{r} = \mathbf{o}_i$ and condition on $S_t = \mathbf{r}$. Then

$$\widehat{p}_i(\mathbf{s}) = \sum_{\mathbf{r}:\mathbf{s}\dot{+}\mathbf{r}=\mathbf{o}_i} Pr(S_t = \mathbf{r}) Pr(S_{t+1} = \mathbf{r} + \mathbf{e}_i | S_t = \mathbf{r})$$

The first probability is exactly $\binom{|\mathbf{r}|}{r_1, r_2, \ldots, r_n} n^{-|\mathbf{r}|}$ and the second probability is exactly $1/n$. ■

Since $V(\mathbf{s})$ can be written as an expected path length, we can obtain a similar expression as a sum of multinomials for $V(\mathbf{s})$:

Proposition 5.2

$$V(\mathbf{s}) = \sum_{i=1}^{n} \sum_{\mathbf{r}:\mathbf{r}\dot{+}\mathbf{s}=\mathbf{o}_i} (|\mathbf{r}| + 1) \binom{|\mathbf{r}|}{r_1, r_2, \ldots, r_n} \left(\frac{1}{n}\right)^{|\mathbf{r}|+1}.$$

5.3 Randomized Approximations

Computing the exact value $V(\mathbf{s})$ for large but non-asymptotic values of the state vector is difficult because we have no polynomial time algorithm. On the other hand, finding a randomized approximation to $V(\mathbf{s})$ can be done very efficiently. Indeed, as we now have a representation of $V(\mathbf{s})$ in terms of the length of a random walk, we can simply run the random walk $S_1, S_2, \ldots$ several times, note the length $T(\mathbf{s})$, and return the mean. Such random approximations require that the distribution on $T(\mathbf{s})$ has low-variance, yet this certainly holds in the case at hand. While the random walk requires at least $n(k+1)$ iterations to finish, a simple argument shows that with probability $1-\delta$ the random walk completes in less than $nk \log(nk/\delta)$ rounds.

Algorithm 1 Random Approximation to $V(\mathbf{s})$

```
Input: state s
t ← 0
for i = 1, ..., NUMITER do
  z ← s
  repeat
    Sample i ∈ {1, ..., n} u.a.r.
    z ← z ∔ e_i
    t ← t + 1
  until z = d
end for
Return t / (n · NUMITER).
```

If $R(\mathbf{s})$ is the random variable returned by the above algorithm, then clealy $\mathbb{E}R(\mathbf{s}) = V(\mathbf{s})$. By increasing NUMITER, the variance of this estimate can be reduced quickly.

A randomized approximation for $\widehat{p}(\mathbf{s})$ can be obtained similarly. Again the above algorithm approximately computes $\widehat{p}(\mathbf{s})$ in the following sense: If $R(\mathbf{s})$ is the random variable returned by the above algorithm, then clearly $\mathbb{E}R(\mathbf{s}) = \widehat{p}(\mathbf{s})$. Again increasing NUMITER, reduces the variance of the estimate.

Algorithm 2 Random Approximation to $\widehat{p}(\mathbf{s})$

```
Input: state s ≠ d
p ← 0
for i = 1, ..., NUMITER do
  z ← s
  repeat
    Sample i ∈ {1, ..., n} u.a.r.
    z ← z ∔ e_i
  until z = o_j for some j
  p ← p + e_j
end for
Return p / NUMITER.
```

5.4 A Simple Strategy in a Randomized Setting

In the particular case of betting against the Casino, it may be necessary for the Gambler to compute $\widehat{p}_i(\mathbf{s})$ in order to place his bets optimally. In an alternative setting, however, a randomized algorithm may be sufficient. Let us consider the case in which the Gambler chooses to bet according to the outcome of several coin tosses. Further assume that the Casino can observe his strategy but cannot see the outcome of the coin tosses or his final bets. In this scenario, the Gambler can even bet all of his money on a random event $I \in \{1, \ldots, n\}$ drawn according to some distribution as long

