

Semi-Iterative Minimum Cross-Entropy Algorithms for Rare-Events, Counting, Combinatorial and Integer Programming

Reuven Rubinstein

Faculty of Industrial Engineering and Management,
Technion, Israel Institute of Technology, Haifa, Israel

ierrr01@ie.technion.ac.il

iew3.technion.ac.il:8080/ierrr01.phtml

Published in Methodology and Computing in Applied Probability, 10, 121-178, 2008

^{0†} This research was supported by the Israel Science Foundation (grant No 191-565)

Abstract

We present a new generic minimum cross-entropy method, called the *semi-iterative MinxEnt*, or simply SME, for rare-event probability estimation, counting, and approximation of the optimal solutions of a broad class of NP-hard linear integer and combinatorial optimization problems (COP's).

The main idea of our approach is to associate with each original problem an auxiliary *single-constrained* convex MinxEnt program of a special type, which has a closed-form solution. We prove that the optimal pdf obtained from the solution of such a specially designed MinxEnt program is a zero variance pdf, provided the “temperature” parameter is set to minus infinity. In addition we prove that the parametric pdf based on the product of marginals obtained from the optimal zero variance pdf coincides with the parametric pdf of the standard *cross-entropy* (CE) method. Thus, originally designed at the end of 1990-s as a heuristics for estimation of rare-events and COP's, CE has strong connection with MinxEnt, and thus, strong mathematical foundation.

The crucial difference between the proposed SME method and the standard CE counterparts lies in their simulation-based versions: in the latter we always require to generate (via Monte Carlo) a sequence of tuples including the temperature parameter and the parameter vector in the optimal marginal pdf's, while in the former we can fix in advance the temperature parameter (to be set to a large negative number) and then generate (via Monte Carlo) a sequence of parameter vectors of the optimal marginal pdf's alone. In addition, in contrast to CE, neither the elite sample no the rarity parameter is needed in SME. As result, the proposed SME algorithm becomes simpler, faster and at least as accurate as the standard CE.

Motivated by the SME method we introduce a new updating rule for the parameter vector in the parametric pdf of the CE program. We show that the CE algorithm based on the new updating rule, called the *combined CE* (CCE), is at least as fast and accurate as its standard CE and SME counterparts. We also found numerically that the variance minimization (VM) -based algorithms are the most robust ones. We, finally, demonstrate numerically that the proposed algorithms, and in particular the CCE one, allows accurate estimation of counting quantities up to the order of hundred of decision variables and hundreds of constraints.

Contents

1	Introduction	4
2	The Classic MinxEnt Method	6
3	Rare Events, Counting and MinxEnt	9
3.1	The New Indicator-Based MinxEnt Method	11
4	The Semi-Iterative Minx-Ent (SME) Counting Algorithm	20
4.1	Standard CE and the Iterative IME	21
4.2	The Main SME Counting Algorithm	23
4.3	Extensions	25
4.4	Introducing Dependence Between the Components of \mathbf{X}	26
5	SME for Counting the Number of Feasible Solutions in an Integer Program	27
5.1	Depth- r updating for block-separable function using MinxEnt	29
6	Combining CE with SME	30
7	SME for Unconstrained Optimization, Single Event Probabilities and Counting	31
8	Applications	36
9	Numerical Results	39
9.1	Decision Making	41
9.2	Counting	42
9.2.1	Counting the Number of Feasible Solutions in Constrained Optimization Problems	45
9.2.2	“Honesty” of the Algorithms	49
9.3	Rare Events	52
9.4	Optimization	54
10	Conclusion and Further Research	55

1 Introduction

Let $H(\mathbf{x})$ be a continuous function defined on some closed bounded n -dimensional domain \mathcal{X} . Assume that \mathbf{x}^* is a unique minimum point over \mathcal{X} . The following theorem is due to Pincus [14].

Theorem 1.1 *Let $H(\mathbf{x})$ be a real-valued continuous function over closed bounded n -dimensional domain \mathcal{X} . Further assume that there is a unique minimum point \mathbf{x}^* over \mathcal{X} at which $\min_{\mathbf{x}} H(\mathbf{x})$ attains (there is no restriction on the number of local minima). Then the coordinates of x_k^* , $k = 1, \dots, n$ of \mathbf{x}^* are given by*

$$x_k^* = \lim_{\lambda \rightarrow \infty} \frac{\int_{\mathcal{X}} x_k \exp(-\lambda H(\mathbf{x})) d\mathbf{x}}{\int_{\mathcal{X}} \exp(-\lambda H(\mathbf{x})) d\mathbf{x}}, \quad k = 1, \dots, n. \quad (1)$$

The proof of the theorem is based on Laplace's formula, which for sufficiently large λ can be written as

$$\int_{\mathcal{X}} x_k \exp(-\lambda H(\mathbf{x})) d\mathbf{x} \approx x_k^* \exp(-\lambda H(\mathbf{x}^*)),$$

$$\int_{\mathcal{X}} \exp(-\lambda H(\mathbf{x})) d\mathbf{x} \approx \exp(-\lambda H(\mathbf{x}^*)).$$

This is due to fact that for large λ the major contribution to the integrals appearing in (1) comes from a small neighborhood of the minimizer \mathbf{x}^* .

Pincus' theorem holds for discrete optimization as well (assuming $|\mathcal{X}| < \infty$). In this case the integrals should be replaced by relevant sums.

There are many Monte-Carlo methods for evaluating the coordinates of \mathbf{x}^* , that is, for approximating the ratio appearing in (1). Among them is the celebrated simulated annealing method, which is based on the MCMC (Markov chain Monte Carlo), also called Metropolis' sampling procedure. The idea of the method is to sample from the Boltzmann density

$$g(\mathbf{x}) = \frac{\exp(-\lambda H(\mathbf{x}))}{\int_{\mathcal{X}} \exp(-\lambda H(\mathbf{x})) d\mathbf{x}} \quad (2)$$

without resorting to calculation of the integral (the denominator). For details see [22].

It is important to note that in general sampling from the complex multi-dimensional pdf $g(\mathbf{x})$ is a formidable task. If, however, the function $H(\mathbf{x})$ is separable that is, can be presented as

$$H(\mathbf{x}) = \sum_{k=1}^n H_k(x_k),$$

then the pdf $g(\mathbf{x})$ in (2) *decomposes* as the product of its marginal pdfs, that is, it can be written as

$$g(\mathbf{x}) = \frac{\prod_{k=1}^n \exp(-\lambda H_k(x_k))}{\prod_{k=1}^n \int_{\mathcal{X}} \exp(-\lambda H_k(x_k)) dx_k}. \quad (3)$$

Clearly, for a decomposable function $H(\mathbf{x})$ sampling from the one-dimensional marginal pdfs of $g(\mathbf{x})$ is fast.

Consider application of the simulated annealing method to *combinatorial optimization problems* (COP's). As an example, consider TSP with n cities. In this case [1], simulated annealing runs a Markov chain with $(n-1)!$ states and $H(\mathbf{x})$ denotes the length of the tour. As $\lambda \rightarrow \infty$ the stationary distribution of \mathbf{Y} will become a degenerated one, that is, it converges to the optimal solution \mathbf{x}^* (shortest

tour in the case of TSP). It can be proved [1] that in the case of multiple solutions, say R solutions, we have that as $\lambda \rightarrow \infty$ the stationary distribution of \mathbf{Y} will be *uniform* on the set of the R optimal solutions.

The main drawback of simulated annealing is that it is slow and λ , called the *annealing temperature*, must be chosen heuristically.

In this work we present a different Monte Carlo method, which we call the *semi-iterative MinxEnt*, or simply SME. It is also associated with the Boltzmann distribution, which is obtained by solving a MinxEnt program of a special type and is suitable for rare-event probability estimation, counting and approximation of the optimal solutions of a broad class of NP-hard linear integer and combinatorial optimization problems (COP's).

The main idea of our approach is to associate with each original problem an auxiliary *single-constrained* convex MinxEnt program of a special type, which has a closed form solution. We prove that the optimal pdf obtained from the solution of such specially designed MinxEnt is a zero variance pdf, provided the temperature parameter is set to minus infinity. In addition, we prove that the parametric pdf based on the product of marginals obtained from the optimal zero variance pdf coincide with the parametric pdf of the standard *cross-entropy* (CE) method. Thus, originally designed at the end of 1990-s as a heuristics for estimation of rare-events and COP's, it follows that CE has strong mathematical foundation because the proposed SME has such.

The crucial difference between the proposed SME method and CE counterparts lies in their simulation-based versions: in the latter we always require to generate (via Monte Carlo) a sequence of tuples including the temperature parameter and the parameter vector in the optimal marginal pdf's, while in the former we can fix in advance the temperature parameter (to be set to a large negative number) and then generate (via Monte Carlo) a sequence of parameter vectors of the optimal marginal pdf's alone. In addition, in contrast to CE, neither the elite sample nor the rarity parameter is needed in SME. As result, the proposed SME Algorithm becomes simpler, faster and at least as accurate as the standard CE.

Motivated by the SME method we introduce a new updating rule for the parameter vector in the parametric pdf of the CE method. We show that the CE algorithm, based on the new updating rule, called the *combined CE* (CCE), is at least as fast and accurate as its standard CE and SME counterparts. We also found numerically that the variance minimization (VM) -based algorithms are the most robust ones. We, finally, demonstrate numerically that the proposed algorithms and in particular the CCE one allows accurate estimation of counting quantities up to the order of hundred of decision variables and hundreds of constraints.

The rest of our paper is organized as follows. In Section 2 we present some background on the classic MinxEnt program. Section 3 is our main one. Here we establish connections between counting, rare-event probability estimation and MinxEnt, and we present our new MinxEnt method, which involves indicator functions in the MinxEnt programs and is called *indicator-based MinxEnt* or simply the *IME* program. We also discuss the relation of the proposed IME program to the earlier CE and MinxEnt ones considered in [17], [18] and show that the proposed program is quite different. In particular we show that the optimal pdf obtained from the IME program coincides with *zero variance* importance sampling (IS) pdf, provided the temperature parameter $\lambda = -\infty$. This is quite a remarkable result. In Section 4 we present our main SME algorithms for counting. In Section 5 we show how counting of the set of feasible solutions of LIP's (linear integer programs) can be performed with our SME algorithm. Motivated by the SME method we introduce in Section 6 the so-called *combined CE* (CCE) algorithm, which typically performs at least as fast and accurately as its SME counterpart. Section 7 deals with unconstrained optimization, where a slightly modified version of the main

SME counting algorithm is introduced. Section 8 introduces several typical LIP's and COP's like the knapsack, TSP, set covering, set partitioning and satisfiability problem, to which our algorithms are applied in Section 9. Finally, in Section 10 conclusions and some final remarks are given.

2 The Classic MinxEnt Method

The classic MinxEnt program reads as

$$\begin{aligned}
 \min_g \mathcal{D}(g, h) &= \min_g \int \ln \frac{g(\mathbf{x})}{h(\mathbf{x})} g(\mathbf{x}) d\mathbf{x} = \min_g \mathbb{E}_g \left[\ln \frac{g(\mathbf{X})}{h(\mathbf{X})} \right] \\
 \text{(P}_0\text{)} \quad \text{s.t.} \quad &\int S_i(\mathbf{x}) g(\mathbf{x}) d\mathbf{x} = \mathbb{E}_g[S_i(\mathbf{X})] = b_i, \quad i = 1, \dots, m, \\
 &\int g(\mathbf{x}) d\mathbf{x} = 1.
 \end{aligned} \tag{4}$$

Here g and h are n -dimensional joint pdfs, $S_i(\mathbf{x})$, $i = 1, \dots, m$ are given functions, and \mathbf{x} is an n -dimensional vector. Here h is assumed to be known. The program (P₀) is called the *minimum cross-entropy* or simply the MinxEnt program. If the prior h is constant, then $\mathcal{D}(g, h) = \int g(\mathbf{x}) \ln g(\mathbf{x}) d\mathbf{x} + \text{constant}$, so that the minimization of $\mathcal{D}(g, h)$ in (P₀) can be replaced with the *maximization* of

$$\mathcal{S}(g) = - \int g(\mathbf{x}) \ln g(\mathbf{x}) d\mathbf{x} = -\mathbb{E}_g[\ln g(\mathbf{X})], \tag{5}$$

where $\mathcal{S}(g)$ is the *Shannon entropy* [10]. The corresponding program is called the Jaynes' *MinxEnt* program. Note that the former minimizes the Kullback-Leibler cross-entropy, while the later maximizes the Shannon entropy [10]. For a nice paper on the generalization of MinxEnt see [4].

In typical counting and combinatorial optimization problems (COP)'s h is chosen as an n -dimensional pdf with independent uniformly distributed marginals. For example, while counting the number of satisfiability assignments in a SAT problem we assume that each component of the n -dimensional random vector \mathbf{X} is distributed $\text{Ber}(u)$ with $u = 1/2$. As for another example, when estimating rare-events in stochastic models, like queuing models, we assume that h has a fixed pdf. In particular, in an $M/M/1$ queue h would be a two-dimensional pdf with independent marginals, where the first marginal is the interarrival $\text{Exp}(\lambda)$ pdf, while the second one is the service $\text{Exp}(\mu)$ pdf.

The MinxEnt program, which under mild conditions [3] presents a convex constrained functional optimization problem, can be solved via Lagrange multipliers. The solution is given by [3]

$$g(\mathbf{x}) = \frac{h(\mathbf{x}) \exp \left\{ - \sum_{i=1}^m \lambda_i S_i(\mathbf{x}) \right\}}{\mathbb{E}_h \left[\exp \left\{ - \sum_{i=1}^m \lambda_i S_i(\mathbf{X}) \right\} \right]}, \tag{6}$$

where λ_i , $i = 1, \dots, m$ are obtained from the solution of the following system of equations

$$\frac{\mathbb{E}_h \left[S_i(\mathbf{X}) \exp \left\{ - \sum_{j=1}^m \lambda_j S_j(\mathbf{X}) \right\} \right]}{\mathbb{E}_h \left[\exp \left\{ - \sum_{j=1}^m \lambda_j S_j(\mathbf{X}) \right\} \right]} = b_i. \tag{7}$$

Note that $g(\mathbf{x})$ can be written as

$$g(\mathbf{x}) = C(\lambda_1, \dots, \lambda_m) h(\mathbf{x}) \exp \left\{ - \sum_{i=1}^m \lambda_i S_i(\mathbf{x}) \right\}, \tag{8}$$

where

$$C^{-1}(\lambda_1, \dots, \lambda_m) = \mathbb{E}_h \left[\exp \left\{ - \sum_{i=1}^m \lambda_i S_i(\mathbf{X}) \right\} \right] \quad (9)$$

is the normalization constant. Note also that $g(\mathbf{x})$ presents a density function, that is, $g(\mathbf{x}) \geq 0$.

In the particular case where each $S_i(\mathbf{X})$, $\mathbf{X} = (X_1, \dots, X_n)$ is *coordinate-wise separable*, that is,

$$S_i(\mathbf{X}) = \sum_{k=1}^n S_{ik}(X_k), \quad i = 1, \dots, m \quad (10)$$

and the components $X_k, k = 1, \dots, n$ of the random vector $\mathbf{X} = (X_1, \dots, X_n)$ distributed $h(\mathbf{x})$ are independent, the joint pdf $g(\mathbf{x})$ in (6) reduces to the *product of marginal pdfs*. In such case we say that $g(\mathbf{x})$ is *decomposable*.

In particular, the k -th component of $g(\mathbf{x})$ can be written as

$$g_k(x_k) = \frac{h_k(x) \exp \left\{ - \sum_{i=1}^m \lambda_i S_{ik}(x_k) \right\}}{\mathbb{E}_{h_k} \left[\exp \left\{ - \sum_{i=1}^m \lambda_i S_{ik}(X_k) \right\} \right]}, \quad k = 1, \dots, n. \quad (11)$$

Remark 2.1 It is well known [5] that the optimal solution of the single-dimensional single-constrained MinxEnt program

$$\begin{aligned} \min_g \mathcal{D}(g, h) &= \min_g \mathbb{E}_g \left[\ln \frac{g(X)}{h(X)} \right] \\ \text{s.t. } \mathbb{E}_g[S(X)] &= b, \\ \int g(x) dx &= 1 \end{aligned} \quad (12)$$

coincides with the celebrated optimal *exponential change of measure* (ECM). Note that typically in a multi-dimensional ECM one twists each component separately, using possibly different twisting parameters. In contrast, the optimal solution to the MinxEnt program is parameterized by a *single-dimensional* parameter λ , so for the multi dimensional case ECM differs from MinxEnt.

