

# Generalized Weighted Model Counting: An Efficient Monte-Carlo Meta-Algorithm

(Working paper)

Lirong Xia

SEAS, Harvard University, USA, lxia@seas.harvard.edu

**Abstract.** In this paper, we focus on computing the prices of securities represented by logical formulas in combinatorial prediction markets when the price function is represented by a Bayesian network. This problem turns out to be a natural extension of the weighted model counting (WMC) problem [15], which we call *generalized weighted model counting (GWMC)* problem. In GWMC, we are given a logical formula  $F$  and a polynomial-time computable weight function. We are asked to compute the total weight of the valuations that satisfy  $F$ . Based on importance sampling, we propose a Monte-Carlo meta-algorithm that has a good theoretical guarantee for formulas in disjunctive normal form (DNF). The meta-algorithm queries an oracle algorithm that computes marginal probabilities in Bayesian networks, and has the following theoretical guarantee. When the weight function can be approximately represented by a Bayesian network for which the oracle algorithm runs in polynomial time, our meta-algorithm becomes a *fully polynomial-time randomized approximation scheme (FPRAS)*.

## 1 Introduction

Prediction markets are a type of financial markets that aggregates agents' probabilistic information about some random event. The *Iowa Electronic Market* and *Intrade* are two examples of real-world prediction markets with a long history of tested results [1, 2]. See [6] for a recent survey.

In many real-life situations, the number of outcomes of the random event is exponentially large and has a combinatorial structure. Such situations are called *combinatorial prediction markets* [4, 5, 8, 10, 11]. For example, in the NCAA men's basketball tournament, there are 64 teams and therefore 63 matches in total to predict. Each match can be seen as a binary variable. It follows that the prediction market for this tournament has  $2^{63} \approx 9.2 \times 10^{18}$  outcomes. In such situations, even though usually the price of a security corresponding to a single outcome is computationally easy, it is not surprising to see that computing and updating the prices by directly using the cost function becomes computationally intractable. In fact, pricing LMSR-based combinatorial prediction markets is #P-hard [4].

To overcome this computational intractability, one promising idea is to use a *Bayesian network* to represent the prices of the securities corresponding to disjoint and exhaustive outcomes. This idea was first explored by [5] for a class of LMSR-based combinatorial prediction markets for tournaments. They modeled the market price distribution by a Bayesian network whose graph is a balanced binary tree, and identified two

types of *structure-preserving* securities: after any shares of any such securities are sold, the updated market price distribution can still be represented by a Bayesian network with the same structure. A followup work obtained a full characterization for structure-preserving securities for any network structure [14].

However, the two papers mentioned above focused on price-updating after trading some shares of a security rather than the #P-hard problem of computing the price of a given security when the market prices are represented by a Bayesian network. In this paper, we tackle this problem by designing a sample-based Monte-Carlo algorithm. Mathematically, the problem corresponds to computing the total weight (prices) of valuations (outcomes) that satisfy a given logical formula  $F$ , which is a natural generalization of the *model counting problem* (a.k.a. #SAT), one of the most important problems in artificial intelligence. Our main technical contribution is a Monte-Carlo meta-algorithm (Algorithm 1) based on importance sampling that uses two sub-procedures. One procedure computes the marginal probabilities in Bayesian networks (we call this the oracle algorithm), and the other computes the weight of a given valuation.<sup>1</sup> Our algorithm has the following theoretical guarantee: Suppose the weights can be approximately represented by a Bayesian network  $\pi^*$  (we will formally define this in Definition 2). For any DNF formula  $F$  and any error rate  $\epsilon$ , Algorithm 1 is an unbiased estimator for the total weight of the outcomes that satisfies  $F$ , and with probability larger than  $3/4$  the multiplicative error is no more than  $\epsilon$ . Moreover, the algorithm runs in polynomial time plus polynomial calls to the oracle algorithm, where the running time (respectively the number of calls) is polynomial in the input size and  $1/\epsilon$ .