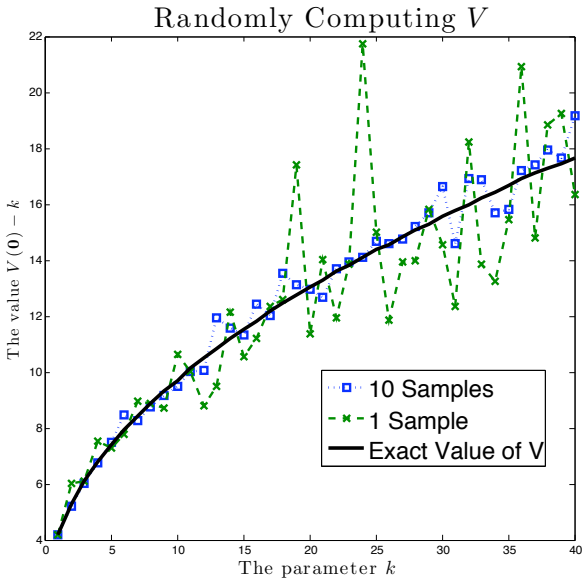

Figure 1: We illustrate the accuracy of the randomized approximation to $V(\mathbf{0})$ stated in Algorithm 1. The plot compares the exact value of $V(\mathbf{0})$ to that obtained by using either 1 or 10 samples of the random walk. Here $n = 100$.

as $\mathbb{E}\mathbf{1}[I = i] = \widehat{p}_i(\mathbf{s})$ for all i, and indeed his expected loss would be $\widehat{p}(\mathbf{s}) \cdot \ell$.

For this scenario, randomly approximating $\widehat{p}$ is not necessary: *only one sample is needed!* To be precise, the Gambler can take the state $\mathbf{s}$, run the random walk until the state reaches $\mathbf{o}_i$ for some i, and then bet his full dollar on event i. This bet will be correct in expectation, i.e. he will pick event i with probability $\widehat{p}_i(\mathbf{s})$, and thus his expected loss will be exactly $\widehat{p} \cdot \ell$. The key here is that sampling from the distribution $\widehat{p}(\mathbf{s})$ may be quite easy even when computing it exactly may take more time.

Note that the above method based on one sample is similar to the way the Randomized Weighted Majority algorithm approximates the Weighted Majority algorithm (more precisely the WMC algorithm of [LW94]). More precisely `NUMITER=1` of Algorithm 5.3 corresponds to WMR, and `NUMITER` $\to \infty$ corresponds to WMC.

6 Comparison to Previous Bounds

As mentioned in the introduction, the bound obtainable base on exponential weights [FS97] is

$$k + \sqrt{2k \log n} + \log n \tag{7}$$

and can be shown to be asymptotically optimal [Vov98].[2] Having computed the minimax solution to the same game, we can compute the game-theoretically optimal bound of $V(\mathbf{0})$ using Algorithm 1. For small values of n and k, these bounds do differ quite substantially. We present in Figure 2 a comparison of the regret for $n = 2, 10, 100$ and $k = 1, \ldots, 20$.

[2] A slightly better but more complicated bound than (7) was given in [Vov98]. In the full paper we compare the optimal bound to this one as well.

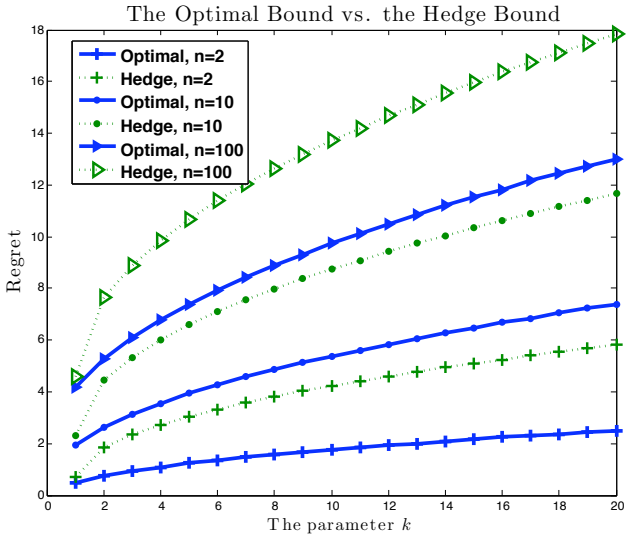

Figure 2: We compare the optimal regret bound we obtain from $V(\mathbf{0})$ to that found in [FS97], which we refer to as the hedge bound. While asymptotically optimal, we observed that the hedge bound of $k + \sqrt{2k \log n} + \log n$ is not tight for small values of n and k.

7 Connections to classic problems of probabilistic enumerative combinatorics.

Theorem 12 shows that an optimal strategy for the Casino requires unit vector plays. This leads to alternative interpretations of the game in terms of well studied random processes.

For example, one can easily confirm that our game also describes the random process underlying a generalized form of the Coupon Collector's Problem [**?**] in which the collector buys cereal boxes one by one in order to obtain $K = k+1$ complete sets of n baseball cards, assuming one card is randomly placed within each cereal box. The value of our game, $V(0, 0)$, is in fact the expected number of cereal boxes, per baseball card, needed to obtain the desired K complete sets.