Example 2.1 (Die Tossing) To obtain better insight into the MinxEnt program, consider a particular case of (12) associated with die tossing. We assume $S(x) = x$ and $h(x) = h(x; \mathbf{u})$ is a discrete distribution over the 6 faces of the die, where $\mathbf{u} = (u_1, \dots, u_6)$ denotes the parameter vector. In this case it is readily seen the the functional program (12) leads to the following parametric one

$$\begin{aligned} \min_{\mathbf{p}} \mathcal{D}(\mathbf{p}|\mathbf{u}) &= \min_{\mathbf{p}} \sum_{k=1}^6 p_k \ln \frac{p_k}{u_k} \\ \text{s.t. } \sum_{k=1}^6 k p_k &= b, \\ \sum_{k=1}^6 p_k &= 1. \end{aligned} \quad (13)$$

The optimal parameter vector $\mathbf{p}^* = (p_1, \dots, p_6)$, derived from the solution of (13) can be written component wise as

$$p_k = \frac{u_k \exp \{-k\lambda\}}{\sum_{r=1}^6 u_r \exp \{-r\lambda\}} = \frac{\mathbb{E}_{\mathbf{u}}[I_{\{X=k\}} \exp \{-X\lambda\}]}{\mathbb{E}_{\mathbf{u}}[\exp \{-X\lambda\}]}, \quad k = 1, \dots, 6, \quad (14)$$

where λ is derived from the numerical solution of

$$\frac{\sum_{k=1}^6 k u_k \exp \{-k\lambda\}}{\sum_{k=1}^6 u_k \exp \{-k\lambda\}} = b. \quad (15)$$

Table 1, which is an exact replica of Table 4.1 of [10], presents λ , \mathbf{p} and the entropy $\mathcal{S}(\mathbf{p})$ as functions of b for a fair die, that is, with the prior ($u_1 = \frac{1}{6}, \dots, u_6 = \frac{1}{6}$). The table is self-explanatory.

Table 1: λ , \mathbf{p} and $\mathcal{S}(\mathbf{p})$ as function of b for a fair die.

b	λ	p_1	p_2	p_3	p_4	p_5	p_6	$\mathcal{S}(\mathbf{p})$
1.0	∞	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.00000
1.5	1.0870	0.6637	0.2238	0.0755	0.0255	0.0086	0.0029	0.95356
2.0	0.6296	0.4781	0.2548	0.1357	0.0723	0.0385	0.0205	1.36724
2.5	0.3710	0.3475	0.2398	0.1654	0.1142	0.0788	0.0544	1.61373
3.0	0.1746	0.2468	0.2072	0.1740	0.1461	0.1227	0.1031	1.74843
3.5	0.0000	0.1666	0.1666	0.1666	0.1666	0.1666	0.1666	1.79176
4.0	-0.1746	0.1031	0.1227	0.1461	0.1740	0.2072	0.2468	1.74843
4.5	-0.3710	0.0544	0.0788	0.1142	0.1654	0.2398	0.3475	1.61373
5.0	-0.6296	0.0205	0.0385	0.0723	0.1357	0.2548	0.4781	1.36724
5.5	-1.0870	0.0029	0.0086	0.0255	0.0755	0.2238	0.6637	0.95356
6.0	$-\infty$	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.00000

Note that

- $\mathbf{p}(b = 3.5) = \mathbf{u} = (\frac{1}{6}, \dots, \frac{1}{6})$ and, thus $g = h$.
- $\mathcal{S}(\mathbf{p})$ is strictly concave in b and the maximal entropy $\max_b \mathcal{S}(\mathbf{p}) = \mathcal{S}(3.5) = 1.79176$.
- For the extreme values of b , that is, for $b = 6$ and $b = 1$, the corresponding optimal solutions are $\mathbf{p}^* = (0, 0, \dots, 1)$ and $\mathbf{p}^* = (1, 0, \dots, 0)$ respectively, that is, the pdf g becomes degenerated. For these cases
 1. The entropy is $\mathcal{S}(\mathbf{p}) = 0$, and thus there is no uncertainty (for both degenerated vectors, $\mathbf{p} = (0, 0, \dots, 1)$ and $\mathbf{p} = (1, 0, \dots, 0)$).
 2. For $\mathbf{p} = (0, 0, \dots, 1)$ and $\mathbf{p} = (1, 0, \dots, 0)$ we have that $\lambda = -\infty$ and $\lambda = \infty$, respectively. This important observation is in the spirit of Pincus [14] Theorem 1.1 and will play an important role below.
 3. It can also be readily shown that \mathbf{p} is degenerated regardless of the prior \mathbf{u} .

The above observations for the die example can be readily extended to the case where instead of $S(x) = x$ one considers $S(x) = \sum_{k=1}^r a_k I_{x=k}$ with $r > 1$ and with arbitrary a_k 's.

Remark 2.2 Taking into account that MaxEnt (with the objective function $\mathcal{S}(g)$, see (5)) can be viewed as a particular case of MinxEnt with $h(\mathbf{x}) = \text{const}$, we can rewrite the basic MinxEnt formulas (6) and (7) as

$$g(\mathbf{x}) = \frac{\exp \left\{ -\sum_{i=1}^m \lambda_i S_i(\mathbf{x}) \right\}}{\sum_{\mathbf{x}'} \left[\exp \left\{ -\sum_{i=1}^m S_i(\mathbf{x}) \lambda_i \right\} \right]}, \quad (16)$$

where λ_i , $i = 1, \dots, m$ are obtained from the solution of the following system of equations

$$\frac{\sum_{\mathbf{x}} S_i(\mathbf{x}) \exp \left\{ -\sum_{j=1}^m \lambda_j S_j(\mathbf{x}) \right\}}{\sum_{\mathbf{x}} \exp \left\{ -\sum_{j=1}^m \lambda_j S_j(\mathbf{x}) \right\}} = b_i. \quad (17)$$

We extend next the MinxEnt program (P_0) to both equality and inequality constraints, that is, we consider the following program

$$\begin{aligned}
& \min_g \mathcal{D}(g, h) = \min_g \int \ln \frac{g(\mathbf{x})}{h(\mathbf{x})} g(\mathbf{x}) d\mathbf{x} = \min_g \mathbb{E}_g \left[\ln \frac{g(\mathbf{X})}{h(\mathbf{X})} \right] \\
(P_{00}) \quad & \text{s.t.} \quad \int S_i(\mathbf{x})g(\mathbf{x}) d\mathbf{x} = \mathbb{E}_g[S_i(\mathbf{X})] = b_i, \quad i = 1, \dots, m_1, \\
& \int S_j(\mathbf{x})g(\mathbf{x}) d\mathbf{x} = \mathbb{E}_g[S_j(\mathbf{X})] \geq b_j, \quad j = m_1 + 1, \dots, m_1 + m_2, \\
& \int g(\mathbf{x})d\mathbf{x} = 1.
\end{aligned} \tag{18}$$

In this case applying the Kuhn-Tucker conditions to the program (P_{00}) we readily obtain that $g(\mathbf{x})$ remains the same as in (6), while $\boldsymbol{\lambda} = (\lambda_0, \lambda_1, \dots, \lambda_m)$, $m = m_1 + m_2$ are found from the solution of the following convex program

$$\begin{aligned}
& \max_{\boldsymbol{\lambda}} \left(-\sum_{i=0}^m \lambda_i b_i - \mathbb{E}_h[\exp \{-\sum_{i=0}^m \lambda_i S_i(\mathbf{X})\}] \right) \\
\text{s.t.} \quad & \lambda_j \geq 0, \quad \forall j = m_1 + 1, \dots, m_1 + m_2.
\end{aligned} \tag{19}$$

3 Rare Events, Counting and MinxEnt

Here we establish the connection between rare-event probabilities, MinxEnt and counting. In particular we discuss how to employ MinxEnt for estimating the following rare-event probability

$$\ell = \mathbb{E}_{\mathbf{u}} [I_{\{S(\mathbf{X}) \geq b\}}], \tag{20}$$

where $S(\mathbf{X})$ is quite an arbitrary sample function, $\mathbf{X} \sim f(\mathbf{x}; \mathbf{u})$, where $f(\mathbf{x}; \mathbf{u})$ is a fixed distribution parametrized by a vector \mathbf{u} , and b is large number, so ℓ is a very small probability.

We can estimate ℓ using the following non-parametric IS estimator

$$\widehat{\ell} = \frac{1}{N} \sum_{k=1}^N \left[I_{\{S(\mathbf{X}_k) \geq b\}} \frac{f(\mathbf{X}_k; \mathbf{u})}{g(\mathbf{X}_k)} \right], \tag{21}$$

or using a parametric one

$$\widehat{\ell} = \frac{1}{N} \sum_{k=1}^N \left[I_{\{S(\mathbf{X}_k) \geq b\}} \frac{f(\mathbf{X}_k; \mathbf{u})}{f(\mathbf{X}_k; \mathbf{p})} \right], \tag{22}$$

respectively. Here $\mathbf{X}_1, \dots, \mathbf{X}_N$ in (21) and (22) is a random sample from $g(\mathbf{x})$ and from $f(\mathbf{x}; \mathbf{p})$, respectively. Note that \mathbf{p} is a parameter vector, that is, typically different from \mathbf{u} . At this point it is crucial to note that in order to obtain a low-variance estimator $\widehat{\ell}$ we shall use below $g(\mathbf{x})$ and \mathbf{p} obtained from the MinxEnt program.

If not stated otherwise we assume below that $f(\mathbf{x}; \mathbf{u})$ is the *uniform* distribution. Since any counting quantity can be derived using the probability (20) (see [18]), we shall use the *same* IS pdfs $g(\mathbf{x})$ and $f(\mathbf{x}; \mathbf{p})$ given in (21) and (22) to estimate the counting quantity, denoted by $|\mathcal{X}^*|$. It is readily seen [18] that using $g(\mathbf{x})$ and $f(\mathbf{x}; \mathbf{p})$, the estimator of $|\mathcal{X}^*|$ can be written as

$$|\widehat{\mathcal{X}^*}| = \frac{1}{N} \sum_{k=1}^N I_{\{S(\mathbf{X}_k) \geq b\}} \frac{1}{g(\mathbf{X}_k)}, \tag{23}$$

and as

$$|\widehat{\mathcal{X}^*}| = \frac{1}{N} \sum_{k=1}^N I_{\{S(\mathbf{X}_k) \geq b\}} \frac{1}{f(\mathbf{X}_k; \mathbf{p})}, \quad (24)$$

respectively, provided $f(\mathbf{x}; \mathbf{u})$ is a uniform distribution. Here again $\mathbf{X}_1, \dots, \mathbf{X}_N$ is a random sample either from $f(\mathbf{x}; \mathbf{p})$ or from g .

To establish the connection between MinxEnt and rare-events note (see [17]) that while estimating rare events probabilities using (21) and (22) as IS pdfs $g(\mathbf{x})$ and $f(\mathbf{x}; \mathbf{p})$, it is common to take the ones obtained from the solution of the following *single constrained* MinxEnt program

$$\begin{aligned} \min_g \mathcal{D}(g, h) &= \min_g \int \ln \frac{g(\mathbf{x})}{h(\mathbf{x})} g(\mathbf{x}) d\mathbf{x} = \min_g \mathbb{E}_g \left[\ln \frac{g(\mathbf{X})}{h(\mathbf{X})} \right] \\ \text{s.t. } \mathbb{E}_g[S(\mathbf{X})] &= b, \\ \int g(\mathbf{x}) d\mathbf{x} &= 1. \end{aligned} \quad (25)$$

In other words, one typically takes the optimal MinxEnt pdf $g(\mathbf{x})$ derived from (25) as the importance sampling pdf in (21). Alternatively, if $g(\mathbf{x})$ is a complex pdf, (which is typically the case) one can *approximate* $g(\mathbf{x})$ by the product of its *marginal* pdf's $g_i(x_i) = f_i(x_i, p_i)$, $i = 1, \dots, n$ [17], that is use (22) instead of (21), where $f(\mathbf{x}; \mathbf{p})$ is a parametric pdf, that differs from the prior pdf $h(\mathbf{x}) = f(\mathbf{x}; \mathbf{u})$ only in \mathbf{p} . For an alternative approach of using the product of marginal pdf's of $g(\mathbf{x})$ for rare event estimation see [8].

We shall explain now how to derive \mathbf{p} from the optimal pdf $g(\mathbf{x})$. If not stated otherwise we assume for simplicity that $\mathbf{X} = (X_1, \dots, X_n)$ is a binary random vector with probabilities $\mathbf{u} = (u_1, \dots, u_n)$ and independent components. In other words $\mathbf{X} \sim \text{Ber}(\mathbf{u})$. By summing $g(\mathbf{x})$ over all $x_k, k \neq j$, we obtain the marginal pdf for its j -th component and we approximate $g(\mathbf{x})$ by the product of these marginals. In particular, let $g(\mathbf{x})$ be the optimal MinxEnt pdf derived from (25), and $h(\mathbf{x}) = f(\mathbf{x}; \mathbf{u})$ the prior pdf, then under g we have $X_j \sim \text{Ber}(p_j)$, with

$$p_j = \mathbb{E}_g[X_j] = \frac{\sum_{\mathbf{x}} x_j h(\mathbf{x}) e^{-\lambda S(\mathbf{x})}}{\mathbb{E}_{\mathbf{u}}[e^{-\lambda S(\mathbf{X})}]},$$

so that

$$p_j = \frac{\mathbb{E}_{\mathbf{u}}[X_j \exp\{-S(\mathbf{X})\lambda\}]}{\mathbb{E}_{\mathbf{u}}[\exp\{-S(\mathbf{X})\lambda\}]}, \quad j = 1, \dots, n \quad (26)$$

with λ satisfying (7) (for $m = 1$).

It is important to note that formula (26) is similar to the corresponding CE one [21]

$$p_j = \frac{\mathbb{E}_{\mathbf{u}}[X_j I_{\{S(\mathbf{X}) \geq b\}}]}{\mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X}) \geq b\}}]}, \quad (27)$$

with one main difference: the indicator function $I_{\{S(\mathbf{X}) \geq b\}}$ in the CE formula is replaced by $\exp\{-\lambda S(\mathbf{X})\}$.

Remark 3.1 (Exponential Families) Formula (26) can be generalized such that it holds for any exponential family parameterized by the mean, in the same way that the CE formula (27) holds for such families. More specifically, suppose under prior $h(\mathbf{x}) = f(\mathbf{x}; \mathbf{u})$ the random vector $\mathbf{X} = (X_1, \dots, X_n)$ has independent components and that each X_i is distributed according to some 1-parameter exponential family $f_i(x_i; u_i)$ and is parameterized by its mean — thus, $\mathbb{E}_h[X_i] = \mathbb{E}_{\mathbf{u}}[X_i] = u_i$, with

$\mathbf{u} = (u_1, \dots, u_n)$. The expectation of X_j under the MinxEnt solution is (in the continuous case)

$$\mathbb{E}_g[X_j] = \frac{\int_{\mathcal{X}} x_j h(\mathbf{x}) e^{-\lambda S(\mathbf{x})} d\mathbf{x}}{\mathbb{E}_{\mathbf{u}}[e^{-\lambda S(\mathbf{X})}]} = \frac{\mathbb{E}_{\mathbf{u}}[X_j e^{-\lambda S(\mathbf{X})}]}{\mathbb{E}_{\mathbf{u}}[e^{-\lambda S(\mathbf{X})}]}, \quad j = 1, \dots, n, \quad (28)$$

Let $\mathbf{v} = (v_1, \dots, v_n)$ be another parameter vector for the exponential family. Then the above analysis suggests to carry out the importance sampling using v_j equal to $\mathbb{E}_g[X_j]$ given in (28), so that

$$v_j = \frac{\mathbb{E}_{\mathbf{u}}[X_j e^{-\lambda S(\mathbf{X})}]}{\mathbb{E}_{\mathbf{u}}[e^{-\lambda S(\mathbf{X})}]}, \quad j = 1, \dots, n, \quad (29)$$

The diagram connecting rare-events, MinxEnt and counting can be represented as

$$\{\mathbf{x} \in \mathbb{R}^n : S(\mathbf{x}) \geq b\} \longrightarrow \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq b\}} \longrightarrow \text{MinxEnt (26)} \longrightarrow \text{Count as (24)}. \quad (30)$$

Note that the p_j 's in (26) and (27) were extensively used in [17] for rare-event estimation and COP's while updating the parameter vector \mathbf{p} using simulation. In this paper we shall use a different approach for deriving $g(\mathbf{x})$ and the associated parameter vector \mathbf{p} .