Of course the performance of the meta-algorithm depends on both  $\pi^*$  and the oracle algorithm that computes the marginal probabilities. One important corollary is that if there exists a polynomial time algorithm that computes marginal probabilities in  $\pi^*$ , then Algorithm 1 is a *fully polynomial-time randomized approximation scheme* (FPRAS). For example, when  $\pi^*$  can be represented by a Bayesian network whose structure is a polytree,<sup>2</sup> then we can use the celebrated *belief-propagation algorithm* [13] to obtain an FPRAS. Therefore, we automatically obtain another FPRAS for pricing LMSR-based combinatorial prediction markets for tournaments [18].

**Related Work.** Our work is closest to the following two lines of research. Conceptually, it is closely related to and is a natural generalization of *weighted model counting problems* (WMC) [15]. We will see that WMC is a very special case of GWMC where the weight function can be represented by a Bayesian network without edges (Example 1). Therefore, our main algorithm is an FPRAS for WMC, and we are not aware of a previous FPRAS for WMC. On the technical level, the main application of WMC is to solve Bayesian inference by reducing a Bayesian inference problem to a WMC problem. Our meta-algorithm, on the other hand, aims at exploring the power of Bayesian inference algorithms to compute GWMC for various applications, including pricing combinatorial prediction markets.

Technically, our work extends the idea of the FPRAS algorithm for model counting by Karp, Luby, and Madras [12] (KLM for short), and the FPRAS algorithm for pricing combinatorial prediction markets for tournaments in our previous work [18].

<sup>1</sup> The second procedure is usually given as part of the input.

<sup>2</sup> A polytree is a directed graph that does not contain undirected cycles.

However, the KLM algorithm only deals with (unweighted) model counting problems. While the framework of our meta-algorithm is similar to Algorithm 1 in [18], our meta-algorithm uses a Bayesian inference algorithm as an oracle, and the algorithm in [18] heavily depends on the assumption that  $\pi^*$  can be represented by a Bayesian network whose structure is a balanced binary tree.

## 2 Preliminaries

### 2.1 Combinatorial Prediction Markets

In *combinatorial prediction markets* [4], the set of outcomes  $\Omega$  has a combinatorial structure. That is, each outcome is characterized by the values of a set of variables  $\mathcal{V} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ , where for each  $k \leq n$ ,  $\mathbf{x}_k$  takes a value in a domain  $\Omega_k = \{0, 1\}$ . It follows that  $\Omega = \Omega_1 \times \dots \times \Omega_n$ . In this paper, a security is represented by a logical formula  $F$  over  $\mathcal{V}$  in *conjunctive normal form (CNF)*. That is,  $F = C_1 \wedge \dots \wedge C_K$ , where for any  $j \leq K$ ,  $C_j = l_1^j \wedge \dots \wedge l_{s_j}^j$ , and  $l_i^j$  is either  $\mathbf{x}$  or  $\neg\mathbf{x}$  for some variable  $\mathbf{x}$ .  $C_j$  is called a *clause* and  $l_i^j$  is called a *literal*. We assume that no clause contains both  $\mathbf{x}$  and  $\neg\mathbf{x}$  for any variable  $\mathbf{x}$ . If  $F$  is satisfied under the eventual true outcome (which is a valuation over  $\mathcal{V}$ ), then the market maker should pay the agent \$1 for each share of  $F$  the agent holds; otherwise the agent receives nothing for holding  $F$ .

At any point, we let  $p(\cdot)$  denote the price function. By definition, the instantaneous price of  $F$  is the sum of the prices of the securities that correspond to the valuations under which  $F$  is satisfied. That is,  $p(F) = \sum_{\mathbf{x}: F(\mathbf{x})=1} p(\mathbf{x})$ .

