Complexity of Combinatorial Market Makers*

Yiling Chen
Yahoo! Research
111 W. 40th St., 17th Floor
New York, NY 10018

Lance Fortnow
EECS Department
Northwestern University
2133 Sheridan Road
Evanston, IL 60208

Nicolas Lambert
Department of Computer Science
Stanford University
Stanford, CA 94305

David M. Pennock
Yahoo! Research
111 W. 40th St., 17th Floor
New York, NY 10018

Jennifer Wortman
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

ABSTRACT

We analyze the computational complexity of market maker pricing algorithms for combinatorial prediction markets. We focus on Hanson's popular logarithmic market scoring rule market maker (LMSR). Our goal is to implicitly maintain correct LMSR prices across an exponentially large outcome space. We examine both permutation combinatorics, where outcomes are permutations of objects, and Boolean combinatorics, where outcomes are combinations of binary events. We look at three restrictive languages that limit what traders can bet on. Even with severely limited languages, we find that LMSR pricing is #P-hard, even when the same language admits polynomial-time matching without the market maker. We then propose an approximation technique for pricing permutation markets based on an algorithm for online permutation learning. The connections we draw between LMSR pricing and the literature on online learning with expert advice may be of independent interest.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Economics, Theory

Keywords

Prediction markets, logarithmic market scoring rule market makers, online learning with expert advice

*Part of this work was done while L. Fortnow, N. Lambert and J. Wortman were visiting Yahoo! Research, New York.

EC'08, July 8–12, 2008, Chicago, Illinois, USA.

1. INTRODUCTION

One way to elicit information is to ask people to bet on it. A *prediction market* is a common forum where people bet with each other or with a market maker [9, 10, 23, 20, 21]. A typical binary prediction market allows bets along one dimension, for example either for or against Hillary Clinton to win the 2008 US Presidential election. Thousands of such one- or small-dimensional markets exist today, each operating independently. For example, at the racetrack, betting on a horse to win does not directly impact the odds for that horse to finish among the top two, as logically it should, because the two bet types are handled separately.

A *combinatorial prediction market* is a central clearinghouse for handling logically-related bets defined on a combinatorial space. For example, the outcome space might be all $n!$ possible permutations of n horses in a horse race, while bets are properties of permutations such as "horse A finishes 3rd" or "horse A beats horse B." Alternately, the outcome space might be all 2^{50} possible state-by-state results for the 2008 US Presidential election, while bets are Boolean statements such as "Democrat wins in Ohio and Florida but not in Texas."

Low liquidity marginalizes the value of prediction markets, and combinatorics only exacerbates the problem by dividing traders' attention among an exponential number of outcomes. A combinatorial matching market—the combinatorial generalization of a standard double auction—may simply fail to find any trades [11, 4, 5].

In contrast, an *automated market maker* is always willing to trade on *every* bet at some price. A combinatorial market maker implicitly or explicitly maintains prices across all (exponentially many) outcomes, thus allowing any trader at any time to place any bet, if transacted at the market maker's quoted price.

Hanson's [13, 14] logarithmic market scoring rule market maker (LMSR) is becoming the de facto standard market maker for prediction markets, largely because it has a number of desirable properties, including bounded loss that grows logarithmically in the number of outcomes, infinite liquidity, and modularity that respects some independence relationships. LMSR is used by a number of companies, including Microsoft, inklingmarkets.com, thewsx.com, and

yoonew.com, and is the subject of a number of research studies [7, 15, 8].

In this paper, we analyze the computational complexity of LMSR in several combinatorial betting scenarios. We examine both permutation combinatorics and Boolean combinatorics. We show that both computing instantaneous prices and computing payments of transactions are #P-hard in all cases we examine, even when we restrict participants to very simplistic and limited types of bets. For example, in the horse race analogy, if participants can place bets only of the form "horse A finishes in position N", then pricing these bets properly according to LMSR is #P-hard, even though matching up bets of the exact same form (with no market maker) can be done in polynomial time [4].

On a more positive note, we examine an approximation algorithm for LMSR pricing in permutation markets that makes use of powerful techniques from the literature on online learning with expert advice [3, 19, 12]. We briefly review this online learning setting, and examine the striking parallels that exist between the specific form of standard LMSR prices and the expert weights employed by the Weighted Majority algorithm [19]. We then show how a recent extension of Weighted Majority to permutation learning [16] can be transformed into an approximation algorithm for pricing in permutation markets in which the market maker is guaranteed to have bounded loss.

2. RELATED WORK

Fortnow et al. [11] study the computational complexity of finding acceptable trades among a set of bids in a Boolean combinatorial market. In their setting, the center is an *auctioneer* who takes no risk, only matching together willing traders. They study a call market setting in which bids are collected together and processed once en masse. They show that the auctioneer matching problem is co-NP-complete when orders are divisible and Σ_2^p-complete when orders are indivisible, but identify a tractable special case in which participants are restricted to bet on disjunctions of positive events or single negative events.

Chen et al. [4] analyze the the auctioneer matching problem for betting on permutations, examining two bidding languages. *Subset bets* are bets of the form "candidate i finishes in positions x, y, or z" or "candidate i, j, or k finishes in position x." *Pair bets* are of the form "candidate i beats candidate j." They give a polynomial-time algorithm for matching divisible subset bets, but show that matching pair bets is NP-hard.

Hanson highlights the use of LMSR for Boolean combinatorial markets, noting that the subsidy required to run a combinatorial market on 2^n outcomes is no greater than that required to run n independent one-dimensional markets [13, 14]. He discusses the computational difficulty of maintaining LMSR prices on a combinatorial space, and proposes running market makers on overlapping subsets of events, allowing traders to synchronize the markets via arbitrage.

The work closest to our own is that of Chen, Goel, and Pennock [6], who study a special case of Boolean combinatorics in which participants bet on how far a team will advance in a single elimination tournament, for example a sports playoff like the NCAA college basketball tournament. They provide a polynomial-time algorithm for LMSR pricing in this setting based on a Bayesian network representation of prices. They also show that LMSR pricing is NP-hard for a very general bidding language. They suggest an approximation scheme based on Monte Carlo simulation or importance sampling. We believe ours are the first non-trivial hardness results and worst-case bounded approximation scheme for LMSR pricing.

3. BACKGROUND

3.1 Logarithmic Market Scoring Rules

Proposed by Hanson [13, 14], a logarithmic market scoring rule is an automated market maker mechanism that always maintains a consistent probability distribution over an outcome space Ω reflecting the market's estimate of the likelihood of each outcome. A generic LMSR offers a security corresponding to each possible outcome ω. The security associated to outcome ω pays off \$1 if the outcome ω happens, and \$0 otherwise. Let $\boldsymbol{q} = (q_\omega)_{\omega\in\Omega}$ indicate the number of outstanding shares for all securities. The LMSR market maker starts the market with some initial shares of securities, $\boldsymbol{q^0}$, which may be $\mathbf{0}$. The market keeps track of the outstanding shares of securities $\boldsymbol{q}$ at all times, and maintains a cost function

$$C(\boldsymbol{q}) = b \log \sum_{\omega\in\Omega} \mathrm{e}^{q_\omega/b}, \tag{1}$$

and an instantaneous price function for each security

$$p_\omega(\boldsymbol{q}) = \frac{\mathrm{e}^{q_\omega/b}}{\sum_{\tau\in\Omega} \mathrm{e}^{q_\tau/b}}, \tag{2}$$

where b is a positive parameter related to the depth of the market. The cost function captures the total money wagered in the market, and $C(\boldsymbol{q^0})$ reflects the market maker's maximum subsidy to the market. The instantaneous price function $p_\omega(\boldsymbol{q})$ gives the current cost of buying an infinitely small quantity of the security for outcome ω, and is the partial derivative of the cost function, i.e. $p_\omega(\boldsymbol{q}) = \partial C(\boldsymbol{q})/\partial q_\omega$. We use $\boldsymbol{p} = (p_\omega(\boldsymbol{q}))_{\omega\in\Omega}$ to denote the price vector. Traders buy and sell securities through the market maker. If a trader wishes to change the number of outstanding shares from $\boldsymbol{q}$ to $\tilde{\boldsymbol{q}}$, the cost of the transaction that the trader pays is $C(\tilde{\boldsymbol{q}}) - C(\boldsymbol{q})$, which equals the integral of the price functions following any path from $\boldsymbol{q}$ to $\tilde{\boldsymbol{q}}$.