3.1 The New Indicator-Based MinxEnt Method

Consider counting on the set

$$\mathcal{X}^* = \{\mathbf{x} \in \mathbb{R}^n : S_i(\mathbf{x}) \geq b_i, \quad j = 1, \dots, m\}, \quad (31)$$

where $S_i(\mathbf{x})$, $i = 1, \dots, m$ are arbitrary functions. In this case we can associate with (31) the following *multiple-event* probability

$$\ell = \mathbb{P}_{\mathbf{u}} \left\{ \bigcap_{i=1}^m [S_i(\mathbf{X}) \geq b_i] \right\} = \mathbb{E}_{\mathbf{u}} \left[\prod_{i=1}^m I_{\{S_i(\mathbf{X}) \geq b_i\}} \right]. \quad (32)$$

Note that (32) extends (20) in the sense that it involves simultaneously an *intersection* of m events $\{S_i(\mathbf{X}) \geq b_i\}$, that is, *multiple* events rather than a *single* one $\{S(\mathbf{X}) \geq b\}$. Note also that some of the constraints may be equality ones, that is, $\{S_i(\mathbf{X}) = b_i\}$. Note finally that (32) has some interesting applications in rare-event simulation. For example, in a queueing model one might be interested in estimating the probability of the simultaneous occurrence of two events, $\{S_1(\mathbf{X}) \geq b_1\}$ and $\{S_2(\mathbf{X}) \geq b_2\}$, where the first is associated with buffer overflow (the number of customers S_1 is at least b_1), and the second is associated with the sojourn time (the waiting time of the customers S_2 in the queueing system is at least b_2).

We assume that each individual event $\{S_i(\mathbf{X}) \geq b_i\}$, $i = 1, \dots, m$, is *not rare*, that is each probability $\mathbb{P}_{\mathbf{u}}\{S_i(\mathbf{X}) \geq b_i\}$ is not a rare-event probability, say $\mathbb{P}_{\mathbf{u}}\{S_i(\mathbf{X}) \geq b_i\} \geq 10^{-4}$, but their intersection forms a *rare-event probability* ℓ . Similar to the single-event case in (20) we are interested in efficient estimation of ℓ defined in (32). As before, we shall use the IS estimators (22) and (24). The crucial issue is how to approximate efficiently $g(\mathbf{x})$ and in particular how to estimate efficiently the parameter vector \mathbf{p} in $f(\mathbf{x}, \mathbf{p})$.

The main idea of the new approach is to design an IS pdf $g(\mathbf{x})$ such that under $g(\mathbf{x})$ all constraints $\{S_i(\mathbf{x}) \geq b_i, \quad i = 1, \dots, m\}$ are fulfilled. This is equivalent of saying that the rare-event probability ℓ in (32) becomes certain under $g(\mathbf{x})$, that is,

$$\mathbb{E}_g \left[\prod_{i=1}^m I_{\{S_i(\mathbf{X}) \geq b_i\}} \right] = 1. \quad (33)$$

In other words, (33) states that under such an *ideal* IS pdf $g(\mathbf{x})$ all m indicators must be equal to unity with probability 1. This can also be written as

$$\mathbb{P}_g \left\{ \left(\sum_{i=1}^m C_i(\mathbf{X}) \right) = m \right\} = \mathbb{E}_g [I_{\{\mathcal{C}(\mathbf{X})=m\}}] = 1, \quad (34)$$

where

$$\mathcal{C}(\mathbf{X}) = \sum_{i=1}^m C_i(\mathbf{X}) = \sum_{i=1}^m I_{\{S_i(\mathbf{X}) \geq b_i\}}, \quad (35)$$

and $C_i(\mathbf{X}) = I_{\{S_i(\mathbf{X}) \geq b_i\}}$. Similar to (33), formula (34) states that under $g(\mathbf{x})$ the probability of the sum of m indicator random variables $C_i(\mathbf{X})$ being equal to m , (m is the number of constraints) must be equal 1.

Lets pose the following questions:

- Does such an ideal IS $g(\mathbf{x})$ exist in the frame work of MinxEnt?
- If so, is it uniformly distributed over the desired set \mathcal{X}^* , or, in other words, is $g(\mathbf{x})$ a zero-variance IS pdf ?

We shall show next that the answer to both questions is affirmative.

It follows from the above that in order for $g(\mathbf{x})$ to fulfill all the constraints $\{S_i(\mathbf{x}) \geq b_i, i = 1, \dots, m\}$ we need to consider an MinxEnt program with the following *single* constraint

$$\mathbb{E}_g \left(\sum_{i=1}^m C_i(\mathbf{X}) \right) = m. \quad (36)$$

That is, similar to (25) we define the following single-constrained MinxEnt program

$$\begin{aligned} \min_g \mathcal{D}(g, h) &= \min_g \int \ln \frac{g(\mathbf{x})}{h(\mathbf{x})} g(\mathbf{x}) d\mathbf{x} = \min_g \mathbb{E}_g \left[\ln \frac{g(\mathbf{X})}{h(\mathbf{X})} \right] \\ \text{s.t. } \mathbb{E}_g [\sum_{i=1}^m C_i(\mathbf{X})] &= m \\ \int g(\mathbf{x}) d\mathbf{x} &= 1. \end{aligned} \quad (37)$$

In other words, in order to estimate the rare-event probability ℓ given in (32) and to count the cordinality of the set (31) we shall use the single-constrained MinxEnt program (37). The solution of (37), which is based on the sum of the indicator random variables $C_i(\mathbf{X})$ is

$$g(\mathbf{x}) = \frac{h(\mathbf{x}, \mathbf{u}) \exp \left\{ -\lambda \sum_{i=1}^m C_i(\mathbf{x}) \right\}}{\mathbb{E}_{\mathbf{u}} \left[\exp \left\{ -\sum_{i=1}^m \lambda C_i(\mathbf{X}) \right\} \right]}, \quad (38)$$

where λ is obtained from the solution of the following equation

$$\frac{\mathbb{E}_{\mathbf{u}} \left[\sum_{i=1}^m C_i(\mathbf{X}) \exp \left\{ -\lambda \sum_{j=1}^m C_j(\mathbf{X}) \right\} \right]}{\mathbb{E}_{\mathbf{u}} \left[\exp \left\{ -\lambda \sum_{j=1}^m C_j(\mathbf{X}) \right\} \right]} = m. \quad (39)$$

It is important to note that we can declare that the set $\{S_i(\mathbf{x}) \geq b_i, i = 1, \dots, m\}$ is empty, provided (39) has no solution.

It is crucial to note that the classic *multi-constrained* MinxEnt program (4) involves *expectations of $S_i(\mathbf{X})$* , while the proposed single-constrained one (37) is based on the *expectations of the indicators of $S_i(\mathbf{X})$* , so the name indicator MinxEnt program or simply IME program.

For $m = 1$ the IME program (37) reduces to

$$\begin{aligned} \min_g \mathcal{D}(g, h) &= \min_g \mathbb{E}_g \left[\ln \frac{g(\mathbf{X})}{h(\mathbf{X})} \right] \\ \text{s.t. } \mathbb{E}_g[C(\mathbf{X})] &= 1 \\ \int g(\mathbf{x}) d\mathbf{x} &= 1, \end{aligned} \tag{40}$$

where $C(\mathbf{X}) = I_{\{S(\mathbf{X}) \geq b\}}$.

Observe also that in this case the single-constrained programs (40) and (25) do not coincide: in the former case we use an expectation of the indicator of $S(\mathbf{X})$, that is $\mathbb{E}\{I_{\{S(\mathbf{X}) \geq b\}}\}$, while in the later case we use an expectation of $S(\mathbf{X})$, that is, $\mathbb{E}\{S(\mathbf{X})\}$. We shall treat the program (40) in more details in Section 7.

The following Lemmas 3.1-3.3 provide affirmative answers to the questions posed above.

Lemma 3.1 *The optimal λ of the IME program (37) satisfying (39) is $\lambda = -\infty$.*

Proof The proof is given for a discrete domain \mathcal{X} . For a continuous domain we replace the summations by integrations.

To prove that the optimal λ of the IME program (37) is $\lambda = -\infty$ we proceed as follows. Denoting, as before, $\mathcal{C}(\mathbf{x}) = \sum_{i=1}^m C_i(\mathbf{x}) \in \{0, 1, \dots, m\}$ we can write (39) as

$$\begin{aligned} & \lim_{\lambda \rightarrow -\infty} \frac{\mathbb{E}_{\mathbf{u}}[\sum_{i=1}^m C_i(\mathbf{X}) \exp\{-\lambda \sum_{j=1}^m C_j(\mathbf{X})\}]}{\mathbb{E}_{\mathbf{u}}[\exp\{-\lambda \sum_{j=1}^m C_j(\mathbf{X})\}]} \\ &= \lim_{\lambda \rightarrow -\infty} \sum_{\mathbf{x} \in \mathcal{X}} h(\mathbf{x}; \mathbf{u}) \mathcal{C}(\mathbf{x}) e^{-\lambda \mathcal{C}(\mathbf{x})} \left(\sum_{\mathbf{x} \in \mathcal{X}} h(\mathbf{x}; \mathbf{u}) e^{-\lambda \mathcal{C}(\mathbf{x})} \right)^{-1} \\ &= \lim_{\lambda \rightarrow -\infty} \left(\sum_{\mathbf{x} \text{ s.t. } \mathcal{C}(\mathbf{x})=m} h(\mathbf{x}; \mathbf{u}) \cdot m \cdot e^{-\lambda m} + \sum_{\mathbf{x} \text{ s.t. } \mathcal{C}(\mathbf{x}) < m} h(\mathbf{x}; \mathbf{u}) \mathcal{C}(\mathbf{x}) e^{-\lambda \overbrace{\mathcal{C}(\mathbf{x})}^{< m}} \right) \\ & \quad \times \left(\sum_{\mathbf{x} \text{ s.t. } \mathcal{C}(\mathbf{x})=m} h(\mathbf{x}; \mathbf{u}) e^{-\lambda m} + \sum_{\mathbf{x} \text{ s.t. } \mathcal{C}(\mathbf{x}) < m} h(\mathbf{x}; \mathbf{u}) e^{-\lambda \overbrace{\mathcal{C}(\mathbf{x})}^{< m}} \right)^{-1}. \end{aligned}$$

The right sum in each of the two factors above is negligible with respect to its corresponding left sum. Therefore, the above expression equals

$$\begin{aligned} & \lim_{\lambda \rightarrow -\infty} \left(\sum_{\mathbf{x} \text{ s.t. } \mathcal{C}(\mathbf{x})=m} h(\mathbf{x}; \mathbf{u}) \cdot m \cdot e^{-\lambda m} \right) \left(\sum_{\mathbf{x} \text{ s.t. } \mathcal{C}(\mathbf{x})=m} h(\mathbf{x}; \mathbf{u}) e^{-\lambda m} \right)^{-1} \\ &= \lim_{\lambda \rightarrow -\infty} \frac{\mathbb{P}_{\mathbf{u}}(\mathcal{C}(\mathbf{X})=m) \cdot m \cdot e^{-\lambda m}}{\mathbb{P}_{\mathbf{u}}(\mathcal{C}(\mathbf{X})=m) e^{-\lambda m}} = m. \end{aligned}$$

□

The following lemma answers our second question.

Lemma 3.2 *The optimal pdf $g(\mathbf{x})$ in (38) corresponds to a uniform pdf over the set $\{\mathbf{x} \in \mathbb{R}^n : S_i(\mathbf{x}) \geq b_i, i = 1, \dots, m\}$.*

Proof The proof is given for a discrete domain \mathcal{X} . For a continuous domain we replace the summations by integrations. By Lemma 3.1, $\lambda \rightarrow -\infty$ and, denoting as before $\mathcal{C}(\mathbf{x}) = \sum_{i=1}^m C_i(\mathbf{x}) \in \{0, 1, \dots, m\}$, we can write (38) as

$$\begin{aligned}
g(\mathbf{x}) &= \lim_{\lambda \rightarrow -\infty} h(\mathbf{x}; \mathbf{u}) e^{-\lambda \mathcal{C}(\mathbf{x})} \cdot \left(\sum_{\mathbf{x}' \in \mathcal{X}} h(\mathbf{x}'; \mathbf{u}) e^{-\lambda \mathcal{C}(\mathbf{x}')} \right)^{-1} \\
&= \lim_{\lambda \rightarrow -\infty} h(\mathbf{x}; \mathbf{u}) e^{-\lambda \mathcal{C}(\mathbf{x})} \\
&\cdot \left(\sum_{\mathbf{x}' \text{ s.t. } \mathcal{C}(\mathbf{x}')=m} h(\mathbf{x}'; \mathbf{u}) e^{-\lambda m} + \sum_{\mathbf{x}' \text{ s.t. } \mathcal{C}(\mathbf{x}') < m} h(\mathbf{x}'; \mathbf{u}) e^{-\lambda \overbrace{\mathcal{C}(\mathbf{x}')}^{< m}} \right)^{-1}.
\end{aligned}$$

The right-hand side sum is negligible with respect to the left one. Therefore,

$$\begin{aligned}
g(\mathbf{x}) &= \lim_{\lambda \rightarrow -\infty} h(\mathbf{x}; \mathbf{u}) e^{-\lambda \mathcal{C}(\mathbf{x})} \cdot \left(\sum_{\mathbf{x}' \text{ s.t. } \mathcal{C}(\mathbf{x}')=m} h(\mathbf{x}'; \mathbf{u}) e^{-\lambda m} \right)^{-1} = \\
&\lim_{\lambda \rightarrow -\infty} \frac{h(\mathbf{x}; \mathbf{u}) e^{-\lambda \mathcal{C}(\mathbf{x})}}{\mathbb{P}_{\mathbf{u}}(\mathcal{C}(\mathbf{X})=m) e^{-\lambda m}} = \lim_{\lambda \rightarrow -\infty} \frac{h(\mathbf{x}; \mathbf{u})}{\mathbb{P}_{\mathbf{u}}(\mathcal{C}(\mathbf{X})=m) e^{-\lambda(m-\mathcal{C}(\mathbf{x}))}} = \\
&\begin{cases} 0 & , \mathcal{C}(\mathbf{x}) \in \{0, 1, \dots, m-1\}; \\ h(\mathbf{x}; \mathbf{u}) / \mathbb{P}_{\mathbf{u}}(\mathcal{C}(\mathbf{X})=m) & , \mathcal{C}(\mathbf{x}) = m. \end{cases}
\end{aligned}$$

Now, substituting $|\widehat{\mathcal{X}^*}| = |\mathcal{X}| \widehat{\ell}$ into the IS estimator,

$$\widehat{\ell} = \frac{1}{N} \sum_{k=1}^N I_{\{\mathcal{C}(\mathbf{X}_k)=m\}} \frac{h(\mathbf{X}_k; \mathbf{u})}{g(\mathbf{X}_k)}, \quad \mathbf{X}_k \sim g,$$

and taking into account that $\mathbb{P}_g(\mathcal{C}(\mathbf{X})=m) = 1$, we finally obtain

$$\widehat{\ell} = \frac{1}{N} \sum_{k=1}^N \frac{h(\mathbf{X}_k; \mathbf{u})}{h(\mathbf{X}_k; \mathbf{u}) / \mathbb{P}_{\mathbf{u}}(\mathcal{C}(\mathbf{X})=m)} = \mathbb{P}_{\mathbf{u}}(\mathcal{C}(\mathbf{X})=m),$$

which is a constant. \square

Lemma 3.2 automatically implies that the optimal $g(\mathbf{x})$ is a zero-variance IS. Thus, solving the MinxEnt program (37) we obtained a zero variance IS sampling pdf $g(\mathbf{x}, \lambda)$ with $\lambda = -\infty$. But this is exactly what Pincus' Theorem 1.1 requires.