### 2.2 Importance Sampling

Importance sampling is a general technique for Monte-Carlo methods that reduces the variance of estimation, which in turn improves the convergence rate. Suppose we want to evaluate the expectation of a function  $f : \{1, \dots, N\} \rightarrow \mathbb{R}$  when the variable is chosen from a probability distribution  $\pi$  over  $\{1, \dots, N\}$ . That is, we want to evaluate the expectation of  $f$  w.r.t.  $\pi$ , denoted by  $E[f; \pi]$ . The most straightforward Monte-Carlo method is to generate  $Z$  samples  $X_1, \dots, X_Z$  i.i.d. according to  $\pi$ , and use  $\frac{1}{Z} \sum_{i=1}^Z f(X_i)$  as an unbiased estimator for  $E[f; \pi]$ . The convergence rate is guaranteed by the following lemma, which follows directly from Chebyshev's inequality.

**Lemma 1 (Follows from Chebyshev's inequality).** *Let  $H_1, \dots, H_Z$  be i.i.d. random variables with  $\mu = E[H_i]$  and variance  $\sigma^2$ . If  $Z \geq 4\sigma^2/(\epsilon^2\mu^2)$ , then,*

$$\Pr(|\frac{1}{Z} \sum_{i=1}^Z H_i - \mu| < \epsilon\mu) \geq 3/4$$

This lemma illustrates that for a fixed  $\epsilon$ , the smaller  $\sigma^2/\mu^2$ , the faster this sampling method converges. Importance sampling reduces the variance by generating the outcomes that have higher  $f$  values more often. Suppose we have another distribution  $\pi^*$  such that for every outcome  $i$ ,  $[\pi^*(i) = 0] \implies [f(i)\pi(i) = 0]$ . We can then use  $\pi^*$  to provide an unbiased estimator for  $E[f; \pi]$  as follows. Let  $H$  denote the random variable that takes  $\frac{f(i)\pi(i)}{\pi^*(i)}$  with probability  $\pi^*(i)$ . We generate  $Z$  i.i.d. samples of  $H$ , denoted by  $H_1, \dots, H_Z$ , and use  $\frac{1}{Z} \sum_{i=1}^Z H_i$  as an estimator for  $E[f; \pi]$ . It is easy to check

that this estimator is unbiased, and  $\text{Var}(H)/E[H]^2$  might be significantly smaller than  $\text{Var}(f)/E[f; \pi]^2$ .

A good  $\pi^*$  can greatly reduce the ratio of variance over square expectation, therefore in turn improving the performance of the Monte-Carlo method. The best scenario is that for any outcome  $i$ ,  $\pi^*(i)$  is proportional to  $f(i)\pi(i)$ . Then, the variance becomes 0 and we only need 1 sample to calculate the expectation. In practice, sometimes we would like to choose  $\pi^*$  that is a good approximation to  $f(i)\pi(i)$  and is much easier to handle, as we will see in our meta-algorithm.

An algorithm  $A$  is a *fully polynomial-time randomized approximation scheme (FPRAS)* for a function  $f$ , if for any input  $x$  and any error rate  $\epsilon$ , (1) the output of the algorithm  $A$  is in  $[(1 - \epsilon)f(x), (1 + \epsilon)f(x)]$  with probability at least  $3/4$ ,<sup>3</sup> (2) the runtime of  $A$  is polynomial in  $1/\epsilon$  and the size of  $x$ .

### 3 Generalized Weighted Model Counting

In this section, we formally model the computational problem of pricing in combinatorial prediction markets as a generalized weighted model counting problem.

**Definition 1** *In the generalized weighted model counting (GWMC) problem, we are given a logical formula  $F$  and a polynomial-time computable weight function  $w$  defined over all valuations of  $\mathcal{X}$ . We are asked to compute the total weight of the valuations that satisfy  $F$ , denoted by  $W(F)$ . Formally,*

$$W(F) = \sum_{\mathbf{x}} w(\mathbf{x})F(\mathbf{x}) = \sum_{\mathbf{x}:F(\mathbf{x})=1} w(\mathbf{x})$$

We note that  $w$  can be represented by a probability distribution  $\pi_w$  over  $\mathcal{X}$  and a total weight  $W$ , where  $W = \sum_{\mathbf{x}} w(\mathbf{x})$  and  $\pi_w(\mathbf{x}) = w(\mathbf{x})/W$ . Therefore, if  $\pi_w$  can be represented by a compact Bayesian network, then  $w$  is polynomial-time computable. However, inference under simple Bayesian network might still be NP-hard [7]. Our algorithm will use a Bayesian network  $\pi^*$  that approximately represents  $w$ . We next define the notion of such approximation.