When the outcome space is large, it is often natural to offer only compound securities on sets of outcomes. A compound security S pays \$1 if one of the outcomes in the set $S \subset \Omega$ occurs and \$0 otherwise. Such a security is the combination of all securities $\omega \in S$. Buying or selling q shares of the compound security S is equivalent to buying or selling q shares of each security $\omega \in S$. Let Θ denote the set of all allowable compound securities. Denote the outstanding shares of all compound securities as $Q = (q_S)_{S\in\Theta}$. The cost function can be written as

$$\begin{aligned} C(Q) &= b \log \sum_{\omega\in\Omega} \mathrm{e}^{\sum_{S\in\Theta:\omega\in S} q_S/b} \\ &= b \log \sum_{\omega\in\Omega} \prod_{S\in\Theta:\omega\in S} \mathrm{e}^{q_S/b}. \end{aligned} \tag{3}$$

The instantaneous price of a compound security S is computed as the sum of the instantaneous prices of the securities

that compose the compound security S,

$$\begin{aligned} p_S(Q) &= \frac{\sum_{\omega\in S} \mathrm{e}^{q_\omega/b}}{\sum_{\tau\in\Omega} \mathrm{e}^{q_\tau/b}} \\ &= \frac{\sum_{\omega\in S} \mathrm{e}^{\sum_{S'\in\Theta:\omega\in S'} q_{S'}/b}}{\sum_{\tau\in\Omega} \mathrm{e}^{\sum_{S'\in\Theta:\tau\in S'} q_{S'}/b}} \\ &= \frac{\sum_{\omega\in S} \prod_{S'\in\Theta:\omega\in S'} \mathrm{e}^{q_{S'}/b}}{\sum_{\tau\in\Omega} \prod_{S'\in\Theta:\tau\in S'} \mathrm{e}^{q_{S'}/b}} \ . \end{aligned} \quad (4)$$

Logarithmic market scoring rules are so named because they are based on *logarithmic scoring rules*. A logarithmic scoring rule is a set of reward functions

$$\{s_\omega(\boldsymbol{r}) = a_\omega + b\log(r_\omega) : \omega \in \Omega\},$$

where $\boldsymbol{r} = (r_\omega)_{\omega\in\Omega}$ is a probability distribution over Ω, and a_ω is a free parameter. An agent who reports $\boldsymbol{r}$ is rewarded $s_\omega(\boldsymbol{r})$ if outcome ω happens. Logarithmic scoring rules are *proper* in the sense that when facing them a risk-neutral agent will truthfully report his subjective probability distribution to maximize his expected reward. A LMSR market can be viewed as a sequential version of logarithmic scoring rule, because by changing market prices from $\boldsymbol{p}$ to $\tilde{\boldsymbol{p}}$ a trader's net profit is $s_\omega(\tilde{\boldsymbol{p}}) - s_\omega(\boldsymbol{p})$ when outcome ω happens. At any time, a trader in a LMSR market is essentially facing a logarithmic scoring rule.

LMSR markets have many desirable properties. They offer consistent pricing for combinatorial events. As market maker mechanisms, they provide infinite liquidity by allowing trades at any time. Although the market maker subsidizes the market, he is guaranteed a worst-case loss no greater than $C(\boldsymbol{q^0})$, which is $b\log n$ if $|\Omega| = n$ and the market starts with 0 share of every security. In addition, it is a dominant strategy for a myopic risk-neutral trader to reveal his probability distribution truthfully since he faces a proper scoring rule. Even for forward-looking traders, truthful reporting is an equilibrium strategy when traders' private information is independent conditional on the true outcome [7].

3.2 Complexity of Counting

The well-known class NP contains questions that ask whether a search problem has a solution, such as whether a graph is 3-colorable. The class #P consists of functions that *count* the number of solutions of NP search questions, such as the number of 3-colorings of a graph.

A function g is #P-hard if, for every function f in #P, it is possible to compute f in polynomial time given an oracle for g. Clearly oracle access to such a function g could additionally be used to solve any NP problem, but in fact one can solve much harder problems too. Toda [24] showed that every language in the polynomial-time hierarchy can be solved efficiently with access to a #P-hard function.

To show a function g is a #P-hard function, it is sufficient to show that a function f reduces to g where f was previously known to be #P-hard. In this paper we use the following #P-hard functions to reduce from:

- **Permanent**: The permanent of an n-by-n matrix $A = (a_{i,j})$ is defined as

$$\mathrm{perm}(A) = \sum_{\sigma\in\Omega}\prod_{i=1}^{n} a_{i,\sigma(i)} \ , \quad (5)$$

where Ω is the set of all permutations over $\{1, 2, ..., n\}$. Computing the permanent of a matrix A containing only 0-1 entries is #P-hard [25].

- **#2-SAT**: Counting the number of satisfying assignments of a formula given in conjunctive normal form with each clause having two literals is #P-hard [26].

- **Counting Linear Extensions**: Counting the number of total orders that extend a partial order given by a directed graph is #P-hard [2].

#P-hardness is the best we can achieve since all the functions in this paper can themselves be reduced to some other #P function.

4. LMSR FOR PERMUTATION BETTING

In this section we consider a particular type of market combinatorics in which the final outcome is a ranking over n competing candidates. Let the set of candidates be $\mathcal{N}_n = \{1, \ldots, n\}$, which is also used to represent the set of positions. In the setting, Ω is the set of all permutations over $\mathcal{N}_n$. An outcome $\sigma \in \Omega$ is interpreted as the scenario in which each candidate i ends up in position $\sigma(i)$. Chen et al. [4] propose two betting languages, *subset betting* and *pair betting*, for this type of combinatorics and analyze the complexity of the auctioneer's order matching problem for each. In what follows we address the complexity of operating an LMSR market for both betting languages.

4.1 Subset Betting

As in Chen et al. [4], participants in a LMSR market for subset betting may trade two types of compound securities: (1) a security of the form $\langle i|\Phi\rangle$ where $\Phi \subset \mathcal{N}_n$ is a subset of positions; and (2) a security $\langle\Psi|j\rangle$ where $\Psi \subset \mathcal{N}_n$ is a subset of candidates. The security $\langle i|\Phi\rangle$ pays off \$1 if candidate i stands at a position that is an element of Φ and \$0 otherwise. Similarly, the security $\langle\Psi|j\rangle$ pays off \$1 if any of the candidates in Ψ finishes at position j and \$0 otherwise. For example, in a horse race, participants can trade securities of the form "horse A will come in the second, fourth, or fifth place," or "either horse B or horse C will come in the third place."

Note that owning one share of $\langle i|\Phi\rangle$ is equivalent to owning one share of $\langle i|j\rangle$ for every $j \in \Phi$, and similarly owning one share of $\langle\Psi|j\rangle$ is equivalent to owing one share of $\langle i|j\rangle$ for every $i \in \Psi$. We therefore restrict our attention to a simplified market where securities traded are of the form $\langle i|j\rangle$. We show that even in this simplified market it is #P-hard for the market maker to provide the instantaneous security prices, evaluate the cost function, or calculate payments for transactions, which implies that the running an LMSR market for the more general case of subset betting is also #P-hard.