Lemma 3.3 *For $\lambda = -\infty$ the optimal IME pdf $g(\mathbf{x})$ in (38) coincides with the classic IS zero-variance pdf.*

Proof We will show that for $\lambda = -\infty$ the optimal IME pdf $g(\mathbf{x})$ in (38), that is,

$$g(\mathbf{x}) = \frac{h(\mathbf{x}, \mathbf{u}) e^{-\lambda \mathcal{C}(\mathbf{x})}}{\mathbb{E}_{\mathbf{u}} [e^{-\lambda \mathcal{C}(\mathbf{X})}]},$$

coincides with the classic zero variance pdf

$$g^*(\mathbf{x}) = \frac{h(\mathbf{x}, \mathbf{u}) I_{\{\mathcal{C}(\mathbf{x})=m\}}}{\mathbb{E}_{\mathbf{u}} [I_{\{\mathcal{C}(\mathbf{X})=m\}}]},$$

where $\mathcal{C}(\mathbf{x}) = \sum_{i=1}^m C_i(\mathbf{x})$.

Our proof assumes a discrete domain \mathcal{X} . For a continuous domain replace the summations by integrations.

Noting that

$$I_{\{\mathcal{C}(\mathbf{x})=m\}} = I_{\{\mathbf{x} \in \mathcal{X}^*\}}, \quad \mathbf{x} \in \mathcal{X}$$

we have

$$\begin{aligned}
g(\mathbf{x}) &\triangleq \lim_{\lambda \rightarrow -\infty} \frac{h(\mathbf{x}, \mathbf{u}) e^{-\lambda C(\mathbf{x})}}{\mathbb{E}_{\mathbf{u}}[e^{-\lambda C(\mathbf{x})}]} = h(\mathbf{x}, \mathbf{u}) \lim_{\lambda \rightarrow -\infty} \frac{e^{-\lambda C(\mathbf{x})}}{|\mathcal{X}|^{-1} \sum_{\mathbf{x} \in \mathcal{X}} e^{-\lambda C(\mathbf{x})}} \\
&= h(\mathbf{x}, \mathbf{u}) \lim_{\lambda \rightarrow -\infty} \frac{I_{\mathbf{w} \in \mathcal{X}^*} e^{-\lambda m} + I_{\mathbf{w} \notin \mathcal{X}^*} e^{-\lambda \overbrace{C(\mathbf{X})}^{< m}}}{|\mathcal{X}|^{-1} \left(\sum_{\mathbf{x} \in \mathcal{X}^*} e^{-\lambda m} + \sum_{\mathbf{x} \notin \mathcal{X}^*} e^{-\lambda \overbrace{C(\mathbf{X})}^{< m}} \right)}.
\end{aligned}$$

The right-hand side terms in the numerator and denominator are negligible with respect to their corresponding left-hand side ones. Therefore,

$$\begin{aligned}
g(\mathbf{x}) &= h(\mathbf{x}, \mathbf{u}) \lim_{\lambda \rightarrow -\infty} \frac{I_{\mathbf{w} \in \mathcal{X}^*} e^{-\lambda m}}{|\mathcal{X}|^{-1} \sum_{\mathbf{x} \in \mathcal{X}^*} e^{-\lambda m}} = \frac{h(\mathbf{x}, \mathbf{u}) I_{\mathbf{w} \in \mathcal{X}^*}}{|\mathcal{X}|^{-1} \sum_{\mathbf{x} \in \mathcal{X}^*} 1} \\
&= \frac{h(\mathbf{x}, \mathbf{u}) I_{\mathbf{w} \in \mathcal{X}^*}}{|\mathcal{X}|^{-1} \sum_{\mathbf{x} \in \mathcal{X}} I_{\{\mathbf{x} \in \mathcal{X}^*\}}} = \frac{h(\mathbf{x}, \mathbf{u}) I_{\{C(\mathbf{w})=m\}}}{\mathbb{E}_{\mathbf{u}}[I_{\{\mathbf{X} \in \mathcal{X}^*\}]}] \triangleq g^*(\mathbf{x}).
\end{aligned}$$

□

It is important to note that in contrast to the optimal pdf $g(\mathbf{x})$ in the IME program (37), the optimal pdf $g(\mathbf{x})$ in the classic MinxEnt program (4) is not zero-variance, (it leads only to variance reduction).

Observe again that generating samples from a multidimensional Boltzmann pdf, like $g(\mathbf{x})$ in (38) is a difficult task. The only available MCMC (Markov Chain Monte Carlo) algorithm [22] is very slow, in particular when the “temperature” parameter λ is low. Recently a dynamic programming approach has been introduced in [6] to sample efficiently from the Boltzmann pdfs. The method of [6] for efficient generation from the pdf (38) will be implemented some where else.

Similar to [17] we shall approximate $g(\mathbf{x})$ in (38) by the *product of its marginal pdf's* $g_i(x_i) = f_i(x_i, p_i)$, $i = 1, \dots, n$, that is we shall write the components p_i , $i = 1, \dots, n$ of the optimal vector \mathbf{p} as

$$p_i = \frac{\mathbb{E}_{\mathbf{u}} [X_i \exp \{-\lambda \sum_{i=1}^m C_i(\mathbf{X})\}]}{\mathbb{E}_{\mathbf{u}} [\exp \{-\sum_{i=1}^m \lambda C_i(\mathbf{X})\}]}, \quad i = 1, \dots, n, \quad (41)$$

which coincides with (26) up to the notations. Note that when each component of \mathbf{X} is an arbitrary r -point discrete random variable then (41) extends to

$$p_{ij} = \frac{\mathbb{E}_{\mathbf{u}} [I_{X_i=j} \exp \{-\lambda \sum_{i=1}^m C_i(\mathbf{X})\}]}{\mathbb{E}_{\mathbf{u}} [\exp \{-\sum_{i=1}^m \lambda C_i(\mathbf{X})\}]}, \quad i = 1, \dots, n; \quad j = 1, \dots, r. \quad (42)$$

Remark 3.2 (The Standard CE Method) Similar to (41) (see also (27)) we can define the following CE updating formula

$$p_j = \frac{\mathbb{E}_{\mathbf{u}} [X_j I_{\{\sum_{i=1}^m C_i(\mathbf{X})=m\}}]}{\mathbb{E}_{\mathbf{u}} [I_{\{\sum_{i=1}^m C_i(\mathbf{X})=m\}}]}. \quad (43)$$

In summary, to estimate efficiently ℓ in (34) and the associated counting quantity $|\mathcal{X}^*|$, we shall use again the IS estimator (22), where \mathbf{p} in $f(\mathbf{x}; \mathbf{p})$ is given in (41) and it is obtained from the solution of the IME program (37).

The diagram explaining the connection between the rare-events, IME and counting is similar to (30) and it can be presented as

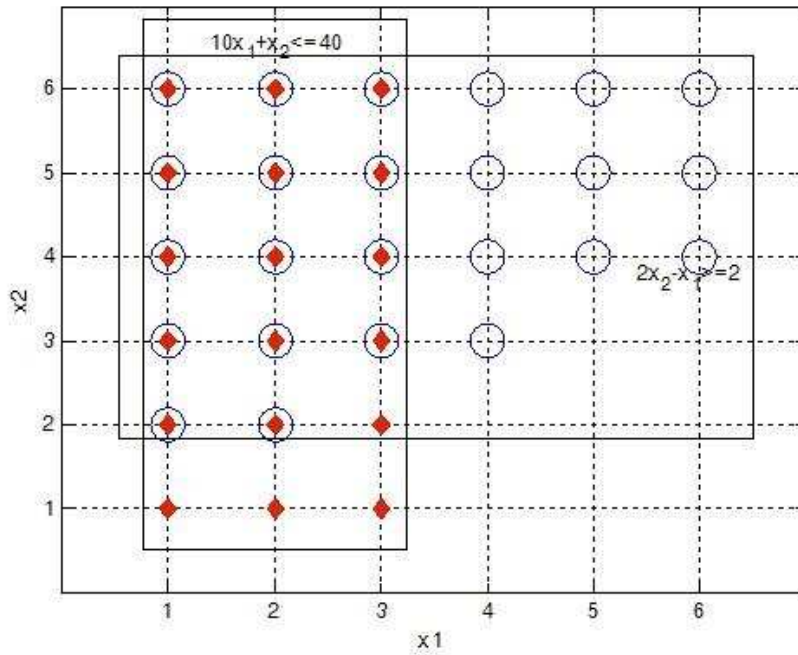
$$\{S_i(\mathbf{x}) \geq b_i, \quad i = 1, \dots, m\} \longrightarrow \mathbb{E}_{\mathbf{u}} \left[\prod_{i=1}^m I_{\{S_i(\mathbf{x}) \geq b_i\}} \right] \longrightarrow \text{IME (37)} \longrightarrow \text{Count via (24)}. \quad (44)$$

Example 3.1 (Counting With Two Symmetric Dice) Consider counting the number of feasible solutions of the following constraints system associated with rolling two symmetric dice

$$\begin{aligned} 10x_1 + x_2 &\leq 40 \\ 2x_2 - x_1 &\geq b \\ x_k &\in \{1, \dots, 6\}, k = 1, 2. \end{aligned} \tag{45}$$

Figure 1 depicts the feasible region defined by the two constraints (45) with $b = 2$.

Figure 1: The region (45) with $b = 2$



It is readily seen that the constraints (45) define 14 feasible points.

Table 2 presents $C_1(x_1, x_2) = I_{\{10x_1 + x_2 \leq 40\}}$ as function of x_1 and x_2 .

Table 2: $C_1(x_1, x_2) = I_{\{10x_1 + x_2 \leq 40\}}$ as function of x_1 and x_2

$x_1 \backslash x_2$	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0

Table 3 presents $C_2(x_1, x_2, b) = I_{\{2x_2 - x_1 \geq b\}}$ as function of x_1 and x_2 for different values of b .

Table 3: $C_2(x_1, x_2, b)$ as function of x_1 and x_2 for different values of b

$x_1 \backslash x_2$ $b = 1$	1	2	3	4	5	6	$x_1 \backslash x_2$ $b = 2$	1	2	3	4	5	6
1	1	1	1	1	1	1	1	0	1	1	1	1	1
2	0	1	1	1	1	1	2	0	1	1	1	1	1
3	0	1	1	1	1	1	3	0	0	1	1	1	1
4	0	0	1	1	1	1	4	0	0	1	1	1	1
5	0	0	1	1	1	1	5	0	0	0	1	1	1
6	0	0	0	1	1	1	6	0	0	0	1	1	1
$x_1 \backslash x_2$ $b = 3$	1	2	3	4	5	6	$x_1 \backslash x_2$ $b = 4$	1	2	3	4	5	6
1	0	1	1	1	1	1	1	0	0	1	1	1	1
2	0	0	1	1	1	1	2	0	0	1	1	1	1
3	0	0	1	1	1	1	3	0	0	0	1	1	1
4	0	0	0	1	1	1	4	0	0	0	1	1	1
5	0	0	0	1	1	1	5	0	0	0	0	1	1
6	0	0	0	0	1	1	6	0	0	0	0	1	1
$x_1 \backslash x_2$ $b = 5$	1	2	3	4	5	6	$x_1 \backslash x_2$ $b = 6$	1	2	3	4	5	6
1	0	0	1	1	1	1	1	0	0	0	1	1	1
2	0	0	0	1	1	1	2	0	0	0	1	1	1
3	0	0	0	1	1	1	3	0	0	0	0	1	1
4	0	0	0	0	1	1	4	0	0	0	0	1	1
5	0	0	0	0	1	1	5	0	0	0	0	0	1
6	0	0	0	0	0	1	6	0	0	0	0	0	1

By Lemma 3.1 $\lambda = -\infty$ for all values of b . Also, by Lemma 3.2 the optimal pdf $g(\mathbf{x})$ is uniformly distributed over the 14 points (see Figure 1) defined by the constraints (45).

Table 4 presents the optimal $g(x_1, x_2)$ as function of x_1 and x_2 .

Table 4: The optimal $g(x_1, x_2)$ as function of x_1 and x_2 for $b = 2$

$x_1 \backslash x_2$	1	2	3	4	5	6
1	0	1/14	1/14	1/14	1/14	1/14
2	0	1/14	1/14	1/14	1/14	1/14
3	0	0	1/14	1/14	1/14	1/14
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0

We next calculate $g_1(x_1)g_2(x_2)$. Before that we calculate the components p_{ij} , $i = 1, 2$; $j = 1, \dots, 6$ of the vector \mathbf{p} according to (42), which for our toy example reduces to

$$p_{ij} = \frac{\mathbb{E}_u [I_{X_i=j} \exp \{-\sum_{i=1}^2 \lambda C_i(X)\}]}{\mathbb{E}_u [\exp \{-\sum_{i=1}^2 \lambda C_i(X)\}]} \quad (46)$$

Table 5 presents the vector \mathbf{p} , $\mathcal{S}(\mathbf{p})$ and $|\mathcal{X}^*$ along with the estimator

$$|\widehat{\mathcal{X}^*}| = \frac{1}{N} \sum_{k=1}^N I_{\{C_1(X_k) + C_2(X_k) = 2\}} \frac{1}{g(X_k, \mathbf{p})} \quad (47)$$

for different values of values of b .

Table 5: \mathbf{p} , $\mathcal{S}(\mathbf{p})$, $|\mathcal{X}^*|$ and $|\widehat{\mathcal{X}^*}|$ for different values of b

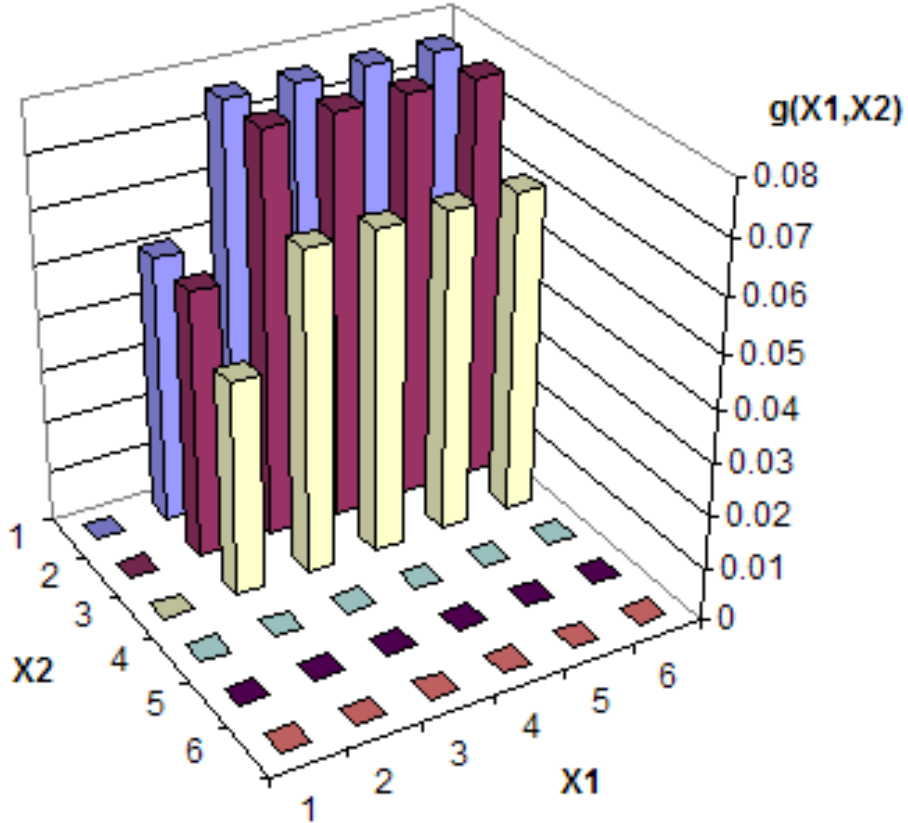
b	$ \mathcal{X}^* $	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$	$p_{1,5}$	$p_{1,6}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$	$p_{2,5}$	$p_{2,6}$	$\mathcal{S}(\mathbf{p})$	$ \widehat{\mathcal{X}^*} $
1	16	$\frac{6}{16}$	$\frac{5}{16}$	$\frac{5}{16}$	0	0	0	$\frac{1}{16}$	$\frac{3}{16}$	$\frac{3}{16}$	$\frac{3}{16}$	$\frac{3}{16}$	$\frac{3}{16}$	2.69	16.06
2	14	$\frac{3}{14}$	$\frac{4}{14}$	$\frac{4}{14}$	0	0	0	0	$\frac{1}{14}$	$\frac{3}{14}$	$\frac{3}{14}$	$\frac{3}{14}$	$\frac{3}{14}$	2.69	13.96
3	13	$\frac{3}{13}$	$\frac{4}{13}$	$\frac{4}{13}$	0	0	0	0	$\frac{1}{13}$	$\frac{3}{13}$	$\frac{3}{13}$	$\frac{3}{13}$	$\frac{3}{13}$	2.64	12.97
4	11	$\frac{4}{11}$	$\frac{4}{11}$	$\frac{3}{11}$	0	0	0	0	0	$\frac{2}{11}$	$\frac{3}{11}$	$\frac{3}{11}$	$\frac{3}{11}$	2.46	11.01
5	10	$\frac{4}{10}$	$\frac{3}{10}$	$\frac{3}{10}$	0	0	0	0	0	$\frac{2}{10}$	$\frac{3}{10}$	$\frac{3}{10}$	$\frac{3}{10}$	2.40	10.04
6	8	$\frac{5}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	0	0	0	0	0	$\frac{2}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	2.16	8.00

Based on the results of Table 5, Table 6 and Figure 2 presents the component-wise product of the marginal distributions of $g(\mathbf{x})$, that is, $g_1(x_1)g_2(x_2) = f_1(x_1, \mathbf{p}_1)f_2(x_2, \mathbf{p}_2)$ for $b = 2$.