**Definition 2** *For any  $c > 1$ , we say that a probability distribution  $\pi^*$  is a  $c$ -approximation to  $\pi_w$ , if for any outcome  $\mathbf{x}$ ,  $\frac{1}{c}\pi_w(\mathbf{x}) \leq \pi^*(\mathbf{x}) \leq c\pi_w(\mathbf{x})$ .*

Of course we can always set  $\pi^* = \pi_w$ , but a good  $\pi^*$  may greatly reduce computational complexity. Such an approximation  $\pi^*$  may come from expert knowledge. We next show that GWMC is an extension of weighted model counting (WMC) [3] with a very simple Bayesian network, and can be used to compute prices in combinatorial prediction markets [18].

In WMC, we are given a propositional logical formula  $F$  and a weight  $\bar{w}(\mathbf{x})$  for each valuation  $\mathbf{x}$ . We are asked to compute  $\sum_{\mathbf{x}:F(\mathbf{x})=1} \bar{w}(\mathbf{x})$ , that is, the total weight of the valuations that satisfy  $F$ . The weight function is represented compactly in the following way: Each literal contributes a multiplicative factor to the weight of a valuation. More

<sup>3</sup> By using the *median of means* method, for any  $\delta < 1$ , the success rate of an FPRAS can be increased to  $1 - \delta$ , at the cost of increasing the runtime by a multiplicative factor of  $\ln(\delta^{-1})$  (cf. Exercise 28.1 in [17]).

precisely, for each variable  $\mathbf{x}$  there are two weights  $w_{\mathbf{x}}^0$  and  $w_{\mathbf{x}}^1$ . For any valuation  $x_i$  of  $\mathbf{x}_i$ , let  $\bar{w}(x_i) = w_{\mathbf{x}}^0$  if and only if  $x_i = 0$  and let  $\bar{w}(x_i) = w_{\mathbf{x}}^1$  if and only if  $x_i = 1$ . For any valuation  $\mathbf{x}$  of all variables, let  $\bar{w}(\mathbf{x}) = \prod_i \bar{w}(x_i)$ .

**Example 1** *WMC is an special case of GWMC. To see this, we show how to model  $\bar{w}$  as a simple Bayesian network. Suppose in the BN there are no edges, and for any variable  $\mathbf{x}$ , let  $\pi_w(\mathbf{x} = 1) = w_{\mathbf{x}}^1$  and  $\pi_w(\mathbf{x} = 0) = w_{\mathbf{x}}^0$ . Let  $\pi_w$  denote the probability distribution and  $W = \sum_{\mathbf{x}} \bar{w}(\mathbf{x})$ . It follows that for any valuation  $\mathbf{x}$ ,  $\bar{w}(\mathbf{x}) = W \pi_w(\mathbf{x})$ .*

**Example 2** *In combinatorial prediction market, for each valuation  $\mathbf{x}$ , the price for its corresponding security, denoted by  $p(\mathbf{x})$  and usually can be computed in polynomial time. When a security is represented by a logical formula  $F$ , the price of which is defined to be  $p(F) = \sum_{\mathbf{x}: F(\mathbf{x})=1} p(\mathbf{x})$ . Computing  $p(F)$  is a GWMC problem, where we let  $w = p$ .*

## 4 The Meta Algorithm

The meta-algorithm we present in this section contains two procedures. (1) Procedure CompMargin is the oracle algorithm that computes the marginal probability in a Bayesian network. (2) Procedure QueryWeight returns the weight of a given valuation. The second procedure is usually given as part of the input. First, we do not specify Procedure CompMargin, and will evaluate the computational complexity of Algorithm 1 by number of calls to Procedure CompMargin. The input of our meta-algorithm consists in  $F$ ,  $w$ ,  $\epsilon$ , and  $\pi^*$  that is a  $c$ -approximation to  $\pi_w$  for some constant  $c$ .