Specifically, the probability generating function for the generalized Coupon Collector's Problem is [MW03]

$$G_{n,K}(z) = \frac{n}{(K-1)!} \int_0^\infty e^{-nt/z} t^{K-1} \left[\sum_{j \geq k} \frac{t^j}{j!} \right]^{n-1} dt.$$

Taking the derivative at $z = 1$ and dividing by n, we derive the expected number of steps to obtain K sets, which is also the value of our game, viz.

$$V(0^n) = \frac{n}{(K-1)!} \int_0^\infty t^K e^{-nt} \left[\sum_{j \geq k} \frac{t^j}{j!} \right]^{n-1} dt. \tag{8}$$

Equation (8) gives us an elegant closed form for the two-card case ($n = 2$):

$$V(\langle 0, 0 \rangle) = K + \frac{K}{2^{2K}} \binom{2K}{K}$$

From (8) we also obtain the well known asymptotic expression for the value, for large n and fixed K,

$$V(0^n) \rightarrow_{n\rightarrow\infty} \log n + (K-1)\log\log n[1+o(1)].$$

The same asymptotic form appears in the analysis of an evolving random graph. [ER60] The random walk on the state lattice provides yet another interpretation of the same dynamics.

For $K >> n >> 1$, the law of large numbers gives [NS60]

$$V(\langle 0^n \rangle) = K + O(K^{1/2}).$$

8 Conclusion

We showed in Corollary 13 that against the optimal learning algorithm the optimal strategy of the adversary is to choose one of the unit loss vectors as his response. Curiously enough it can be show that this is also true of the Weighted Majority algorithm (1). That is, any trial in which $q > 1$ experts incurred a unit of loss can be split into q trials in which a single expert has a unit of loss, and doing this always increases the loss of the algorithm for all update factor $\beta \in [0, 1)$. This observation about the Weighted Majority algorithm might actually lead to improved loss bounds for this algorithm, perhaps in the way the parameter β is tuned.

There remains also a deep question regarding the techniques introduced in this paper: how general is this method of computing the value of a game based on a random path? Can it handle slightly more involved problems? Examples we have considered include competing against m-sized sets of experts, discussed in [WK06], in which the loss of the algorithm is compared to the loss of the best m-subset. Another example is the problem of competing against permutations of n objects [HW07], where the loss of a permutation is linearly assigned. Our preliminary investigation suggests that similar techniques can be adapted to also handle such more complex problems. In the full paper we hope to delineate the scope of our new method of optimal algorithm design.

References

[ALW07] J. Abernethy, J. Langford, and M. K. Warmuth. Continuous experts and the Binning algorithm. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT06)*, pages 544–558. Springer, June 2007.

[CBFH+97] Nicolò Cesa-Bianchi, Yaov Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *J. ACM*, 44(3):427–485, 1997.

[CBFHW96] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, and M. K. Warmuth. On-line prediction and conversion strategies. *Machine Learning*, 25:71–110, 1996.

[ER60] P. Erdos and A. Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5A:17–61, 1960.

[FS97] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to Boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997. Special Issue for EuroCOLT '95.

[HW07] D. Helmbold and M. K. Warmuth. Learning permutations with exponential weights. In *Proceedings of the 20th Annual Conference on Learning Theory (COLT07)*. Springer, 2007.

[KW99] Jyrki Kivinen and Manfred K. Warmuth. Averaging expert predictions. In *Computational Learning Theory, 4th European Conference, EuroCOLT '99, Nordkirchen, Germany, March 29-31, 1999, Proceedings*, volume 1572 of *Lecture Notes in Artificial Intelligence*, pages 153–167. Springer, 1999.

[LW94] N. Littlestone and M. K. Warmuth. The Weighted Majority algorithm. *Inform. Comput.*, 108(2):212–261, 1994. Preliminary version in in FOCS 89.

[MW03] A. Myers and H. S. Wilf. Some new aspects of the Coupon-Collector's problem. *SIAM J. Disc. Math.*, 17:1–17, 2003.

[NS60] D. Newman and L. Shepp. The Double Dixie Cup problem. *Amer. Math Monthly.*, 67:541–574, 1960.

[Vov98] V. Vovk. A game of prediction with expert advice. *J. of Comput. Syst. Sci.*, 56(2):153–173, 1998. Special Issue: Eighth Annual Conference on Computational Learning Theory.

[WK06] M. K. Warmuth and D. Kuzmin. Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. In *Advances in Neural Information Processing Systems 19 (NIPS 06)*. MIT Press, December 2006.