Table 6: The IS pdf $g_1(x_1)g_2(x_2)$ as function of x_1 and x_2 for $b = 2$.

$\mathbf{x}_1 \backslash \mathbf{x}_2$	1	2	3	4	5	6
1	0	$10/14^2$	$15/14^2$	$15/14^2$	$15/14^2$	$15/14^2$
2	0	$10/14^2$	$15/14^2$	$15/14^2$	$15/14^2$	$15/14^2$
3	0	$8/14^2$	$12/14^2$	$12/14^2$	$12/14^2$	$12/14^2$
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0

Figure 2: The IS pdf $g_1(x_1)g_2(x_2)$ for $b = 2$.



It is readily seen that in contrast to the joint optimal uniform pdf $g(\mathbf{x})$ defined over the set of 14 points, the product of its associated marginal distributions $g_1(x_1)g_2(x_2)$ is defined over the set of 15 points, that is, it contains an extra point x_{32} , which is *outside the feasible region* (45) with probability $p_{32} > 0$. It follows from Table 5 that using the product of its optimal marginal distributions $g_1(x_1)g_2(x_2)$ instead of the optimal (zero variance) one $g(\mathbf{x})$ we still get a low variance estimator $|\widehat{\mathcal{X}^*}|$ of $|\mathcal{X}^*|$.

Example 3.2 Counting 0-1 Tables with Fixed Margins

The set $\mathbf{Ax} = \mathbf{b}$ is given as

$$\begin{aligned}\sum_{i=1}^n x_{ij} &= b_j^{(1)}, \quad j = 1, \dots, m \\ \sum_{j=1}^m x_{ij} &= b_i^{(2)}, \quad i = 1, \dots, n \\ X_{ij} &\in \{0, 1\}, \quad \forall i, j\end{aligned}\tag{48}$$

We shall approximate $|\mathcal{X}^*|$ using both the IME and SME approaches for $m = n = 3$ with different values of the the vector $(\mathbf{b}^{(1)}, \mathbf{b}^{(2)})$. For both IME and SME approaches we shall approximate $g(\mathbf{x})$ using the (i) product of its marginal pdfs and (ii) pair wise dependence.

The following table presents such a summary of the results.

Table 7: The vector \mathbf{p} for several values of the vector $(\mathbf{b}^{(1)}, \mathbf{b}^{(2)})$.

$\mathbf{b}^{(1)}$	$\mathbf{b}^{(2)}$	p_{11}	p_{12}	p_{13}	p_{21}	p_{22}	p_{23}	p_{31}	p_{32}	p_{33}	$\mathcal{S}(p)$	$ \widehat{\mathcal{X}^*} $
(0,0,0)	(0,0,0)	0	0	0	0	0	0	0	0	0	0	1
(0,0,1)	(1,0,0)	0	0	1	0	0	0	0	0	0	0	1
(0,2,2)	(2,2,0)	0	1	1	0	1	0	0	0	0	0	1
(1,1,2)	(2,1,1)	0.6	0.6	0.8	0.2	0.2	0.6	0.2	0.2	0.6	5.19	5.07
(1,1,3)	(2,2,1)	0.5	0.5	1	0.5	0.5	1	0	0	1	2.77	1.97
(2,2,2)	(3,1,2)	1	1	1	0.33	0.33	0.33	0.66	0.66	0.66	3.82	3.015
(3,3,3)	(3,3,3)	1	1	1	1	1	1	1	1	1	0	1

Note that for the extreme values of the vectors $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$, namely for $\mathbf{b}^{(1)} = \mathbf{b}^{(2)} = (0, 0, 0)$ and $\mathbf{b}^{(1)} = \mathbf{b}^{(2)} = (3, 3, 3)$, we obtain degenerated solutions (all components of \mathbf{p} are either zeros or all ones, respectively). In this case, we also that have $\mathcal{S}(p) = 0$ and $|\widehat{\mathcal{X}^*}| = 1$, as expected. Similar, for $\mathbf{b}^{(1)} = (1, 1, 3)$ and $\mathbf{b}^{(2)} = (2, 2, 1)$ we obtain two feasible solutions with the corresponding values of \mathbf{X} : $\mathbf{X}_1 = (1, 0, 0, 0, 1, 0, 1, 1, 1)$ and $\mathbf{X}_2 = (0, 1, 0, 1, 0, 0, 1, 1, 1)$, respectively. In this case the estimate of $|\mathcal{X}^*|$ (based on a sample $N = 1000$) is $|\widehat{\mathcal{X}^*}| = 1.9712$ and similarly for the other values of $(\mathbf{b}^{(1)}, \mathbf{b}^{(2)})$.

Consider finally the extreme case where m and n are even, $b_j^{(1)} = \frac{n}{2}, j = 1, \dots, m$, and $b_i^{(2)} = \frac{m}{2}, i = 1, \dots, n$. In this case, clearly, the optimal IME vector $\mathbf{p} = (1/2, \dots, 1/2)$, that is, it coincides with the initial one \mathbf{p}_0 and thus the IME based on $f(\mathbf{x}, \mathbf{p})$ is useless.

Theorem 3.1 For $\lambda = -\infty$ the optimal parameter vector \mathbf{p} in (41) of the marginal pdf's of the optimal $g(\mathbf{x})$ in (38) coincides with the \mathbf{p} in (43) for the CE method.

Proof The proof is very similar to Lemma 3.3 and is omitted. \square

Theorem 3.1 is crucial for the foundations of the CE method. Indeed, designed originally in [16] as a heuristics for rare-event estimation and COP's, Theorem 3.1 states that CE has strong connections with the IME program (37) and, thus, has strong mathematical foundation. The main reason is that the the optimal parametric pdf $f(\mathbf{x}, \mathbf{p}) = f(\mathbf{x}, \mathbf{p}, \lambda)$ (with \mathbf{p} in (41) and $\lambda = -\infty$) and the CE

pdf (with \mathbf{p} as in (43)) obtained heuristically from the solution of the following cross-entropy program

$$\min_{\mathbf{p}} \mathbb{E}_g \left[\ln \frac{g(\mathbf{X})}{f(\mathbf{X}, \mathbf{p})} \right]$$

are the same, provided $g(\mathbf{x})$ is the zero variance IS pdf, that is, (see [21])

$$g(\mathbf{x}) = \frac{h(\mathbf{x}, \mathbf{u}) I_{\{C(\mathbf{x})=m\}}}{\mathbb{E}_{\mathbf{u}} [I_{\{\mathbf{X} \in \mathcal{X}^*\} }]}.$$

The crucial difference between the proposed SME method and its CE counterparts lies in their simulation-based versions: in the latter we always require to generate a sequence of tuples $\{\mathbf{p}_t, m_t\}$, while in the former we can fix in advance the temperature parameter λ (to be set a large negative number) and then generate a sequence of parameter vectors $\{\mathbf{p}_t\}$ based on (41) alone. In addition, in contrast to CE, *neither the elite sample nor the rarity parameter are involved in SME*. As result, the proposed SME Algorithm becomes typically simpler, faster and at least as accurate as the standard CE based on (43).

Remark 3.3 It is not difficult to prove that Lemmas 3.1-3.3 still remain valid if we split the constraint $\sum_{i=1}^m C_i(\mathbf{X}) = m$ of the program (37) into several ones. In particular the two-constrained version of the program (37) can be written as

$$\begin{aligned} \min_g \mathcal{D}(g, h) &= \min_g \int \ln \frac{g(\mathbf{x})}{h(\mathbf{x})} g(\mathbf{x}) d\mathbf{x} = \min_g \mathbb{E}_g \left[\ln \frac{g(\mathbf{X})}{h(\mathbf{X})} \right] \\ \text{s.t. } \mathbb{E}_g [\sum_{i=1}^{m_1} C_i(\mathbf{X})] &= m_1 \\ \mathbb{E}_g [\sum_{j=1+m_1}^m C_j(\mathbf{X})] &= m_2 \\ \int g(\mathbf{x}) d\mathbf{x} &= 1, \end{aligned} \tag{49}$$

where $m = m_1 + m_2$. In this case it is easy to prove that both λ_1 and λ_2 of the optimal zero variance pdf

$$g(\mathbf{x}) = \frac{h(\mathbf{x}, \mathbf{u}) \exp \left\{ -\lambda_1 \sum_{i=1}^{m_1} C_i(\mathbf{x}) - \lambda_2 \sum_{j=m_1+1}^m C_j(\mathbf{x}) \right\}}{\mathbb{E}_{\mathbf{u}} \left[\exp \left\{ -\lambda_1 \sum_{i=1}^{m_1} C_i(\mathbf{X}) - \lambda_2 \sum_{j=m_1+1}^m C_j(\mathbf{X}) \right\} \right]} \tag{50}$$

are equal to $-\infty$ and similarly for an arbitrary splitting of the constraint $\sum_{i=1}^m C_i(\mathbf{X}) = m$ into k , $k = 1, \dots, m$ groups.

4 The Semi-Iterative Minx-Ent (SME) Counting Algorithm

Here we present the semi-iterative Minx-Ent (SME) counting algorithm for rare-events and counting the number of feasible solutions on the set \mathcal{X}^* defined by

$$\begin{aligned} S_i(\mathbf{x}) &= b_i, \quad i = 1, \dots, m_1, \\ S_j(\mathbf{x}) &\leq b_j, \quad j = m_1 + 1, \dots, m. \end{aligned} \tag{51}$$

We call our method, the semi-iterative MinxEnt (SME), to distinguish it from the iterative cross-entropy (CE) for the following reasons. As we shall see below

1. CE

- Generates iteratively a sequence of tuples $\{\hat{\mathbf{p}}_t, \hat{m}_t\}$, where $\hat{\mathbf{p}}_t$ and \hat{m}_t , denote the estimates of the optimal parameter vector in the parametric pdf $f(\mathbf{x}, \mathbf{p})$ and the approximation of m at the t -th iteration, respectively.
- Involves a *rarity parameter* ρ and *elite sampling*, while generating the sequence $\{\hat{\mathbf{p}}_t, \hat{m}_t\}$.

2. SME

- Generates only a sequence of vectors $\{\mathbf{p}_t\}$, while fixing the temperature parameter λ in advance (to be a large negative number).
- Neither rarity parameter ρ , no elite samples are involved in SME.

For this reasons SME is much simpler than CE and MinxEnt and, as we shall see below from our numerical results, it is faster and at least as accurate as its counterparts CE and MinxEnt.

It is also important to keep in mind that the SME method is based on the MinxEnt program (37), which combines the multiple (deterministic) constraints (51) into a single (stochastic) one given as $\mathbb{E}_g[\sum_{i=1}^m C_i(\mathbf{X})] = m$.

If not stated otherwise we shall assume that $g(\mathbf{x})$ is approximated by the product of its marginal pdf's $g_i(x_i) = f_i(x_i, p_i)$, $i = 1, \dots, n$, where the components p_i , $i = 1, \dots, n$ of the optimal vector \mathbf{p} are given in (41). As soon as an estimate of \mathbf{p} is derived we estimate ℓ and $|\mathcal{X}^*|$ using the IS estimator (22) and (24), respectively.

Before proceeding with the SME algorithm we introduce in Section 4.1 the so-called iterative IME algorithms to distinguish it from our main SME Algorithm for counting. Note that in the iterative IME we are purposely ignoring the fact that $\lambda = -\infty$ and, thus we generate a sequence of triplets $\{\hat{\mathbf{p}}_t, \hat{m}_t, \hat{\lambda}_t\}$. Here we also show the similarity of the IME algorithm and the standard CE one. In section 4.2 we introduce the SME algorithm, where, as mentioned, we take into consideration the fact that $\lambda = -\infty$. By doing so, SME generates a sequence of vectors $\{\hat{\mathbf{p}}_t\}$ instead of the sequence of triplets $\{\hat{\mathbf{p}}_t, \hat{m}_t, \hat{\lambda}_t\}$, and is thus faster.

4.1 Standard CE and the Iterative IME

In the standard CE and the iterative IME approaches one uses a *multi-level* approach, that is, one generates simultaneously a sequence of the parameter vector \mathbf{p}_t of the parametric pdf's $f(\mathbf{x}, \mathbf{p}_t)$ and levels $\{m_t\}$. Starting with $f(\mathbf{x}, \mathbf{p}_0) = h(\mathbf{x}, \mathbf{u})$, that is, taking the prior $h(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}, \mathbf{p}_0)$, one

1. Updates m_t as

$$m_t = \mathbb{E}_{g_{t-1}}[\mathcal{C}(\mathbf{X}) | \mathcal{C}(\mathbf{X}) \geq q_t],$$

where q_t is the $(1 - \rho)$ -quantile of $\mathcal{C}(\mathbf{X})$ under g_t and as before $\mathcal{C}(\mathbf{X}) = \sum_{i=1}^m C_i(\mathbf{X})$.

2. Updates g_t as the solution to the above MinxEnt program for level m_t , rather than m .

The updating formula for m_t is based on the constraint $\mathbb{E}_g[\mathcal{C}(\mathbf{X})] = m$ in the MinxEnt program. However, instead of simply updating as $m_t = \mathbb{E}_{g_{t-1}}[\mathcal{C}(\mathbf{X})]$, we take the expectation of $\mathcal{C}(\mathbf{X})$ with respect to g_{t-1} *conditional* upon $\mathcal{C}(\mathbf{X})$ being greater than its $(1 - \rho)$ quantile, here denoted as q_t . In contrast, in the standard CE method the level m_t is simply updated as q_t .

Note that in IME each g_t is completely determined by its multiplier, say λ_t , which is the solution to (39) with m_t instead of m . In practice both m_t and λ_t have to be replaced with their respective stochastic versions \hat{m}_t and $\hat{\lambda}_t$, respectively.