Traders can trade securities $\langle i|j\rangle$ for all $i \in \mathcal{N}_n$ and $j \in \mathcal{N}_n$ with the market maker. Let $q_{i,j}$ be the total number of outstanding shares for security $\langle i|j\rangle$ in the market. Let $Q = (q_{i,j})_{i\in\mathcal{N}_n, j\in\mathcal{N}_n}$ denote the outstanding shares for all securities. The market maker keeps track of Q at all times. From Equation 4, the instantaneous price of security $\langle i|j\rangle$ is

$$p_{i,j}(Q) = \frac{\sum_{\sigma\in\Omega:\sigma(i)=j}\prod_{k=1}^{n} \mathrm{e}^{q_{k,\sigma(k)}/b}}{\sum_{\tau\in\Omega}\prod_{k=1}^{n} \mathrm{e}^{q_{k,\tau(k)}/b}} \ , \quad (6)$$

and from Equation 3, the cost function for subset betting is

$$C(Q) = b \log \sum_{\sigma \in \Omega} \prod_{k=1}^{n} \mathrm{e}^{q_{k,\sigma(k)}/b} \ . \tag{7}$$

We will show that computing instantaneous prices, the cost function, and/or payments of transactions for a subset betting market is #P-hard by a reduction from the problem of computing the permanent of a (0,1)-matrix.

THEOREM 1. *It is #P-hard to compute instantaneous prices in a LMSR market for subset betting. Additionally, it is #P-hard to compute the value of the cost function.*

PROOF. We show that if we could compute the instantaneous prices or the value of the cost function for subset betting for any quantities of shares purchased, then we could compute the permanent of any $(0,1)$-matrix in polynomial time.

Let n be the number of candidates, $A = (a_{i,j})$ be any n-by-n (0,1)-matrix, and $N = n! + 1$. Note that $\prod_{i=1}^{n} a_{i,\sigma(i)}$ is either 0 or 1. From Equation 5, $\mathrm{perm}(A) \leq n!$ and hence $\mathrm{perm}(A) \mod N = \mathrm{perm}(A)$. We show how to compute $\mathrm{perm}(A) \mod N$ from prices in subset betting markets in which $q_{i,j}$ shares of $\langle i|j \rangle$ have been purchased, where $q_{i,j}$ is defined by

$$q_{i,j} = \begin{cases} b \ln N & \text{if } a_{i,j} = 0, \\ b \ln(N+1) & \text{if } a_{i,j} = 1 \end{cases} \tag{8}$$

for any $i \in \mathcal{N}_n$ and any $j \in \mathcal{N}_n$.

Let $B = (b_{i,j})$ be a n-by-n matrix containing entries of the form $b_{i,j} = \mathrm{e}^{q_{i,j}/b}$. Note that $b_{i,j} = N$ if $a_{i,j} = 0$ and $b_{i,j} = N+1$ if $a_{i,j} = 1$. Thus, $\mathrm{perm}(A) \mod N = \mathrm{perm}(B) \mod N$. Thus, from Equation 6, the price for $\langle i|j \rangle$ in the market is

$$\begin{aligned} p_{i,j}(Q) &= \frac{\sum_{\sigma \in \Omega : \sigma(i) = j} \prod_{k=1}^{n} b_{k,\sigma(k)}}{\sum_{\tau \in \Omega} \prod_{k=1}^{n} b_{k,\tau(k)}} \\ &= \frac{b_{i,j} \sum_{\sigma \in \Omega : \sigma(i) = j} \prod_{k \neq i} b_{k,\sigma(k)}}{\sum_{\tau \in \Omega} \prod_{k=1}^{n} b_{k,\tau(k)}} \\ &= \frac{b_{i,j} \cdot \mathrm{perm}(M_{i,j})}{\mathrm{perm}(B)} \end{aligned}$$

where $M_{i,j}$ is the matrix obtained from B by removing the ith row and jth column. Thus the ability to efficiently compute prices gives us the ability to efficiently compute $\mathrm{perm}(M_{i,j})/\mathrm{perm}(B)$.

It remains to show that we can use this ability to compute $\mathrm{perm}(B)$. We do so by telescoping a sequence of prices. Let B_i be the matrix B with the first i rows and columns removed. From above, we have $\mathrm{perm}(B_1)/\mathrm{perm}(B) = p_{1,1}(Q)/b_{1,1}$. Define Q_m to be the $(n-m)$-by-$(n-m)$ matrix $(q_{i,j})_{i>m, j>m}$, that is, the matrix of quantities of securities $(q_{i,j})$ with the first k rows and columns removed. In a market with only $n-m$ candidates, applying the same technique to the matrix Q_m, we can obtain $\mathrm{perm}(B_{m+1})/\mathrm{perm}(B_m)$ from market prices for $m = 1, ..., (n-2)$. Thus by computing $n-1$ prices, we can compute

$$\begin{aligned} \left(\frac{\mathrm{perm}(B_1)}{\mathrm{perm}(B)}\right)\left(\frac{\mathrm{perm}(B_2)}{\mathrm{perm}(B_1)}\right) \quad \cdots \quad \left(\frac{\mathrm{perm}(B_{n-1})}{\mathrm{perm}(B_{n-2})}\right) \\ = \quad \left(\frac{\mathrm{perm}(B_{n-1})}{\mathrm{perm}(B)}\right). \end{aligned}$$

Since B_{n-1} only has one element, we thus can compute $\mathrm{perm}(B)$ from market prices. Consequently, $\mathrm{perm}(B) \mod N$ gives $\mathrm{perm}(A)$.

Therefore, given a n-by-n $(0,1)$-matrix A, we can compute the permanent of A in polynomial time using prices in $n-1$ subset betting markets wherein an appropriate quantity of securities have been purchased.

Additionally, note that

$$C(Q) = b \log \sum_{\sigma \in \Omega} \prod_{k=1}^{n} b_{k,\sigma(k)} = b \log \mathrm{perm}(B) \ .$$

Thus if we can compute $C(Q)$, we can also compute $\mathrm{perm}(A)$.

As computing the permanent of a $(0,1)$-matrix is #P-hard, both computing market prices and computing the cost function in a subset betting market are #P-hard. □

COROLLARY 2. *Computing the payment of a transaction in a LMSR for subset betting is #P-hard.*

PROOF. Suppose the market maker starts the market with 0 share of every security. Denote Q^0 as the initial quantities of all securities. If the market maker can compute $C(\tilde{Q}) - C(Q)$ for any quantities $\tilde{Q}$ and Q, it can compute $C(Q) - C(Q^0)$ for any Q. As $C(Q^0) = b \log n!$, the market maker is able to compute $C(Q)$. According to Theorem 1, computing the payment of a transaction is #P-hard. □

4.2 Pair Betting

In contrast to subset betting, where traders bet on absolute positions for a candidate, pair betting allows traders to bet on the relative position of a candidate with respect to another. More specifically, traders buy and sell securities of the form $\langle i > j \rangle$, where i and j are candidates. The security pays off \$1 if candidate i ranks higher than candidate j (*i.e.,* $\sigma(i) < \sigma(j)$ where σ is the final ranking of candidates) and \$0 otherwise. For example, traders may bet on events of the form "horse A beats horse B", or "candidate C receives more votes than candidate D".