---

**Procedure** CompMargin( $\pi^*, C_j$ )

---

**Input:**  $\pi^*$  and a conjunction of literals  $C_j$ .

**Output:**  $\pi^*(C_j)$ .

---



---

**Procedure** QueryWeight( $\cdot$ )

---

**Input:** A valuation  $\mathbf{x}$ .

**Output:**  $w(\mathbf{x})$ .

---

The meta-algorithm is similar to the KLM algorithm and the Algorithm 1 in [18]. It contains the following three steps in each iteration.

(1) Compute  $G = \sum_{j'} \pi^*(C_{j'})$  by calling Procedure CompMargin for  $k$  times. Choose a clause  $C_j$  with probability  $\pi^*(C_j)/G$ .

(2) Choose a valuation  $\mathbf{x}$  with probability  $\pi^*(\mathbf{x}|C_j)$  from all valuations that satisfy  $C_j$  by calling Procedure GenerateValuation, which calls Procedure CompMargin for no more than  $2m$  times.

(3) Finally, compute the number of clauses  $\mathbf{x}$  satisfies, denoted by  $n(\mathbf{x})$ , and add  $\frac{w(\mathbf{x})G}{\pi^*(\mathbf{x})n(\mathbf{x})}$  to a counter  $N$  by calling Procedure QueryWeight once.

Procedure QueryWeight generates values of variables in  $\mathcal{X}$  sequentially in a way similarly to the Monte-Carlo sampling algorithm in [9]. More precisely, given  $j$ , w.l.o.g. let  $C_j = \mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \dots \wedge \mathbf{x}_{m'}$ . Let  $x_1 = \dots = x_{m'} = 1$ . Suppose the values of  $\mathbf{x}_1, \dots, \mathbf{x}_t$  has been determined for some  $m' \leq t < m$ , such that for every  $i \leq t$ ,

$\mathbf{x}_i = x_i$ . Let  $p_1 = \pi^*(\mathbf{x}_1 = x_1, \dots, \mathbf{x}_t = x_t, \mathbf{x}_{t+1} = 1)$  and  $p_0 = \pi^*(\mathbf{x}_1 = x_1, \dots, \mathbf{x}_t = x_t, \mathbf{x}_{t+1} = 0)$ .  $p_1$  and  $p_2$  can be computed by Procedure CompMargin. We let  $\mathbf{x}_{t+1} = \begin{cases} 1 & \text{with probability } p_1/(p_1 + p_0) \\ 0 & \text{with probability } p_0/(p_1 + p_0) \end{cases}$  and let  $t \leftarrow t + 1$ .

---

**Algorithm 1: GWMC**


---

**Input:**  $w, \pi^*, \epsilon$ , and a DNF formula  $F = C_1 \vee \dots \vee C_k$ .  
**Output:** An estimation for  $W(F)$ .

- 1 Compute  $G = \sum_{j' \leq k} \text{CompMargin}(C_{j'})$ .
- 2 **for**  $i = 1$  to  $Z = 4c^4 k^4 / \epsilon^2$  **do**
- 3     Choose an index  $j$  with probability  $\frac{\text{CompMargin}(C_j)}{G}$ .
- 4     Call GenerateValuation( $\pi^*, C_j$ ) to choose a valuation  $\mathbf{x}$  with probability  $\pi^*(\mathbf{x}|C_j)$  from all valuations that satisfy  $C_j$ .
- 5     Compute  $n(\mathbf{x}) = |\{j' : C_{j'}(\mathbf{x}) = 1\}|$ .
- 6     Let  $N \leftarrow N + \frac{\text{QueryWeight}(\mathbf{x})G}{\pi^*(\mathbf{x})n(\mathbf{x})}$ .
- 7 **end**
- 8 **return**  $N/Z$ .

---



---

**Procedure GenerateValuation( $\pi^*, C_j$ )**


---

**Input:**  $\pi^*$  and a conjunction of literals  $C_j$ .  
**Output:** A valuation  $\mathbf{x}$  that satisfies  $C_j$  w.p.  $\pi^*(\mathbf{x}|S_j)$ .