Specifically, m_t can be estimated from a random sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ of g_{t-1} as the average of the $N_e = \lceil \rho N \rceil$ elite sample performances:

$$\hat{m}_t = \frac{\sum_{i=N-N_e+1}^N \mathcal{C}^{(i)}}{N_e}, \quad (52)$$

where $\mathcal{C}^{(i)}$ denotes the i -th order-statistics of the sequence $\mathcal{C}(\mathbf{X}_1), \dots, \mathcal{C}(\mathbf{X}_N)$. The standard way of updating the parameter λ_t via simulation is to solve with respect to λ , the stochastic counterpart of (39), that is

$$\frac{\sum_{k=1}^N \mathcal{C}(\mathbf{X}_k) \exp\{-\hat{\lambda}_t \mathcal{C}(\mathbf{X}_k)\} W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1})}{\sum_{k=1}^N \exp\{-\hat{\lambda}_t \mathcal{C}(\mathbf{X}_k)\} W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1})} = \hat{m}_t, \quad (53)$$

where

$$W(\mathbf{x}) = \frac{f(\mathbf{x}, \mathbf{u})}{f(\mathbf{x}, \hat{\mathbf{p}}_{t-1})}$$

is the likelihood ratio.

Similarly, the component of the the vector \mathbf{p}_t are updated as

$$\hat{p}_{t,j} = \frac{\sum_{k=1}^N X_{k,j} \exp\{-\hat{\lambda}_t \mathcal{C}(\mathbf{X}_k)\} W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1})}{\sum_{k=1}^N \exp\{-\hat{\lambda}_t \mathcal{C}(\mathbf{X}_k)\} W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1})}. \quad (54)$$

In CE the updating of \mathbf{p} is performed in analogy to (54) according to

$$\hat{p}_{t,j} = \frac{\sum_{k=1}^N X_{k,j} I_{\{\mathcal{C}(\mathbf{X}_k) \geq \hat{m}_t\}} W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1})}{\sum_{k=1}^N I_{\{\mathcal{C}(\mathbf{X}_k) \geq \hat{m}_t\}} W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1})}. \quad (55)$$

Note that since the prior pdf $f(\mathbf{x}, \mathbf{u})$ is uniform it is desirable to replace for computational convenience $W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1})$ by $\frac{1}{f(\mathbf{X}_k; \hat{\mathbf{p}}_{t-1})}$.

Below we present the iterative IME algorithm for counting based on the sequence of triplets $\{\hat{\mathbf{p}}_t, \hat{m}_t, \hat{\lambda}_t\}$ and the standard CE one based on the sequence of tuples $\{\hat{\mathbf{p}}_t, \hat{m}_t\}$. As mentioned, in Section 4.2 we present our main SME Algorithm for counting by generating a sequence of vectors $\{\hat{\mathbf{p}}_t\}$ alone instead of the sequence of triplets $\{\hat{\mathbf{p}}_t, \hat{m}_t, \hat{\lambda}_t\}$.

Algorithm 4.1 (Iterative IME Algorithm for Counting)

1. Define $\hat{\mathbf{p}}_0 = \mathbf{u}$. Set $t = 0$ (iteration = level counter).
2. $t \leftarrow t + 1$. Generate a sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from the density $f(\mathbf{x}; \hat{\mathbf{p}}_{t-1})$ and compute \hat{m}_t and $\hat{\lambda}_t$ according to (52) and (53).
3. Use the **same** sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ and update \mathbf{p}_t according to (54). Denote the solution by $\hat{\mathbf{p}}_t$.

4. Smooth out the vector $\widehat{\mathbf{p}}_t$ according to

$$\bar{\mathbf{p}}_t = \alpha \widehat{\mathbf{p}}_t + (1 - \alpha) \widehat{\mathbf{p}}_{t-1}, \quad (56)$$

where α , ($0 < \alpha < 1$) is called the smoothing parameter.

5. If $\widehat{m}_t < m$ reiterate from step 2. Else proceed with step 6.

6. Reiterate steps 2.-4. for 3-4 more iterations. Estimate the counting quantity $|\mathcal{X}^*|$ as

$$|\widehat{\mathcal{X}^*}| = \frac{1}{N} \sum_{k=1}^N I_{\{\mathcal{C}(\mathbf{X}_k)=m\}} \frac{1}{f(\mathbf{X}_k; \widehat{\mathbf{p}}_t)}. \quad (57)$$

Algorithm 4.2 (Standard CE Algorithm for Counting)

1. Define $\widehat{\mathbf{p}}_0 = \mathbf{u}$. Set $t = 0$ (iteration = level counter).
2. $t \leftarrow t + 1$. Generate a sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from the density $f(\mathbf{x}; \widehat{\mathbf{p}}_{t-1})$ and compute \widehat{m}_t according to (52).
3. Use the **same** sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ and update \mathbf{p}_t according to (55). Denote the solution by $\widehat{\mathbf{p}}_t$.
4. Smooth out the vector $\widehat{\mathbf{p}}_t$ according to (56).
5. If $\widehat{m}_t < m$ reiterate from step 2. Else proceed with step 6.
6. Reiterate steps 2.-4. for 3-4 more iterations. Estimate the counting quantity $|\mathcal{X}^*|$ according to (57).

4.2 The Main SME Counting Algorithm

Since in SME λ is fixed (λ is a large negative number), the components of \mathbf{p} can be updated in analogy to (54) according to the following formula

$$\widehat{p}_{t,j} = \frac{\sum_{k=1}^N X_{kj} \exp\{-\lambda \mathcal{C}(\mathbf{X}_k)\} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})}{\sum_{k=1}^N \exp\{-\lambda \mathcal{C}(\mathbf{X}_k)\} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})}. \quad (58)$$

For application purposes we not only set λ to a large negative number, like $\lambda = -100$, but we also use in (62) instead of $\mathcal{C}(\mathbf{X}_k)$, its so-called normalized value

$$\mathcal{C}^{(n)}(\mathbf{X}_k) = \frac{\mathcal{C}(\mathbf{X}_k)}{\max_{k=1, \dots, N} \mathcal{C}(\mathbf{X}_k)}. \quad (59)$$

Using (59) the resulting updating of $\widehat{\mathbf{p}}_t$ can be written as

$$\widehat{p}_{t,j} = \frac{\sum_{k=1}^N X_{kj} \exp\{-\lambda \mathcal{C}^{(n)}(\mathbf{X}_k)\} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})}{\sum_{k=1}^N \exp\{-\lambda \mathcal{C}^{(n)}(\mathbf{X}_k)\} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})}, \quad (60)$$

The main reason for using $\mathcal{C}^{(n)}(\mathbf{X})$ instead of $\mathcal{C}(\mathbf{X})$ is for convenience only; to make sure that $\lambda \mathcal{C}^{(n)}(\mathbf{X}_k)$ is a large negative number, say $\lambda \mathcal{C}^{(n)}(\mathbf{X}_k) = -100$, when $\mathcal{C}^{(n)}(\mathbf{X}_k) = 1$.

Algorithm 4.3 (SME Algorithm for Counting)

1. Define $\widehat{\mathbf{p}}_0 = \mathbf{u}$. Set $\lambda = M$, say $M = -100$. Set $t = 0$ (iteration = level counter).
2. $t \leftarrow t + 1$. Generate a sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from the density $f(\mathbf{x}; \widehat{\mathbf{p}}_{t-1})$ and compute $\widehat{\mathbf{p}}_t$ according to (60).
3. Smooth out the vector $\widehat{\mathbf{p}}_t$ according to (56).
4. If $\mathcal{C}(\mathbf{X}) < m$, reiterate from step 2. Else proceed with step 5.
5. Reiterate steps 2.-3. for 2-4 more iterations. Estimate the counting quantity $|\mathcal{X}^*|$ as

$$|\widehat{\mathcal{X}^*}| = \frac{1}{N} \sum_{k=1}^N I_{\{\mathcal{C}(\mathbf{X}_k)=m\}} \frac{1}{f(\mathbf{X}_k; \widehat{\mathbf{p}}_t)}. \quad (61)$$

Our numerical results of Section 9 clearly show that the SME Algorithm 4.3 is quite robust with respect to λ , provided λ is a large negative number, say $-50 \geq \lambda \geq -1000$. To see this let $\lambda = -100$ and assume for simplicity that $\mathcal{C}^{(n)}(\mathbf{X}_k)$ obtains values from the set $\{1, 0.9, \dots, 0.1\}$. In this case, the updating of the parameter vector \mathbf{p} according to (60) will be based on the following *exponential* sequence $\{\exp(100), \exp(90), \dots, \exp(10)\}$. Clearly, the dominating term is $\exp(100)$, while the remaining ones are negligible. Similar conclusions hold for some other large negative values of λ , like $-50 \geq \lambda \geq -1000$.

Remark 4.1 (Convergence of Algorithm 4.3). Since for fixed λ Algorithm 4.3 updates only the single parameter vector $\widehat{\mathbf{p}}$, the convergence and the speed of the convergence of $\widehat{\mathbf{p}}$ to the true optimal parameter vector \mathbf{p}^* with the components

$$p_j = \frac{\mathbb{E}_{\mathbf{u}} X_j \exp\{-\lambda \mathcal{C}^{(n)}(\mathbf{X})\}}{\mathbb{E}_{\mathbf{u}} \exp\{-\lambda \mathcal{C}^{(n)}(\mathbf{X})\}} \quad (62)$$

follows from Theorems A1 and A2 of [20].

Remark 4.2 (The Method of Moments). Since $h(\mathbf{x}, \mathbf{u})$ is a uniform pdf, as an alternative to the original Shannon's entropy maximization program (4), (5) one can consider the following one

$$\begin{aligned} \min_g \int \{S(\mathbf{x}) - \mathbb{E}_g[S(\mathbf{X})]\}^2 g(\mathbf{x}) d\mathbf{x} &= \min_g \text{Var}_g[S(\mathbf{X})]. \\ \text{s.t. } \int S(\mathbf{x})g(\mathbf{x}) d\mathbf{x} &= \mathbb{E}_g[S(\mathbf{X})] = b, \\ \int g(\mathbf{x}) d\mathbf{x} &= 1. \end{aligned} \quad (63)$$

The nonparametric problem (63) is known as the *problem of moments* [23] and it can be also written as

$$\begin{aligned} \min_g \mathbb{E}_g[S^2(\mathbf{X})]. \\ \text{s.t. } \mathbb{E}_g[S(\mathbf{X})] &= b, \\ \int g(\mathbf{x}) d\mathbf{x} &= 1. \end{aligned} \quad (64)$$

This is because $\text{Var}_g[S(\mathbf{X})] = \mathbb{E}_g[S^2(\mathbf{X})] - (\mathbb{E}_g[S(\mathbf{X})])^2$ and $\mathbb{E}_g[S(\mathbf{X})] = b$.

It is also known that the minimum of the program (64) is attained at a probability distribution supported on a set of two points [23]. That is, the above problem is equivalent to

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{y} \in M, p \in [0,1]} S^2(\mathbf{x})p + S^2(\mathbf{y})(1-p) \\ \text{s.t. } S(\mathbf{x})p + S(\mathbf{y})(1-p) = b, \end{aligned} \quad (65)$$

where M is the support of the distribution of \mathbf{X} . The dual of the above problem can be written as a linear semi-infinite programming problem.

Note that from equation $S(\mathbf{x})p + S(\mathbf{y})(1-p) = b$ we have that $p = (b - S(\mathbf{y})) / (S(\mathbf{x}) - S(\mathbf{y}))$, provided that this solution is in $[0, 1]$. If it is not in $[0, 1]$, then the point (\mathbf{x}, \mathbf{y}) is infeasible. Substituting this into (65) we obtain the following problem

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{y} \in M} [S(\mathbf{x}) + S(\mathbf{y})]b - S(\mathbf{x})S(\mathbf{y}) \\ \text{s.t. } 0 \leq (b - S(\mathbf{y})) / (S(\mathbf{x}) - S(\mathbf{y})) \leq 1. \end{aligned} \quad (66)$$

This problem should be typically solved numerically. Moreover, the theory states [23] that if the (original) problem has a solution, then it is attained at an atomic measure. Such atomic measure does not have density. It may happen, of course, that the problem is infeasible, that is, has no solution.

4.3 Extensions

Note that depending on whether the constraint $\mathbb{E}_g[\sum_{i=1}^m C_i(\mathbf{X})] = m$ is satisfied or not (see (37)) the SME Algorithm (4.3) always puts a corresponding weight 1 or 0 regardless of how far $S_i(\mathbf{x})$ is from b_i , $i = 1, \dots, m$. To provide more flexibility to IME we can use a more sensitive “distance” between $S_i(\mathbf{x})$ and b_i .

In particular, if the set \mathcal{X}^* is given by

1. The system of equalities

$$\mathcal{X}^* = \{\mathbf{x} \in \mathbb{R}^n : S_i(\mathbf{x}) = b_i, i = 1, \dots, m\}, \quad (67)$$

one can consider instead of $\mathbb{E}_g[\mathcal{C}(\mathbf{X})] = \mathbb{E}_g[\sum_{i=1}^m C_i(\mathbf{X})] = m$ the following constraint

$$\mathbb{E}_g(\mathcal{C}(\mathbf{X})) = \mathbb{E}_g\left(\sum_{i=1}^m C_i(\mathbf{X})\right) = \mathbb{E}_g\left(\sum_{i=1}^m |S_i(\mathbf{X}) - b_i|^r\right) = 0, \quad (68)$$

where $r \geq 1$ say $r = 2$.

2. The system of inequalities

$$\mathcal{X}^* = \{\mathbf{x} \in \mathbb{R}^n : \sum_{k=1}^n a_{ik}x_k \leq b_i, i = 1, \dots, m\}, \quad (69)$$

one can consider the following constraint

$$\mathbb{E}_g(\mathcal{C}(\mathbf{X})) = \mathbb{E}_g\left(\sum_{i=1}^m C_i(\mathbf{X})\right) = \mathbb{E}_g\left(\sum_{i=1}^m [\max\{0, [S_i(\mathbf{X}) - b_i]^r\}]\right) = 0, \quad (70)$$

where r is odd, say, $r = 1, 3, \dots$

Clearly, if one has both equality and inequality constraints, (that is, as defined in (51)), then one has to combine both (68) and (70).

Note also that since $g(\mathbf{x})$ is not available in (68) and (70), in practice we check the fulfilment of the sample versions of $\mathbb{E}_{\mathbf{p}}\{\mathcal{C}(\mathbf{X})\} = 0$ rather than fulfilment of $\mathbb{E}_g[\mathcal{C}(\mathbf{X})] = 0$. Here as usual \mathbf{p} means that the expectation is taken under $f(\mathbf{x}, \mathbf{p})$ and the sample versions of $\mathbb{E}_{\mathbf{p}}[\mathcal{C}(\mathbf{X})] = 0$ can be written as

$$\bar{\mathcal{C}} = \frac{1}{N} \sum_{k=1}^N \mathcal{C}(\mathbf{X}_k) = 0. \quad (71)$$

We shall call the MinxEnt program (37) with the single constraint $\mathbb{E}_g[\sum_{i=1}^m C_i(\mathbf{X})] = m$ replaced by the combined version of (68) and (70), the *weighted MinxEnt* (WME), program to distinguish it from the original IME program (37). In short, the WME program coincides with the IME one (37), provided its constraint involving indicator functions are replaced by the ones involving the weight functions based on (68) and (70). Also, since the IME program can be viewed as particular case of the WME one, if not otherwise stated we shall use the WME one. It follows from the above that while using the WME program, the steps 3. and 4. of Algorithm 4.3 should be modified respectively as

1. If $\bar{\mathcal{C}}(\mathbf{X}) > 0$, reiterate from step 2. Else proceed with step 5.
2. Reiterate steps 2.-3. for 2-4 more iterations. Estimate the counting quantity $|\mathcal{X}^*|$ as

$$|\widehat{\mathcal{X}^*}| = \frac{1}{N} \sum_{k=1}^N I_{\{\bar{\mathcal{C}}(\mathbf{X}_k)=0\}} \frac{1}{f(\mathbf{X}_k; \widehat{\mathbf{p}}_t)}. \quad (72)$$

In summary, while referring below to Algorithm 4.3, we shall always mean using $\mathcal{C}(\mathbf{X})$ based on (68) and (70), rather than $\mathcal{C}(\mathbf{X})$ based on in the indicator, that is, defined as $\mathcal{C}(\mathbf{X}) = \sum_{i=1}^m C_i(\mathbf{X})$. Note again that we use in Algorithm 4.3 the normalized function $\mathcal{C}(\mathbf{X})$ (59), that is,

$$\mathcal{C}^{(n)}(\mathbf{X}_k) = \frac{\mathcal{C}(\mathbf{X}_k)}{\max_{k=1, \dots, N} \mathcal{C}(\mathbf{X}_k)},$$

rather than $\mathcal{C}(\mathbf{X})$ itself.