As for subset betting, the current state of the market is determined by the total number of outstanding shares for all securities. Let $q_{i,j}$ denote the number of outstanding shares for $\langle i > j \rangle$. Applying Equations 3 and 4 to the present context, we find that the instantaneous price of the security $\langle i, j \rangle$ is given by

$$p_{i,j}(Q) = \frac{\sum_{\sigma \in \Omega : \sigma(i) < \sigma(j)} \prod_{i',j' : \sigma(i') < \sigma(j')} \mathrm{e}^{q_{i',j'}/b}}{\sum_{\tau \in \Omega} \prod_{i',j' : \tau(i') < \tau(j')} \mathrm{e}^{q_{i',j'}/b}} \ , \tag{9}$$

and the cost function for pair betting is

$$C(Q) = b \log \sum_{\sigma \in \Omega} \prod_{i,j : \sigma(i) < \sigma(j)} \mathrm{e}^{q_{i,j}/b} \ . \tag{10}$$

We will show that computing prices, the value of the cost function, and/or payments of transactions for pair betting is #P-hard via a reduction from the problem of computing the number of linear extensions to any partial ordering.

THEOREM 3. *It is #P-hard to compute instantaneous prices in a LMSR market for pair betting. Additionally, it is #P-hard to compute the value of the cost function.*

PROOF. Let P be a partial order over $\{1, \ldots, n\}$. Recall that a linear (or total) order T is a *linear extension* of P

if whenever $x \leq y$ in P it also holds that $x \leq y$ in T. We denote by $\mathcal{N}(P)$ the number of linear extensions of P.

Recall that (i, j) is a *covering pair* of P if $i \leq j$ in P and there does not exist $\ell \neq i, j$ such that $i \leq \ell \leq j$. Let $\{(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\}$ be a set of covering pairs of P. Note that covering pairs of a partially ordered set with n elements can be easily obtained in polynomial time, and that their number is less than n^2.

We will show that we can design a sequence of trades that, given a list of covering pairs for P, provide $\mathcal{N}(P)$ through a simple function of market prices.

We consider a pair betting market over n candidates. We construct a sequence of k trading periods, and denote by $q^t_{i,j}$ and $p^t_{i,j}$ respectively the outstanding quantity of security $\langle i > j \rangle$ and its instantaneous price at the end of period t. At the beginning of the market, $q^0_{i,j} = 0$ for any i and j. At each period t, $0 < t \leq k$, $b \ln n!$ shares of security $\langle i_t > j_t \rangle$ are purchased.

Let

$$N_t(i,j) = \sum_{\sigma \in \Omega : \sigma(i) < \sigma(j)} \prod_{i', j' : \sigma(i') < \sigma(j')} \mathrm{e}^{q^t_{i',j'}/b} ,$$

and

$$D_t = \sum_{\sigma \in \Omega} \prod_{i', j' : \sigma(i') < \sigma(j')} \mathrm{e}^{q^t_{i',j'}/b} .$$

Note that according to Equation 9, $p^t_{i_t,j_t} = N_t(i_t, j_t)/D_t$.

For the first period, as only the security $\langle i_1 > j_1 \rangle$ is purchased, we get

$$D_1 = \sum_{\sigma \in \Omega : \sigma(i_1) < \sigma(j_1)} n! + \sum_{\sigma : \sigma(i_1) > \sigma(j_1)} 1 = \frac{(n!)^2 + n!}{2} .$$

We now show that D_k can be calculated inductively from D_1 using successive prices given by the market. During period t, $b \ln n!$ shares of $\langle i_t > j_t \rangle$ are purchased. Note also that the securities purchased are different at each period, so that $q^s_{i_t,j_t} = 0$ if $s < t$ and $q^s_{i_t,j_t} = b \ln n!$ if $s \geq t$. We have

$$N_t(i_t, j_t) = N_{t-1}(i_t, j_t) \mathrm{e}^{b \ln(n!)/b} = n! N_{t-1}(i_t, j_t) .$$

Hence,

$$\frac{p^t_{i_t,j_t}}{p^{t-1}_{i_t,j_t}} = \frac{N_t(i_t, j_t)/D_t}{N_{t-1}(i_t, j_t)/D_{t-1}} = \frac{n! D_{t-1}}{D_t} ,$$

and therefore,

$$D_k = (n!)^{k-1} \left(\prod_{\ell=2}^{k} \frac{p^{\ell-1}_{i_\ell,j_\ell}}{p^{\ell}_{i_\ell,j_\ell}} \right) D_1 .$$

So D_k can be computed in polynomial time in n given the prices.

Alternately, since the cost function at the end of period k can be written as $C(Q) = b \log D_k$, D_k can also be computed efficiently from the cost function in period k.

We finally show that given D_k, we can compute $\mathcal{N}(P)$ in polynomial time. Note that at the end of the k trading periods, the securities purchased correspond to the covering pairs of P, such that $\mathrm{e}^{q^k_{i,j}/b} = n!$ if (i, j) is a covering pair of P and $\mathrm{e}^{q^k_{i,j}/b} = 1$ otherwise. Consequently, for a permutation σ that satisfies the partial order P, meaning that $\sigma(i) \leq \sigma(j)$ whenever $i \leq j$ in P, we have

$$\prod_{i', j' : \sigma(i') < \sigma(j')} \mathrm{e}^{q^k_{i',j'}/b} = (n!)^k .$$

On the other hand, if a permutation σ does not satisfy P, it does not satisfy at least one covering pair, meaning that there is a covering pair of P, (i, j), such that $\sigma(i) > \sigma(j)$, so that

$$\prod_{i', j' : \sigma(i') < \sigma(j')} \mathrm{e}^{q^k_{i',j'}/b} \leq (n!)^{k-1} .$$

Since the total number of permutations is $n!$, the total sum of *all* terms in the sum D_k corresponding to permutations that do not satisfy the partial ordering P is less than or equal to $n!(n!)^{k-1} = (n!)^k$, and is strictly less than $(n!)^k$ unless the number of linear extensions is 0, while the total sum of all the terms corresponding to permutations that do satisfy P is $\mathcal{N}(P)(n!)^k$. Thus $\mathcal{N}(P) = \lfloor D_k/(n!)^k \rfloor$.

We know that computing the number of linear extensions of a partial ordering is #P-hard. Therefore, both computing the prices and computing the value of the cost function in pair betting are #P-hard. □

COROLLARY 4. *Computing the payment of a transaction in a LMSR for pair betting is #P-hard.*

The proof is nearly identical to the proof of Corollary 2.

5. LMSR FOR BOOLEAN BETTING

We now examine an alternate type of market combinatorics in which the final outcome is a conjunction of event outcomes. Formally, let $\mathcal{A}$ be event space, consisting of N individual events $A_1, \cdots, A_N$, which may or may not be mutually independent. We define the state space Ω be the set of all possible joint outcomes for the N events, so that its size is $|\Omega| = 2^N$. A Boolean betting market allows traders to bet on Boolean formulas of these events and their negations. A security $\langle \phi \rangle$ pays off \$1 if the Boolean formula ϕ is satisfied by the final outcome and \$0 otherwise. For example, a security $\langle A_1 \vee A_2 \rangle$ pays off \$1 if and only if at least one of events A_1 and A_2 occurs, while a security $\langle A_1 \wedge A_3 \wedge \neg A_5 \rangle$ pays off \$1 if and only if the events A_1 and A_3 both occur and the event A_5 does not. Following the notational conventions of Fortnow et al. [11], we use $\omega \in \phi$ to mean that the outcome ω satisfies the Boolean formula ϕ. Similarly, $\omega \notin \phi$ implies that the outcome ω does not satisfy ϕ.

In this section, we focus our attention to LMSR markets for a very simple Boolean betting language, Boolean formulas of two events. We show that even when bets are only allowed to be placed on disjunctions or conjunctions of two events, it is still #P-hard to calculate the prices, the value of the cost function, and payments of transactions in a Boolean betting market operated by a LMSR market maker.