- 1 **for** any variable  $\mathbf{x}$  whose literals appear in  $C_j$  **do**
- 2     If  $C_j$  contains  $\mathbf{x}$ , then let  $\mathbf{x} = 1$ ; otherwise let  $\mathbf{x} = 0$ .
- 3 **end**
- 4 Let  $C = C_j$ .
- 5 **while** there exists a variable  $\mathbf{x}$  such that  $C$  does not contain  $\mathbf{x}$  or  $\neg\mathbf{x}$  **do**
- 6     Let  $p_1 = \text{CompMargin}(\pi^*, C \wedge \mathbf{x})$ , let  $p_0 = \text{CompMargin}(\pi^*, C \wedge \neg\mathbf{x})$ .
- 7     Choose  $\mathbf{x} = \begin{cases} 1 & \text{w.p. } \frac{p_1}{p_1 + p_0}, \text{ and then let } C \leftarrow C \wedge \mathbf{x} \\ 0 & \text{w.p. } \frac{p_0}{p_1 + p_0}, \text{ and then let } C \leftarrow C \wedge \neg\mathbf{x} \end{cases}$
- 8 **end**
- 9 **return**  $\mathbf{x}$ .

---

**Theorem 1** *If  $\pi^*$  is a  $c$ -approximation to  $\pi_w$  for some constant  $c$ , then given any  $\epsilon > 1$ , Algorithm 1 is an unbiased estimator for  $W(F)$ , and*

$$\Pr((1 - \epsilon)W(F) < \text{Output} < (1 + \epsilon)W(F)) > \frac{3}{4}. \quad (1)$$

*The running time of Algorithm 1 is a polynomial function in  $1/\epsilon$  and the input size, plus polynomial number of calls to Procedure CompMargin.*

**Proof of Theorem 1:** We first prove that Algorithm 1 is an unbiased estimator for  $W(F)$ . Let  $X_i$  (for all  $1 \leq i \leq Z$ ) denote the random variable that corresponds

to the  $i$ th sample added to  $N$  in Step 6. Then,  $E(X_i) = \sum_{\mathbf{x}:F(\mathbf{x})=1} \frac{w(\mathbf{x})G}{\pi^*(\mathbf{x})n(\mathbf{x})} \times \frac{\pi^*(\mathbf{x})n(\mathbf{x})}{G} = \sum_{\mathbf{x}:F(\mathbf{x})=1} w(\mathbf{x}) = W(F)$ . Therefore, the output of Algorithm 1 is an unbiased estimator for  $W(F)$ .

To prove Inequality (1), we note that  $X_i = \frac{w(\mathbf{x})G}{\pi^*(\mathbf{x})n(\mathbf{x})} = W(F)G \frac{\pi_w(\mathbf{x})}{\pi^*(\mathbf{x})n(\mathbf{x})}$ . Because  $\pi^*$  is a  $c$ -approximation to  $\pi_w$  and  $1 \leq n(\mathbf{x}) \leq k$ , we have  $\frac{1}{ck}E(X_i) \leq X_i \leq ckE(X_i)$ , which means that  $Var(X_i)/(E(X_i))^2 \leq (ck)^4$ . Inequality (1) follows after Lemma 1.

Algorithm 1 calls Procedure CompMargin for  $k$  times in Step 1 and calls Procedure GenerateValuation for  $4c^4k^4/\epsilon^2$  times, in each of which Procedure CompMargin is called for no more than  $2m$  times. Therefore, the total number of times Algorithm 1 calls Procedure CompMargin is polynomial in  $1/\epsilon$  and the input size.  $\square$

**Remark:** the power of Theorem 1 is that even if  $\pi_w$  cannot be represented by a Bayesian network where inference is computationally easy, it might still be possible to find a good approximation  $\pi^*$  where inference is computationally easy. Since inference for polytree-structured Bayesian networks can be done in polynomial time by the belief-propagation algorithm [13], we immediately have the following corollary.