We finally remark that one can readily prove that Lemmas 3.1-3.3 hold for the WME program as well. For example, Lemma 3.3 for $\mathcal{C}(\mathbf{X}) = \sum_{i=1}^m [\max\{0, [S_i(\mathbf{X}) - b_i]^r\}]$ (see (70)) reads as

Lemma 4.1 *For $\lambda = -\infty$ the optimal WME pdf $g(\mathbf{x})$ in (38) coincides with the IS zero variance pdf*

$$g^*(\mathbf{x}) = \frac{h(\mathbf{x}, \mathbf{u}) I_{\{\sum_{i=1}^m [\max\{0, [S_i(\mathbf{X}) - b_i]^r\}] = 0\}}}{\mathbb{E}_{\mathbf{u}} [I_{\{\mathbf{X} \in \mathcal{X}^*\}]}}.$$

4.4 Introducing Dependence Between the Components of \mathbf{X}

In some applications the above counting algorithms based on the product of the marginals of $g(\mathbf{x})$, that is, on $f(\mathbf{x}, \mathbf{p})$, might have poor performance. To overcome this difficulty one can introduce dependence between the components of the random vector \mathbf{X} . In particular, one can find the associated k -dimensional marginal pdfs $g_{i_1, \dots, i_k}(x_{i_1}, \dots, x_{i_k})$, $k \leq n$ and the corresponding conditional ones

$$g_{i_1, \dots, i_k}(x_{i_k} | x_{i_1}, \dots, x_{i_{k-1}}) = \frac{g_{i_1, \dots, i_k}(x_{i_1}, \dots, x_{i_k})}{g_{i_1, \dots, i_{k-1}}(x_{i_1}, \dots, x_{i_{k-1}})}, \quad k = 1, \dots, n$$

from the optimal n -dimensional joint pdf $g(\mathbf{x})$. For example, having the two-dimensional marginals $g(x_{i_1}, x_{i_2})$ one can use instead of (41) the following MinxEnt updating

$$p_{ij} = \frac{\mathbb{E}_{\mathbf{u}} [X_i X_j \exp \{-\lambda \sum_{i=1}^m C_i(\mathbf{X})\}]}{\mathbb{E}_{\mathbf{u}} [\exp \{-\sum_{i=1}^m \lambda C_i(\mathbf{X})\}]}, \quad i, j = 1, \dots, n \quad (i \neq j). \quad (73)$$

In short, using $g_{i_1, \dots, i_k}(x_{i_1}, \dots, x_{i_k})$, $2 \leq k \leq n$ one might obtain alternative and more accurate IS estimators of ℓ and $|\mathcal{X}^*|$ given in (22) and (24).

Note also that by analogy to (73) we can define the pair-wise updating in ICE as

$$p_{ij} = \frac{\mathbb{E}_{\mathbf{u}} [X_i X_j I_{\{\sum_{i=1}^m C_i(\mathbf{X})=m\}}]}{\mathbb{E}_{\mathbf{u}} [I_{\{\sum_{i=1}^m C_i(\mathbf{X})=m\}}]}}, \quad i, j = 1, \dots, n \quad (i \neq j). \quad (74)$$

If not stated otherwise we shall consider in this paper with the following truncated version

$$h_{tr}^{(k)}(\mathbf{x}) = g_{i_1}(x_{i_1})g_{i_2|i_1}(x_{i_2}|x_{i_1}) \cdots g_{i_j|i_1, \dots, i_{j-1}}(x_{i_j}|x_{i_1}, \dots, x_{i_{j-1}}) \cdots \\ g_{i_n|i_{n-k+1}, \dots, i_{n-1}}(x_{i_n}|x_{i_{n-k+1}}, \dots, x_{i_{n-1}}) \quad (75)$$

of the true optimal pdf

$$g(\mathbf{x}) \equiv h^{(n)}(\mathbf{x}) = g_{i_1}(x_{i_1})g_{i_2|i_1}(x_{i_2}|x_{i_1}) \cdots g_{i_n|i_1, \dots, i_{n-1}}(x_{i_n}|x_{i_1}, \dots, x_{i_{n-1}}). \quad (76)$$

Note that for $k = 2$ formula (75) reduces to the following ‘‘Markovian’’ type pdf

$$h_{tr}^{(2)}(\mathbf{x}) = g_{i_1}(x_{i_1})g_{i_2|i_1}(x_{i_2}|x_{i_1}) \cdots g_{i_n|i_{n-1}}(x_{i_n}|x_{i_{n-1}}), \quad (77)$$

which will be implemented in our counting algorithms in parallel to the main case, which is based on the independence of marginal pdfs $h_{tr}^{(1)}(\mathbf{x}) = f(\mathbf{x}, \mathbf{p})$ of $g(\mathbf{x})$.

5 SME for Counting the Number of Feasible Solutions in an Integer Program

An integer program with both equality and inequality constraints reads as

$$\begin{aligned} & \min \mathbf{c}'\mathbf{x}, \\ & \text{s.t.} \quad \sum_{k=1}^n a_{ik}x_k = b_i, \quad i = 1, \dots, m_1, \\ & \quad \quad \sum_{k=1}^n a_{jk}x_k \geq b_j, \quad j = m_1 + 1, \dots, m_1 + m_2, \\ & \quad \quad \mathbf{x} \geq \mathbf{0}, \quad x_k \text{ integer } \forall k = 1, \dots, n. \end{aligned} \quad (78)$$

where \mathbf{c} and \mathbf{x} are n -dimensional vector.

Here we shall present a slightly modified version of our generic counting Algorithm 4.3, which can count efficiently the number of feasible solutions on the set containing both equality and inequality constraints defined in (78), that is, on the set

$$\begin{aligned} & \sum_{k=1}^n a_{ik}x_k = b_i, \quad i = 1, \dots, m_1, \\ & \sum_{k=1}^n a_{jk}x_k \geq b_j, \quad j = m_1 + 1, \dots, m_1 + m_2, \\ & \mathbf{x} \geq \mathbf{0}, \quad x_k \text{ integer } \forall k = 1, \dots, n. \end{aligned} \quad (79)$$

Our modification of Algorithm 4.3 takes into account the additivity properties of the functions $S_i(\mathbf{x}) = \sum_{k=1}^n a_{ik}x_k$ in (79).

To proceed with the modification we need to modify the m terms $C_i(\mathbf{X})$ in (36), that is,

$$\mathbb{E}_g \left\{ \sum_{i=1}^m C_i(\mathbf{X}) \right\} = m$$

as follows. Here the first m_1 terms $C_i(\mathbf{X})$'s (out of the total of $m = m_1 + m_2$ terms) are defined as

$$C_i(\mathbf{X}) = I_{\{\sum_{k=1}^n a_{ik}X_k = b_i\}}, \quad i = 1, \dots, m_1, \quad (80)$$

while the remaining m_2 ones are defined as

$$C_i(\mathbf{X}) = I_{\{\sum_{k=1}^n a_{ik}X_k \geq b_i\}}, \quad i = m_1 + 1, \dots, m_1 + m_2. \quad (81)$$

Since for integer problems $S_i(\mathbf{X})$ equals to $S_i(\mathbf{X}) = \sum_{k=1}^n a_{ik}X_k$ we can apply here the results of the previous section. In particular,

- We associate a separate indicator $C_i(\mathbf{X})$ in the MinxEnt program (37) to each constraint in (78).
- In order to count the number $|\mathcal{X}^*|$ of feasible solutions of the program (78), that is, on the set (79) we associate with it the following rare-event probability

$$\ell = \mathbb{P}_{\mathbf{u}}\{\mathbf{X} \in \mathcal{X}^*\} = \mathbb{E}_{\mathbf{u}} \left[\prod_{i=1}^{m_1} I_{(\sum_{k=1}^n a_{ik}X_k = b_i)} \prod_{j=m_1+1}^{m_1+m_2} I_{(\sum_{k=1}^n a_{jk}X_k \geq b_j)} \right]. \quad (82)$$

- To estimate ℓ we apply the IS formula (24), where the optimal parameter vector \mathbf{p} is updated according to the SME method given in (59) or according to the ICE method given in (43). We shall use \mathbf{p} derived from SME. Recall again that we use in Algorithm 4.3 $\mathcal{C}(\mathbf{X})$ based on (68) and (70) rather than the one based on the indicator defined as $\mathcal{C}(\mathbf{X}) = \sum_{i=1}^m C_i(\mathbf{X})$, since we found that Algorithm 4.3 based on the former (weighted) function $\mathcal{C}(\mathbf{X})$ produces more accurate results than on the latter indicator function.

Below we present calculations for the weighted function $\mathcal{C}(\mathbf{X})$ in case of linear constraints, that is, assuming that $S_i(\mathbf{x}) = \sum_{k=1}^n a_{ik}x_k$. In this case formulas (68) and (70) reduce to

$$\mathbb{E}_g(\mathcal{C}(\mathbf{X})) = \mathbb{E}_g \left(\sum_{i=1}^m C_i(\mathbf{X}) \right) = \mathbb{E}_g \left(\sum_{i=1}^m \left| \sum_{k=1}^n a_{ik}X_k - b_i \right|^r \right) = 0, \quad (83)$$

and

$$\mathbb{E}_g(\mathcal{C}(\mathbf{X})) = \mathbb{E}_g \left(\sum_{i=1}^m C_i(\mathbf{X}) \right) = \mathbb{E}_g \left(\sum_{i=1}^m \left[\max \left\{ 0, \left[b_i - \sum_{k=1}^n a_{ik}X_k \right]^r \right\} \right] \right) = 0, \quad (84)$$

respectively. Note that in (83) and (84) r should be even and odd, respectively.

Also, when the a_{ik} 's are different from 0's and 1's it is advisable to use instead of

$$\left[\sum_{k=1}^n a_{ik}X_k - b_i \right]^r, \quad i = 1, \dots, m$$

the following normalized value

$$\left[\sum_{k=1}^n \alpha_{ik} X_k - \beta_i \right]^r, \quad i = 1, \dots, m, \quad (85)$$

where $\alpha_{ik} = \frac{a_{ik}}{\max_k |a_{ik}|}$ and $\beta_i = \frac{b_i}{\max_k |a_{ik}|}$.

At this end note that for large n the estimators (53) and (54) are typically unstable because of the likelihood ratio term W . To overcome this difficulty we shall present alternative ones based on the notion of depth- r updating of \mathbf{p} for block-separable function [19].

5.1 Depth- r updating for block-separable function using Minx-Ent

We start with the following

Example 5.1 Consider a block-separable function $\mathcal{C}(\mathbf{x})$ of the form

$$\mathcal{C}(\mathbf{x}) = \sum_{j=1}^{n-1} C_j(x_j, x_{j+1}),$$

where as before all $C_j(x_j, x_{j+1})$'s are indicator functions.

Suppose we want to find p_2 using the deterministic IME. Consider first the case where $C_3 \equiv 0$. Define $J = \{1, 2, 3\}$ and $\bar{J} = \{4, \dots, n\}$. Let us denote by \mathbf{x}_J the vector with components $\{x_j, j \in J\}$, and similar for $\mathbf{x}_{\bar{J}}$. We can now write $\mathcal{C}(\mathbf{x})$ as

$$\mathcal{C}(\mathbf{x}) = \mathcal{C}_J(\mathbf{x}_J) + \mathcal{C}_{\bar{J}}(\mathbf{x}_{\bar{J}}), \quad (86)$$

with $\mathcal{C}_J(\mathbf{x}_J) = C_1(x_1, x_2) + C_2(x_2, x_3)$ and $\mathcal{C}_{\bar{J}}(\mathbf{x}_{\bar{J}}) = \sum_{j=3}^{n-1} C_j(x_j, x_{j+1})$ being *independent*. In this case, using (41) we can update the component p_j , $j = 2$ of \mathbf{p} as

$$\begin{aligned} p_j &= \frac{\mathbb{E}_{\mathbf{u}} [X_j \exp \{-\lambda (\mathcal{C}_J(\mathbf{X}_J) + \mathcal{C}_{\bar{J}}(\mathbf{X}_{\bar{J}}))\}]}{\mathbb{E}_{\mathbf{u}} [\exp \{-\lambda (\mathcal{C}_J(\mathbf{X}_J) + \mathcal{C}_{\bar{J}}(\mathbf{X}_{\bar{J}}))\}]} \\ &= \frac{\mathbb{E}_{\mathbf{u}_J} [X_j \exp \{-\lambda \mathcal{C}_J(\mathbf{X}_J)\}]}{\mathbb{E}_{\mathbf{u}_J} [\exp \{-\lambda \mathcal{C}_J(\mathbf{X}_J)\}]} \end{aligned} \quad (87)$$

Suppose for a moment that C_3 does not vanish, so that $\mathcal{C}_J(\mathbf{X}_J)$ and $\mathcal{C}_{\bar{J}}(\mathbf{X}_{\bar{J}})$ are dependent. Then, obviously, (87) is not valid any more. Nevertheless, for block-separable functions this formula can still be used as an approximation to the true updating formula (41). The advantage of p_j based on (87) is a greatly reduced variance of the estimator (24) as compared to the CMC.

Remark 5.1 (Depth- r Updating) We explain now the notion of *depth- r updating* by considering again the block-separable function $\mathcal{C}(\mathbf{x}) = C_1(x_1, x_2) + C_2(x_2, x_3) + \dots + C_{n-1}(x_{n-1}, x_n)$. To update p_j via deterministic IME we need to identify the index set $J_j = \{k : x_k \text{ is in the same block as } x_j\}$. For example, $J_2 = \{1, 2, 3\}$, and $J_3 = \{2, 3, 4\}$. Let $J_2^{(2)} = \cup_{k \in J_2} J_k$ be the set of indices that are in the same block as at least one of the elements in J_2 . Thus, in this example $J_2^{(2)} = \{1, 2, 3, 4\}$. Instead of updating p_2 via $J = J_2$, one could take $J = J_2^{(2)}$ instead. We call this *depth-2 updating*. By similar reasoning one can define depth-3, depth-4, etc. updating. For example, the depth-4 index set for p_2 is $J_2^{(4)} = \{1, 2, 3, 4, 5, 6\}$.

With this in mind we can estimate the vector \mathbf{p}_t by analogy to (60) (see also (87)) as

$$\widehat{\mathbf{p}}_{t,j} = \frac{\sum_{k=1}^N X_{kj} \exp\{-\lambda \mathcal{C}^{(n)}(\mathbf{X}_{k,J}^{(r)})\} W_J^{(r)}(\mathbf{X}_{k,J}^{(r)}; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})}{\sum_{k=1}^N \exp\{-\lambda \mathcal{C}^{(n)}(\mathbf{X}_{k,J}^{(r)})\} W_J^{(r)}(\mathbf{X}_{k,J}^{(r)}; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})}, \quad (88)$$

where $W_J^{(r)}$ is the likelihood ratio based on depth- r , which should be selected in advance similar to (60), $\mathcal{C}^{(n)}$ is defined in (59) and λ is fixed. Note that for full depth we have $W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})$.

The main draw-back of the standard CE for counting is that because of the indicator $I_{\{\mathcal{C}(\mathbf{X}_k) \geq \widehat{m}_t\}}$, one needs to use in CE the entire (high-dimensional) likelihood ratio term W instead of the depth- r alternative (as IME does in (88)), while updating \mathbf{p} according to (55). For more details see [19].

Our extensive numerical results indicate that, although depth-1 introduces a little bias, while estimating \mathbf{p} , the variance reduction, as compared to CMC, is quite substantial. It is not difficult to check that for the weight functions (see (83))

$$C_i(\mathbf{X}) = \left(\sum_{k=1}^n a_{ik} X_k - b_i \right)^r$$

(suitable for counting problems with equality constraints) depth-1 produces an unbiased estimator, provided $r = 2$.

If not stated otherwise, we shall use below in SME the depth-1 updating (88). *In other words, we shall use the SME counting Algorithm 4.3, provided the updating of \mathbf{p} is performed according to (88) rather than according to (60) as in the original Algorithm 4.3.*

6 Combining CE with SME

Motivated by SME we shall present now alternative to the standard CE algorithm, which combines the features of both CE and SME. It is called the *combined CE* (CCE).