Let $\mathcal{X}$ be the set containing all elements of $\mathcal{A}$ and their negations. In other words, each event outcome $X_i \in \mathcal{X}$ is either A_j or $\neg A_j$ for some $A_j \in \mathcal{A}$. We begin by considering the scenario in which traders may only trade securities $\langle X_i \vee X_j \rangle$ corresponding to disjunctions of any two event outcomes.

Let $q_{i,j}$ be the total number of shares purchased by all traders for the security $\langle X_i \vee X_j \rangle$, which pays off \$1 in the event of any outcome ω such that $\omega \in (X_i \vee X_j)$ and \$0 otherwise. From Equation 4, we can calculate the instantaneous price for the security $\langle X_i \vee X_j \rangle$ for any two event

outcomes $X_i, X_j \in \mathcal{X}$ as

$$p_{i,j}(Q) = \frac{\sum_{\omega\in\Omega:\omega\in(X_i\vee X_j)} \prod_{1\le i'<j'\le 2N:\omega\in(X_{i'}\vee X_{j'})} \mathrm{e}^{q_{i',j'}/b}}{\sum_{\tau\in\Omega} \prod_{1\le i'<j'\le 2N:\tau\in(X_{i'}\vee X_{j'})} \mathrm{e}^{q_{i',j'}/b}}. \quad (11)$$

Note that if $X_i = \neg X_j$, $p_{i,j}(Q)$ is always \$1 regardless of how many shares of other securities have been purchased. According to Equation 3, the cost function is

$$C(Q) = b \log \sum_{\omega\in\Omega} \prod_{1\le i<j\le 2N:\omega\in(X_i\vee X_j)} \mathrm{e}^{q_{i,j}/b} . \quad (12)$$

Theorem 5 shows that computing prices and the value of the cost function in such a market is #P-hard, via a reduction from the #2-SAT problem.[1]

THEOREM 5. *It is #P-hard to compute instantaneous prices in a LMSR market for Boolean betting when bets are restricted to disjunctions of two event outcomes. Additionally, it is #P-hard to compute the value of the cost function in this setting.*

PROOF. Suppose we are given a 2-CNF (Conjunctive Normal Form) formula

$$(X_{i_1} \vee X_{j_1}) \wedge (X_{i_2} \vee X_{j_2}) \wedge \cdots \wedge (X_{i_k} \vee X_{j_k}) \quad (13)$$

with k clauses, where each clause is a disjunction of two literals (i.e. events and their negations). Assume any redundant terms have been removed.

The structure of the proof is similar to that of the pair betting case. We consider a Boolean betting markets with N events, and show how to construct a sequence of trades that provides, through prices or the value of the cost function, the number of satisfiable assignments for the 2-CNF formula.

We create k trading periods. At period t, a quantity $b\ln(2^N)$ of the security $\langle X_{i_t} \vee X_{j_t}\rangle$ is purchased. We denote by $p^t_{i,j}$ and $q^t_{i,j}$ respectively the price and outstanding quantities of the security $\langle X_i \vee X_j\rangle$ at the end of period t. Suppose the market starts with 0 share of every security. Note that $q^s_{i_t,j_t} = 0$ if $s < t$ and $q^s_{i_t,j_t} = b\ln(2^N)$ if $s \ge t$. Let

$$N_t(i,j) = \sum_{\omega\in\Omega:\omega\in(X_i\vee X_j)} \prod_{1\le i'<j'\le 2N:\omega\in(X_{i'}\vee X_{j'})} \mathrm{e}^{q^t_{i',j'}/b} ,$$

and

$$D_t = \sum_{\omega\in\Omega} \prod_{1\le i'<j'\le 2N:\omega\in(X_{i'}\vee X_{j'})} \mathrm{e}^{q^t_{i',j'}/b} .$$

Thus, $p^t_{i,j} = N_t(i_t,j_t)/D_t$.

Since only one security $\langle X_{i_1} \vee X_{j_1}\rangle$ has been purchased in period 1, we get

$$\begin{aligned} D_1 &= \sum_{\omega\in\Omega:\omega\in(X_{i_1}\vee X_{j_1})} 2^N + \sum_{\omega\in\Omega:\omega\notin(X_{i_1}\vee X_{j_1})} 1 \\ &= 3\cdot 2^{2N-2} + 2^{N-2}. \end{aligned}$$

We then show that D_k can be calculated inductively from D_1. As the only security purchased in period t is $(X_{i_t} \vee X_{j_t})$ in quantity $b\ln(2^N)$, we obtain

$$N_t(i_t,j_t) = N_{t-1}(i_t,j_t)\mathrm{e}^{b\ln(2^N)/b} = N_{t-1}(i_t,j_t)2^N .$$

[1]This can also be proved via a reduction from counting linear extensions using a similar technique to the proof of Theorem 3, but the reduction to #2-SAT is more natural.

Therefore,

$$\frac{p^t_{i_t,j_t}}{p^{t-1}_{i_t,j_t}} = \frac{N_t(i_t,j_t)/D_t}{N_{t-1}(i_t,j_t)/D_{t-1}} = \frac{2^N D_{t-1}}{D_t},$$

and we get

$$D_k = (2^N)^{k-1}\left(\prod_{\ell=2}^{k} \frac{p^{\ell-1}_{i_\ell,j_\ell}}{p^{\ell}_{i_\ell,j_\ell}}\right) D_1 .$$

In addition, since the cost function at the end of period k can be expressed as

$$C(Q) = b\log D_k ,$$

D_k can also be computed efficiently from the cost function in period k.

We now show that we can deduce from D_k the number of satisfiable assignments for the 2-CNF formula (Equation 13). Indeed, each term in the sum

$$\sum_{\omega\in\Omega} \prod_{1\le i'<j'\le 2N:\omega\in(X_{i'}\vee X_{j'})} \mathrm{e}^{q^k_{i',j'}/b}$$

that corresponds to an outcome ω that satisfies the formula is exactly 2^{kN}, as exactly k terms in the product are 2^N and the rest is 1. On the contrary, each term in the sum that corresponds to an outcome ω that does *not* satisfy the 2-CNF formula will be at most $2^{(k-1)N}$ since at most $k-1$ terms in the product will be 2^N and the rest will be 1. Since the total number of outcomes is 2^N, the total sum of *all* terms corresponding to outcomes that do not satisfy Equation 13 is less than or equal to $2^N(2^{(k-1)N}) = 2^{kN}$, and is strictly less than 2^{kN} unless the number of satisfying assignments is 0. Thus the number of satisfying assignments is $\lfloor D_k/2^{kN}\rfloor$.

We know that computing the number of satisfiable assignments of a 2-CNF formula is #P-hard. We have shown how to compute it in polynomial time using prices or the value of the cost function in a Boolean betting market of N events. Therefore, both computing prices and computing the value of the cost function in a Boolean betting market is #P-hard. □

COROLLARY 6. *Computing the payment of a transaction in a LMSR for Boolean betting is #P-hard when traders can only bet on disjunctions of two events.*

The proof is nearly identical to the proof of Corollary 2.

If we impose that participants in a Boolean betting market may only trade securities corresponding to conjunctions of any two event outcomes, $\langle A_i \wedge A_j\rangle$, the following Corollary gives the corresponding complexity results.

COROLLARY 7. *It is #P-hard to compute instantaneous prices in a LMSR market for Boolean betting when bets are restricted to conjunctions of two event outcomes. Additionally, it is #P-hard to compute the value of the cost function in this setting, and #P-hard to compute the payment for a transaction.*

PROOF. Buying q shares of security $\langle A_i \wedge A_j\rangle$ is equivalent to selling q shares of $\langle \neg A_i \vee \neg A_j\rangle$. Thus if we can operate a Boolean betting market for securities of the type $\langle A_i \wedge A_j\rangle$ in polynomial time, we can also operate a Boolean betting market for securities of the type $\langle A_i \vee A_j\rangle$ in polynomial time. The result then follows from Theorem 5 and Corollary 6. □

6. AN APPROXIMATION ALGORITHM FOR SUBSET BETTING

There is an interesting relationship between logarithmic market scoring rule market makers and a common class of algorithms for online learning in an experts setting. In this section, we elaborate on this connection, and show how results from the online learning community can be used to prove new results about an approximation algorithm for subset betting.