**Corollary 1** *If  $\pi^*$  is a  $c$ -approximation to  $\pi_w$  for some constant  $c$  and  $\pi^*$  can be represented by a polytree-structured Bayesian network, then Algorithm 1 is an unbiased FPRAS for  $W(F)$  where the oracle algorithm is the belief-propagation algorithm.*

We have shown in Example 1 that WMCs are GWMCs where  $\pi_w$  can be represented by a Bayesian network without edges. Also, it is assumed in [18] that for combinatorial prediction markets for tournaments, there exists a Bayesian network whose structure is a tree and is a  $c$ -approximation to the price function. Since both structures are polytrees, it follows immediately after Corollary 1 that Algorithm 1 is an FPRAS for these two problems if we use the belief-propagation algorithm as Procedure CompMargin.

## 5 Summary and Future Work

In this paper, we propose an efficient Monte-Carlo meta-algorithm to compute GWMC for DNF formulas, motivated by applications in pricing combinatorial prediction markets. Our algorithm is an FPRAS if the weight function can be approximately represented by a polytree-structured Bayesian network. There are many directions for future research. Technically, it would be interesting to develop techniques for GWMC for CNF formulas, for example, we can study how to extend techniques developed for WMC to GWMC. Another important direction is to find more applications of GWMC and test the performance of the algorithms for real-world applications.

## 6 Acknowledgments

The author is supported by NSF under Grant #1136996 to the Computing Research Association for the CIFellows Project. The author thanks Carla Gomes, David Pennock, and Toby Walsh for helpful comments and suggestions. The author also thanks an anonymous IJCAI-11 reviewer of [18] for a very insightful comment that inspired part of this work.

## References

1. J. Berg, F. Nelson, and T. Rietz. Prediction market accuracy in the long run. *International Journal of Forecasting*, 24:285–300, 2008.
2. Joyce E. Berg, Robert Forsythe, Forrest D. Nelson, and Thomas A. Rietz. Results from a dozen years of election futures markets research. *The Handbook of Experimental Economics Results*, 1:742–751, 2008.
3. Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6–7):772–799, 2008.
4. Yiling Chen, Lance Fortnow, Nicolas Lambert, David M. Pennock, and Jennifer Wortman. Complexity of combinatorial market makers. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pages 190–199, Chicago, IL, USA, 2008.
5. Yiling Chen, Sharad Goel, and David M. Pennock. Pricing combinatorial markets for tournaments. In *STOC '08: Proceedings of the 40th annual ACM Symposium on Theory of Computing*, pages 305–314, 2008.
6. Yiling Chen and David M. Pennock. Designing markets for prediction. *AI Magazine*, 31:42–52, 2010.
7. Gregory F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2–3):393–405, 1990.
8. Lance Fortnow, Joe Kilian, David M. Pennock, and Michael P. Wellman. Betting Boolean-style: a framework for trading in securities based on logical formulas. *Decision Support Systems*, 39(1):87–104, 2004.
9. Vibhav Gogate and Rina Dechter. Studies in solution sampling. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 271–276, Chicago, IL, USA, 2008.
10. Robin Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119, 2003.
11. Robin Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 1:3–15, February 2007.
12. Richard M. Karp, Michael Luby, and Neal Madras. Monte-Carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10:429–448, 1989.
13. Judea Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.
14. David M. Pennock and Lirong Xia. Price updating in combinatorial prediction markets with bayesian networks. In *Proceedings of the 27th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, Barcelona, Spain, 2011.
15. Tian Sang, Paul Bearn, and Henry Kautz. Performing bayesian inference by weighted model counting. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 475–481, Pittsburgh, PA, USA, 2005.
16. Leslie Valiant. The complexity of enumeration and reliability problems. *SIAM J. Computing*, 8(3):410–421, 1979.
17. Vijay Vazirani. *Approximation Algorithms*. Springer Verlag, 2001.
18. Lirong Xia and David M. Pennock. An Efficient Monte-Carlo Algorithm for Pricing Combinatorial Prediction Markets for Tournaments. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, Barcelona, Catalonia, Spain, 2011.