6.1 The Experts Setting

We begin by describing the standard model of online learning with expert advice [19, 12, 27]. In this model, at each time $t \in \{1, \cdots, T\}$, each expert $i \in \{1, \cdots, n\}$ receives a *loss* $\ell_{i,t} \in [0, 1]$. The *cumulative loss* of expert i at time T is $\mathcal{L}_{i,T} = \sum_{t=1}^{T} \ell_{i,t}$. No statistical assumptions are made about these losses, and in general, algorithms are expected to perform well even if the sequence of losses is chosen by an adversary.

An algorithm $\mathcal{A}$ maintains a current weight $w_{i,t}$ for each expert i, where $\sum_{i=1}^{n} w_{i,t} = 1$. These weights can be viewed as distributions over the experts. The algorithm then receives its own instantaneous loss $\ell_{\mathcal{A},t} = \sum_{i=1}^{n} w_{i,t}\ell_{i,t}$, which may be interpreted as the expected loss of the algorithm when choosing an expert according to the current distribution. The cumulative loss of $\mathcal{A}$ up to time T is then defined in the natural way as $\mathcal{L}_{\mathcal{A},T} = \sum_{t=1}^{T} \ell_{\mathcal{A},t} = \sum_{t=1}^{T} \sum_{i=1}^{n} w_{i,t}\ell_{i,t}$. A common goal in such online learning settings is to minimize an algorithm's *regret*. Here the regret is defined as the difference between the cumulative loss of the algorithm and the cumulative loss of an algorithm that would have "chosen" the best expert in hindsight by setting his weight to 1 throughout all the periods. Formally, the regret is given by $\mathcal{L}_{\mathcal{A},T} - \min_{i\in\{1,\cdots,n\}} \mathcal{L}_{i,T}$.

Many algorithms that have been analyzed in the online experts setting are based on exponential weight updates. These exponential updates allow the algorithm to quickly transfer weight to an expert that is outperforming the others. For example, in the Weighted Majority algorithm of Littlestone and Warmuth [19], the weight on each expert i is defined as

$$w_{i,t} = \frac{w_{i,t-1}\mathrm{e}^{-\eta\ell_{i,t}}}{\sum_{j=1}^{n} w_{j,t-1}\mathrm{e}^{-\eta\ell_{j,t}}} = \frac{\mathrm{e}^{-\eta\mathcal{L}_{i,t}}}{\sum_{j=1}^{n} \mathrm{e}^{-\eta\mathcal{L}_{j,t}}} \ , \qquad (14)$$

where η is the *learning rate*, a small positive parameter that controls the magnitude of the updates. The following theorem gives a bound on the regret of Weighted Majority. For a proof of this result and a nice overview of learning with expert advice, see, for example, Cesa-Bianchi and Lugosi [3].

THEOREM 8. *Let $\mathcal{A}$ be the Weighted Majority algorithm with parameter η. After a sequence of T trials,*

$$\mathcal{L}_{\mathcal{A},T} - \min_{i\in\{1,\cdots,n\}} \mathcal{L}_{i,T} \leq \eta T + \frac{\ln(n)}{\eta} \ .$$

6.2 Relationship to LMSR Markets

There is a manifest similarity between the expert weights utilized by Weighted Majority and the prices in an LMSR market; simply compare the form of Equation 14 with the form of Equation 2. One might ask if the results from the experts setting can be applied to the analysis of prediction markets. Our answer is *yes*. For example, it is possible to use Theorem 8 to rediscover the well-known bound of $b\ln(n)$ for the loss of an LMSR market maker with n outcomes.

Let ϵ be a limit on the number of shares that a trader may purchase or sell at each time step; in other words, if a trader would like to purchase or sell q shares, this purchase must be broken down into $\lceil q/\epsilon \rceil$ separate purchases of ϵ or less shares. Note that the total number of time steps T needed to execute such a sequence of purchases and sales is proportional to $1/\epsilon$.

We will construct a sequence of loss functions in a setting with n experts to induce a sequence of weight matrices that correspond to the price matrices of the LMSR market. At each time step t, let $p_{i,t} \in [0, 1]$ be the instantaneous price of security i at the end of period t, and let $q_{i,t} \in [-\epsilon, \epsilon]$ be the number of shares of security i purchased during period t. Let $Q_{i,t}$ be the total number of shares of security i that have been purchased up to time t. Define the instantaneous loss of each expert as $\ell_{i,t} = (2\epsilon - q_{i,t})/(\eta b)$. First notice that this loss is always in $[0, 1]$ as long as $\eta \geq 2\epsilon/b$. From Equations 2 and 14, at each time t,

$$\begin{aligned} p_{i,t} &= \frac{\mathrm{e}^{Q_{i,t}/b}}{\sum_{j=1}^{n} \mathrm{e}^{Q_{j,t}/b}} = \frac{\mathrm{e}^{2\epsilon t/b - \eta\mathcal{L}_{i,t}}}{\sum_{j=1}^{n} \mathrm{e}^{2\epsilon t/b - \eta\mathcal{L}_{j,t}}} \\ &= \frac{\mathrm{e}^{-\eta\mathcal{L}_{i,t}}}{\sum_{j=1}^{n} \mathrm{e}^{-\eta\mathcal{L}_{j,t}}} = w_{i,t} \ . \end{aligned}$$

Applying Theorem 8, and rearranging terms, we find that

$$\max_{i\in\{1,\cdots,n\}} \sum_{t=1}^{T} q_{i,t} - \sum_{t=1}^{T}\sum_{i=1}^{n} p_{i,t}q_{i,t} \leq \eta^2 Tb + b\ln(n).$$

The first term of the left-hand side is the maximum payment that the market maker needs to make, while the second terms of the left-hand side captures the total money the market maker has received. The right hand side is clearly minimized when η is set as small as possible. Setting $\eta = 2\epsilon/b$ gives us

$$\max_{i\in\{1,\cdots,n\}} \sum_{t=1}^{T} q_{i,t} - \sum_{t=1}^{T}\sum_{i=1}^{n} p_{i,t}q_{i,t} \leq 4\epsilon^2 Tb + b\ln(n).$$

Since $T = O(1/\epsilon)$, the term $4\epsilon^2 Tb$ goes to 0 as ϵ becomes very small. Thus in the limit as ϵ approaches 0, we get the well-known result that the worst-case loss of the market maker is bounded by $b\ln(n)$.

6.3 Considering Permutations

Recently Helmbold and Warmuth [16] have shown that many results from the standard experts setting can be extended to a setting in which, instead of competing with the best expert, the goal is to compete with the best permutation over n items. Here each permutation suffers a loss at each time step, and the goal of the algorithm is to maintain a weighting over permutations such that the cumulative regret to the best permutation is small. It is infeasible to treat each permutation as an expert and run a standard algorithm since this would require updating $n!$ weights at each time step. Instead, they show that when the loss has a certain structure (in particular, when the loss of a permutation is the sum of the losses of each of the n mappings), an alternate algorithm can be used that requires tracking only n^2 weights in the form of an $n \times n$ doubly stochastic matrix.

Formally, let W^t be a doubly stochastic matrix of weights maintained by the algorithm $\mathcal{A}$ at time t. Here $W^t_{i,j}$ is the

weight corresponding to the probability associated with item i being mapped into position j. Let $L^t \in [0,1]^{n\times n}$ be the loss matrix at time t. The instantaneous loss of a permutation σ at time t is $\ell_{\sigma,t} = \sum_{i=1}^n L^t_{i,\sigma(i)}$. The instantaneous loss of $\mathcal{A}$ is $\ell_{\mathcal{A},t} = \sum_{i=1}^n \sum_{j=1}^n W^t_{i,j} L^t_{i,j}$, the matrix dot product between W^t and L^t. Notice that $\ell_{\mathcal{A},t}$ is equivalent to the expectation over permutations σ drawn according to W^t of $\ell_{\sigma,t}$. The goal of the algorithm is to minimize the cumulative regret to the best permutation, $\mathcal{L}_{\mathcal{A},T} - \min_{\sigma\in\Omega} \mathcal{L}_{\sigma,T}$ where the cumulative loss is defined as before.

Helmbold and Warmuth go on to present an algorithm called PermELearn that updates the weight matrix in two steps. First, it creates a temporary matrix W', such that for every i and j, $W'_{i,j} = W^t_{i,j} \mathrm{e}^{-\eta L^t_{i,j}}$. It then obtains $W^{t+1}_{i,j}$ by repeatedly rescaling the rows and columns of W' until the matrix is doubly stochastic. Alternately rescaling rows and columns of a matrix M in this way is known as Sinkhorn balancing [22]. Normalizing the rows of a matrix is equivalent to pre-multiplying by a diagonal matrix, while normalizing the columns is equivalent to post-multiplying by a diagonal matrix. Sinkhorn [22] shows that this procedure converges to a unique doubly stochastic matrix of the form RMC where R and C are diagonal matrices if M is a positive matrix. Although there are cases in which Sinkhorn balancing does not converge in finite time, many results show that the number of Sinkhorn iterations needed to scale a matrix so that row and column sums are $1 \pm \epsilon$ is polynomial in $1/\epsilon$ [1, 17, 18].

The following theorem [16] bounds the cumulative loss of the PermELearn in terms of the cumulative loss of the best permutation.

THEOREM 9. *(Helmbold and Warmuth [16]) Let $\mathcal{A}$ be the PermELearn algorithm with parameter η. After a sequence of T trials,*

$$\mathcal{L}_{\mathcal{A},T} \le \frac{n\ln(n) + \eta \min_{\sigma\in\Omega} \mathcal{L}_{\sigma,T}}{1-\mathrm{e}^{-\eta}} \ .$$

6.4 Approximating Subset Betting

Using the PermELearn algorithm, it is simple to approximate prices for subset betting in polynomial time. We start with a $n \times n$ price matrix P^1 in which all entries are $1/n$. As before, traders may purchase securities of the form $\langle i|\Phi\rangle$ that pay off \$1 if and only if horse or candidate i finishes in a position $j \in \Phi$, or securities of the form $\langle\Psi|j\rangle$ that pay off \$1 if and only if a horse or candidate $i \in \Psi$ finishes in position j.

As in Section 6.2, each time a trader purchases or sells q shares, the purchase or sale is broken up into $\lceil q/\epsilon \rceil$ purchases or sales of ϵ shares or less, where $\epsilon > 0$ is a small constant.[2] Thus we can treat the sequence of purchases as a sequence of T purchases of ϵ or less shares, where $T = O(1/\epsilon)$. Let $q^t_{i,j}$ be the number of shares of securities $\langle i|\Phi\rangle$ with $j \in \Phi$ or $\langle\Psi|j\rangle$ with $i \in \Psi$ purchased at time t; then $q^t_{i,j} \in [-\epsilon, \epsilon]$ for all i and j.

The price matrix is updated in two steps. First, a temporary matrix P' is created where for every i and j, $P'_{i,j} = P^t_{i,j} \mathrm{e}^{q^t_{i,j}/b}$ where $b > 0$ is a parameter playing a similar role to b in Equation 2. Next, P' is Sinkhorn balanced to the desired precision, yielding an (approximately) doubly stochastic matrix P^{t+1}.

The following lemma shows that updating the price matrix in this way results in a price matrix that is equivalent to the weight matrix of PermELearn with particular loss functions.

LEMMA 10. *The sequence of price matrices obtained by the approximation algorithm for subset betting on a sequence of purchases $q^t \in [-\epsilon,\epsilon]^{n\times n}$ is equivalent to the sequence of weight matrices obtained by running PermELearn(η) on a sequence of losses L^t with*

$$L^t_{i,j} = \frac{2\epsilon - q^t_{i,j}}{\eta b}$$

for all i and j, for any $\eta \ge 2\epsilon/b$.

PROOF. First note that for any $\eta \ge 2\epsilon/b$, $L^t_{i,j} \in [0,1]$ for all t, i, and j, so the loss matrix is valid for PermELearn. P^1 and W^1 both contain all entries of $1/n$. Assume that $P^t = W^t$. When updating weights for time $t+1$, for all i and j,

$$\begin{aligned} P'_{i,j} &= P^t_{i,j}\mathrm{e}^{q^t_{i,j}/b} = W^t_{i,j}\mathrm{e}^{q^t_{i,j}/b} \\ &= \mathrm{e}^{2\epsilon/b} W^t_{i,j} \mathrm{e}^{-2\epsilon/b + q^t_{i,j}/b} \\ &= \mathrm{e}^{2\epsilon/b} W^t_{i,j} \mathrm{e}^{-\eta L^t_{i,j}} = \mathrm{e}^{2\epsilon/b} W'_{i,j} \ . \end{aligned}$$

Since the matrix W' is a constant multiple of P', the Sinkhorn balancing step will produce the same matrices. □

Using this lemma, we can show that the difference between the amount of money that the market maker must distribute to traders in the worst case (i.e. when the true outcome is the outcome that pays off the most) and the amount of money collected by the market is bounded. We will see in the corollary below that as ϵ approaches 0, the worst case loss of the market maker approaches $bn\ln(n)$, regardless of the number of shares purchased. Unfortunately, if $\epsilon > 0$, this bound can grow arbitrarily large.

THEOREM 11. *For any sequence of valid subset betting purchases q^t where $q^t_{i,j} \in [-\epsilon,\epsilon]$ for all t, i, and j, let $P^1, \cdots, P^T$ be the price matrices obtained by running the subset betting approximation algorithm. Then*

$$\begin{aligned} &\max_{\sigma\in S_n} \sum_{t=1}^T \sum_{i=1}^n q^t_{i,\sigma(i)} - \sum_{t=1}^T \sum_{i=1}^n \sum_{j=1}^n P^t_{i,j} q^t_{i,j} \\ &\quad\le \frac{2\epsilon/b}{1-\mathrm{e}^{-2\epsilon/b}} n\ln(n) + \left(\frac{2\epsilon/b}{1-\mathrm{e}^{-2\epsilon/b}} - 1\right) 2\epsilon n T \ . \end{aligned}$$

PROOF. By Theorem 9 and Lemma 10, after rearranging terms, we have that for any $\eta \ge 2\epsilon/b$,

$$\begin{aligned} &2\epsilon n T - \sum_{t=1}^T \sum_{i=1}^n \sum_{j=1}^n P^t_{i,j} \\ &\quad\le \left(\frac{\eta}{1-\mathrm{e}^{-\eta}}\right) \left(bn\ln n + 2\epsilon n T - \max_{\sigma\in S_n} \sum_{t=1}^T \sum_{i=1}^n q^t_{i,\sigma(i)} \right) \ . \end{aligned}$$

[2] We remark that dividing purchases in this way has the negative effect of creating a polynomial time dependence on the quantity of shares purchased. However, this is not a problem if the quantity of shares bought or sold in each trade is bounded to start, which is a reasonable assumption. The additional time required is then linear only in $1/\epsilon$.

Thus we have

$$\frac{\eta}{1-\mathrm{e}^{-\eta}} \max_{\sigma \in S_n} \sum_{t=1}^{T} \sum_{i=1}^{n} q_{i,\sigma(i)}^t - \sum_{t=1}^{T} \sum_{i=1}^{n} \sum_{j=1}^{n} P_{i,j}^t q_{i,j}^t$$
$$\leq \quad \frac{\eta}{1-\mathrm{e}^{-\eta}} bn \ln(n) + \left(\frac{\eta}{1-\mathrm{e}^{-\eta}} - 1 \right) 2\epsilon n T \ .$$

Notice that $\eta/(1-\mathrm{e}^{-\eta})$ is an increasing function in η, and goes to 1 in the limit as η goes to 0. Thus the right hand side of this equation decreases as η decreases to 0. Setting $\eta = 2\epsilon/b$ yields the result. □

Notice that the number of steps T scales inversely with ϵ since each lump purchase of q shares must be broken into $\lceil q/\epsilon \rceil$ individual purchases. Thus in the limit as ϵ approaches 0, the loss of the market maker is bounded by $bn \ln(n)$.

COROLLARY 12. *For any sequence of valid subset betting purchases broken into T $(= O(1/\epsilon))$ small purchases such that $q_{i,j}^t \in [-\epsilon, \epsilon]$ for all t, i, and j, let $P^1, \cdots, P^T$ be the price matrices obtained by running the Subset Betting Approximation Algorithm. In the limit as ϵ approaches 0,*

$$\max_{\sigma \in S_n} \sum_{t=1}^{T} \sum_{i=1}^{n} q_{i,\sigma(i)}^t - \sum_{t=1}^{T} \sum_{i=1}^{n} \sum_{j=1}^{n} P_{i,j}^t q_{i,j}^t \leq bn \ln(n) \ .$$

This bound is comparable to worst-case loss bounds achieved using alternate methods for operating LMSRs on permutations. A single LMSR operated on the entire outcome space has a guaranteed worst-case loss of $b \ln(n!)$, but is, of course, intractable to operate. A set of n LMSRs operated as n separate markets, one for each position, would also have a total worst-case loss $bn \ln(n)$, but could not guarantee consistent prices. In the limit, our approximation algorithm achieves the same worst-case loss guarantee as if we were operating n separate markets, but prices remain consistent at all times.

7. CONCLUSIONS

We investigate the computational complexity for logarithmic market scoring rule (LMSR) market makers to operate combinatorial betting markets. We examine two specific market combinatorics, permutations and Boolean combinations. In a permutation betting market, the state space is the set of rankings of n competing candidates and is of size of $n!$. In a Boolean betting market, the state space is the set of Boolean combinations of N event outcomes and is of size 2^N.

Since allowing participants to bet on individual states is both intractable and unnatural, we allow participants to trade securities that correspond to simple properties of the final state. For permutation betting, we consider subset betting, which allows traders to bet on a set of positions that a candidate will stand at or a set of traders who will stand at a particular position, and pair betting, which allows traders to wager on the relative ranking of two candidates. For Boolean betting, we consider the situation where traders bet on conjunctions or disjunctions of two events. In all cases, we prove that it is #P-hard to compute the instantaneous prices as well as payments of transactions in a LMSR market. Our results on subset betting in LMSR contrast with those of Chen et al. [4] who show that subset betting is tractable when the market is operated by a central auctioneer who performs riskless order matching. Our results on Boolean betting contrast with those of Chen, Goel, and Pennock [6] who consider betting on single-elimination tournaments—a special form of Boolean combinations—and show that such a market with LMSR may be operated in polynomial time. This raises interesting questions on the connection between the complexity of auctioneers and the complexity of market makers, and on the complexity of other combinatorial betting scenarios.

We also show that there is an interesting relationship between LMSR market makers and a common class of expert learning algorithms. This allows us to design an approximation algorithm for pricing subset betting in an LMSR. We prove that in the limit the worst-case loss of an LMSR market maker that uses our algorithm remains bounded. In the future we plan to further investigate the connection between online learning in expert settings and information markets with automated market makers.

8. ACKNOWLEDGMENTS

The authors are grateful to Sampath Kannan for useful discussions and advice on how to approach the pair betting problem, to Manfred Warmuth for sharing extended details of his work on learning permutations, and to Sharad Goel for insightful discussions.

9. REFERENCES

[1] H. Balakrishnan, I. Hwang, and C. Tomlin. Polynomial approximation algorithms for belief matrix maintenance in identity management. In *43rd IEEE Conference on Decision and Control*, pages 4874–4879, 2004.

[2] G. Brightwell and P. Winkler. Counting linear extensions is #P-complete. In *ACM Symposium on Theory of Computing*, 1991.

[3] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[4] Y. Chen, L. Fortnow, E. V. Nikolova, and D. M. Pennock. Betting on permutations. In *ACM Conference on Electronic Commerce*, 2007.

[5] Y. Chen, L. Fortnow, E. V. Nikolova, and D. M. Pennock. Combinatorial betting. *ACM SIGecom Exchanges*, 7(1):865–877, 2007.

[6] Y. Chen, S. Goel, and D. M. Pennock. Pricing combinatorial markets for tournaments. In *ACM Symposium on Theory of Computing*, 2008. To appear.

[7] Y. Chen, D. M. Reeves, D. M. Pennock, R. D. Hanson, L. Fortnow, and R. Gonen. Bluffing and strategic reticence in prediction markets. In *Workshop on Internet and Network Economics*, 2007.

[8] Yiling Chen and David M. Pennock. A utility framework for bounded-loss market makers. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 49–56, 2007.

[9] R. Forsythe, F. Nelson, G. Neumann, and J. Wright. Anatomy of an experimental political stock market. *The American Economic Review*, 82(5):1142–1161, 1992.

[10] R. Forsythe, T. Rietz, and T. Ross. Wishes, expectations, and actions: A survey on price formation in election stock markets. *Journal of Economic Behavior and Organization*, 39:83–110, 1999.

[11] L. Fortnow, J. Kilian, D. M. Pennock, and M. Wellman. Betting boolean-style: A framework for trading securities based on logical formulas. *Decision Support Systems*, 39(1):87–104, 2004.

[12] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[13] R. D. Hanson. Combinatorial information market design. *Information Systems Frontiers*, (1):105–119, 2003.

[14] R. D. Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 2007.

[15] R. D. Hanson, J. Ledyard, and T. Ishikida. An experimental test of combinatorial information markets. *Journal of Economic Behavior and Organization*, 2008. To appear.

[16] D. Helmbold and M. Warmuth. Learning permutations with exponential weights. In *20th Annual Conference on Learning Theory*, pages 469–483, 2007.

[17] Bahman Kalantari and Leonid Khachiyan. On the complexity of nonnegative-matrix scaling. *Linear Algebra and its applications*, (240):87–103, 1996.

[18] Nathan Linial, Alex Samorodnitsky, and Avi Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Combinatorica*, 20(4):545–568, 2000.

[19] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[20] R. Nelson and D. Bessler. Subjective probabilities and scoring rules: Experimental evidence. *American Journal of Agricultural Economics*, 71(2):363–369, 1989.

[21] K. Oliven and T. Rietz. Suckers are born, but markets are made: Individual rationality, arbitrage, and market efficiency on an electronic futures market. *Management Science*, 50(3):336–351, 2004.

[22] R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

[23] R. Thaler and W. Ziemba. Anomalies: Parimutuel betting markets: Racetracks and lotteries. *Journal of Economic Perspectives*, 2(2):161–174, 1988.

[24] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[25] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[26] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8(3):410–421, 1979.

[27] V. Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56:153–173, 1998.