To motivate CCE we shall first introduce a modification of MCE, which is based on the elite sampling. Let as before \widehat{m}_t , ($\widehat{m}_t \leq m$) denotes the $\rho\%$ elite value of $\mathcal{C}(\mathbf{X}_k) = \sum_{i=1}^m C_i(\mathbf{X}_k)$, $k = 1, \dots, N$ obtained at the t -iteration.

1. Define $\widehat{\mathcal{C}}(\mathbf{X}_k) = \mathcal{C}(\mathbf{X}_k)$, if $\mathcal{C}(\mathbf{X}_k) \geq \widehat{m}_t$; and $\widehat{\mathcal{C}}(\mathbf{X}_k) = 0$, otherwise.

2. Define $\widehat{C}_i(\mathbf{X}_k) = C_i(\mathbf{X}_k)$, if $\mathcal{C}(\mathbf{X}_k) \geq \widehat{m}_t$; and $\widehat{C}_i(\mathbf{X}_k) = 0$, otherwise.

In short, we set each $\widehat{C}_i(\mathbf{X}_k)$ either to $C_i(\mathbf{X}_k)$ or to 0, depending on whether or not $\mathcal{C}(\mathbf{X}_k)$ belongs to the elite sampling.

Let us replace now the values $\mathcal{C}^{(n)}(\mathbf{X}_k)$ in (60) by their elite counterparts $\widehat{\mathcal{C}}^{(n)}(\mathbf{X}_k)$, where in analogy to (59) we define

$$\widehat{\mathcal{C}}^{(n)}(\mathbf{X}_k) = \frac{\widehat{\mathcal{C}}(\mathbf{X}_k)}{\max_{k=1, \dots, N} \mathcal{C}(\mathbf{X}_k)}. \quad (89)$$

It is readily seen that by replacing in (60) $\mathcal{C}^{(n)}(\mathbf{X}_k)$ with $\widehat{\mathcal{C}}^{(n)}(\mathbf{X}_k)$ the updating of \mathbf{p} will almost remain the same. The main reason is that λ is a very large negative number and that the updating is performed mainly based on the maximum (elite) value of $\mathcal{C}^{(n)}(\mathbf{X}_k)$. Clearly, if λ would be finite, or instead of $e^{-\lambda \widehat{\mathcal{C}}^{(n)}(\mathbf{X}_k)}$ we could

use a different (say, slowly changing) function of $\widehat{\mathcal{C}}(\mathbf{X})$, then the elite sample would matter. This is exactly what we are going to do next. In particular, we define the following updating rule

$$\widehat{\mathbf{p}}_{t,j} = \frac{\sum_{k=1}^N X_{kj} \widehat{\mathcal{C}}(\mathbf{X}_k) W(\mathbf{X}_k, \widehat{\mathbf{p}}_{t-1})}{\sum_{k=1}^N \widehat{\mathcal{C}}(\mathbf{X}_k) W(\mathbf{X}_k, \widehat{\mathbf{p}}_{t-1})}, \quad (90)$$

which combines both (55) and (60) (CE and SME). Note that in contrast to SME, the updating (90) contains elite sampling; but in contrast to CE it contains no indicators. It is called *combined CE* (CCE) updating.

Similarly, since no indicators are involved in (90) we can use here, by analogy to (88), the depth- r updating. To do so we can first write (90) as

$$\widehat{\mathbf{p}}_{t,j} = \frac{\sum_{k=1}^N X_{kj} \sum_{i=1}^m \widehat{\mathcal{C}}_i(\mathbf{X}_k) W(\mathbf{X}_k, \widehat{\mathbf{p}}_{t-1})}{\sum_{k=1}^N \sum_{i=1}^m \widehat{\mathcal{C}}_i(\mathbf{X}_k) W(\mathbf{X}_k, \widehat{\mathbf{p}}_{t-1})} \quad (91)$$

and then manipulate with the $\widehat{\mathcal{C}}_i(\mathbf{X}_k)$'s similar as we did in (88) with $\widehat{\mathcal{C}}_i(\mathbf{X}_k)$'s.

Note that the CCE updating (91) is based on the following CE program

$$\max_{\mathbf{p}} \sum_{k=1}^N \sum_{i=1}^m \widehat{\mathcal{C}}_i(\mathbf{X}_k) W(\mathbf{X}_k, \widehat{\mathbf{p}}_{t-1}) \ln f(\mathbf{X}_k, \mathbf{p}). \quad (92)$$

The combined CE (CCE) Algorithm below differs from the standard CE Algorithm 4.2 only in updating the vector \mathbf{p} , namely instead of the updating (55) we use the updating (91), while all other data remaining the same.

Algorithm 6.1 (CCE Algorithm for Counting)

1. Find the components of the vector $\boldsymbol{\pi} = (\pi_1, \dots, \pi_n)$. Define $\widehat{\mathbf{p}}_0 = \mathbf{u}$. Set $t = 1$ (iteration = level counter).
2. Generate a sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from the density $f(\mathbf{x}; \widehat{\mathbf{p}}_{t-1})$ and compute \widehat{m}_t according to (52).
3. Use the **same** sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ and update \mathbf{p}_t according to (91). Denote the solution by $\widehat{\mathbf{p}}_t$.
4. Smooth out the vector $\widehat{\mathbf{p}}_t$ according to (56).
5. If $\widehat{m}_t < m$, set $t = t + 1$ and reiterate from step 2. Else proceed with step 6.
6. Reiterate steps 2.-4. for 3-4 more iterations. Estimate the counting quantity $|\mathcal{X}^*|$ according to (57).

7 SME for Unconstrained Optimization, Single Event Probabilities and Counting

In this section we apply of Algorithm 4.3 to single-event probability estimation, that is, to estimation of ℓ given in (20), and to an associated unconstrained combinatorial

optimization problem. In particular, we shall associate the rare-event probability estimation and optimization problems with the MinxEnt program (40), provided the second constraint is omitted. Note that in this case the parameter b must be updated iteratively using elite sampling and thus in contrast to SME where only a sequence of vectors $\{\widehat{\boldsymbol{p}}_t, \}$ is generated, here we must generate a sequence of tuples $\{\widehat{\boldsymbol{p}}_t, \widehat{b}_t\}$. Also, since the program (40) is based on the indicator function, and not on the weighted functions, we shall use the original name for the proposed algorithm below for unconstrained optimization and counting, that is, we will call it the IME Algorithm. We show that the IME algorithm can handle efficiently rare events, counting and combinatorial optimization, like max-cut and TSP while employing the particular case of the MinxEnt (40). We shall compare numerically the performance of the IME algorithm with its well known counterparts, CE, VM (variance minimization) and MinxEnt [17] in Section 9.

Note that many counting and optimization problem can be often treated using the MinxEnt programs (37) either in the framework of $m = 1$ or of $m > 1$ (both are single-constrained MinxEnt programs). Take, for example, the TSP. It can be treated either in the framework of program (40) or (37). In the former case TSP is formulated as an unconstrained COP [21], while in the latter as the following constrained one.

$$\begin{aligned}
& \min \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \\
& \text{s.t.} \quad \sum_{i, i \neq j} a_{ij} x_{ij} = 1, \quad j = 1, \dots, n \\
& \quad \quad \sum_{j, j \neq i} a_{ij} x_{ij} = 1, \quad i = 1, \dots, n \\
& \quad \quad \sum_{i \in K} \sum_{j \in K} a_{ij} x_{ij} \leq |K| - 1, \quad 2 \leq |K| \leq n - 1, \quad \forall K \subset \{1, \dots, n\} \\
& \quad \quad x_{ij} \in \{0, 1\}, \quad \forall i, j, \quad i \neq j
\end{aligned} \tag{93}$$

Note that if the cities i and j are connected, then $a_{ij} = 1$; and $a_{ij} = 0$, otherwise. Note also that the constraint

$$\sum_{i \in K} \sum_{j \in K} a_{ij} x_{ij} \leq |K| - 1, \quad 2 \leq |K| \leq n - 1, \quad \forall K \subset \{1, \dots, n\}$$

can be written equivalently as

$$\sum_{i \in K} \sum_{j \notin K} a_{ij} x_{ij} \geq 1, \quad \forall K \subset \{1, \dots, n\}, \quad K \neq \emptyset.$$

Here K is a non-empty set of the cities $1, \dots, n$. Note that there are $n(n - 1)$ 0-1 variables in the program (93). To make sure that all variables x_{ii} will be 0 we set very large values for each c_{ii} , say we set each $c_{ii} = n \max_{i,j} c_{ij}$.

Note also that the problem of counting Hamiltonian cycles can be viewed as a particular case of TSP with the elements c_{ij} 's in (93) being either 1's or infinities, depending on whether the cities are connected or not. Taking into account that in a Hamiltonian cycle the length of a complete tour (if any) is n , we can set $b = n$.

At this end recall that

1. The standard MinxEnt [17] is based on program (25), while the IME on program (40).
2. The programs (40) and (25) are different in the sense that in the former we require that the expectation of the indicator $\mathbb{E}\{I_{\{S(\boldsymbol{X}) \geq b\}}\} = 1$, while for the latter we require that the expectation $\mathbb{E}\{S(\boldsymbol{X})\} \geq b$.

3. The parameter vector \mathbf{p} in MinxEnt is updated according to (26) where λ is obtained from (7), while IME according to (41), provided $m = 1$.
4. The crucial difference between the two methods is that in the standard MinxEnt a sequence of the triplets $\{\mathbf{p}_t, b_t, \lambda_t\}$ is generated [17], while in IME only a sequence of tuples $\{\mathbf{p}_t, b_t, \}$ is generated, while λ is fixed and equal to a large negative number.

To motivate the IME program (41) consider again the die rolling example.

Example 7.1 (The Die Rolling Example Using Program (41)) Table 8 presents data similar to Table 1 for the die rolling example using the MinxEnt program (40) with $S(X) = X$. In particular it presents λ , \mathbf{p} and the entropy $\mathcal{S}(\mathbf{p})$ as functions of b for a fair die with the indicator of X , while calculating $\ell = \mathbb{P}(X \geq b)$ using (40). One can see that from the comparison of Table 8 and Table 1 that the entropy $\mathcal{S}(\mathbf{p})$ in the latter is smaller than in the former, which is based on the MinxEnt program (25).

Table 8: λ , \mathbf{p} and $\mathcal{S}(\mathbf{p})$ as function of b for a fair die while calculating $\ell = \mathbb{P}(X \geq b)$

b	p_1	p_2	p_3	p_4	p_5	p_6	$\mathcal{S}(\mathbf{p})$
1.0	0.1666	0.1666	0.1666	0.1666	0.1666	0.1666	1.7917
2.0	0	0.2	0.2	0.2	0.2	0.2	1.6094
3.0	0	0	0.25	0.25	0.25	0.25	1.3863
3.5	0	0	0	0.33	0.33	0.33	1.0485
4.0	0	0	0	0.33	0.33	0.33	1.0485
5.0	0	0	0	0	0.5	0.5	0.6931
6.0	0	0	0	0	0	1	0

The Unconstrained Case: IME for Optimization

Consider the following non-smooth (continuous or discrete) unconstrained optimization program.

$$\max_{\mathbf{x} \in \mathbb{R}^n} S(\mathbf{x}).$$

Denote by b^* , the optimal function value.

In this case the IME program becomes

$$\begin{aligned} \min_g \mathcal{D}(g, h) &= \min_g \mathbb{E}_g \left[\ln \frac{g(\mathbf{X})}{h(\mathbf{X})} \right] \\ \text{s.t. } \mathbb{E}_g \{ I_{\{S(\mathbf{X}) \leq b\}} \} &= 1 \\ \int g(\mathbf{x}) d\mathbf{x} &= 1, \end{aligned} \tag{94}$$

The corresponding updating of the component of the the vector $\hat{\mathbf{p}}_t$ can be written as

$$\hat{p}_{t,j} = \frac{\sum_{k=1}^N X_{kj} \exp\{-\lambda I_{\{S(\mathbf{x}_k) \leq \hat{b}_t\}}\}}{\sum_{k=1}^N \exp\{-\lambda I_{\{S(\mathbf{x}_k) \leq \hat{b}_t\}}\}}, \tag{95}$$

where λ is a big negative number.

Algorithm 7.1 (IME Unconstrained Optimization Algorithm)

1. Define $\hat{\mathbf{p}}_0 = \mathbf{u}$, say chose, $f(\mathbf{x}, \mathbf{u})$ uniformly distributed over \mathcal{X} . Set λ to a big negative number, say $\lambda = -100$. Set, $t = 1$ (iteration = level counter).
2. Generate a sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from the density $f(\mathbf{x}; \hat{\mathbf{p}}_{t-1})$ and compute the elite sampling value \hat{b}_t of $(\mathbf{X}_1), \dots, S(\mathbf{X}_N)$.
3. Use the **same** sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ and compute $\hat{\mathbf{p}}_t$, according to (95).
4. Smooth out the vector $\hat{\mathbf{p}}_t$ according to (56).
5. If the stopping criterion is met, stop; otherwise, set $t = t + 1$ and return to Step 2.

As a **stopping criterion** one can use for example: if for some $t \geq d$, say $d = 5$,

$$\hat{b}_{t-1,(N)} = \hat{b}_{t,(N)} = \dots = \hat{b}_{t-d,(N)} \quad (96)$$

then **stop**.

IME for Single-Event Probability Estimation and Counting

The IME algorithm for single event event probability estimation (see (20)) combines both the original counting Algorithm (4.3) and the optimization Algorithm 7.1. In particular, in order to obtain the single event probability estimation algorithm from Algorithm 7.1 we only need to add LR (likelihood ratio) terms into (95), while updating $\hat{\mathbf{p}}_t$, and finally estimate $|\mathcal{X}^*|$ according to (57). The final rare-event estimator of ℓ can be written as

$$\hat{\ell} = \frac{1}{N} \sum_{k=1}^N I_{\{S(\mathbf{X}_k) \geq b\}} \frac{f(\mathbf{X}_k; \mathbf{u})}{f(\mathbf{X}_k; \hat{\mathbf{p}}_T)}. \quad (97)$$

Our numerical results show that for rare-events and counting (the unconstrained case), IME performs similar to its standard CE counterpart.

Example 7.2 Internet Security The nternet security problem reads as: find two primes, provided we are given a large integer b known to be a product of these primes.

We can write the given number b in the *binary system* with $n + 1$ -bit integers as

$$b = \alpha_0 + 2\alpha_1 + \dots + 2^n \alpha_n.$$

The problem reads as: find binary $x_i, y_j, i, j = 0, 1, \dots, n$ such that

$$\begin{aligned} \sum_{i,j} 2^{i+j} x_i y_j &= (\sum_i 2^i x_i)(\sum_j 2^j y_j) = b \\ x_i \in \{0, 1\}, y_j \in \{0, 1\}, \forall i, j &= 0, 1, \dots, n. \end{aligned} \quad (98)$$

Note that

- The optimal MinxEnt and IME pdf's are

$$g(\mathbf{x}, \mathbf{y}) = \frac{h(\mathbf{x}, \mathbf{y}, \mathbf{u}) \exp \left\{ -(\sum_i 2^i x_i)(\sum_k 2^k y_k) \lambda \right\}}{\mathbb{E}_{\mathbf{u}} \exp \left\{ -(\sum_i 2^i X_i)(\sum_k 2^k Y_k) \lambda \right\}}, \quad j = 1, \dots, n. \quad (99)$$

and

$$g(\mathbf{x}, \mathbf{y}) = \frac{h(\mathbf{x}, \mathbf{y}, \mathbf{u}) \exp \left\{ -I_{\{(\sum_i 2^i x_i)(\sum_j 2^j y_j) = b\}} \lambda \right\}}{\mathbb{E}_{\mathbf{u}} [\exp \left\{ -I_{\{(\sum_i 2^i X_i)(\sum_j 2^j Y_j) = b\}} \lambda \right\}]}, \quad j = 1, \dots, n, \quad (100)$$

respectively.

- For prime numbers the MinxEnt should deliver 2 symmetric solutions: one as $(\sum_i 2^i x_i)$ and $(\sum_j 2^j y_j)$; and the other as $(\sum_j 2^j y_j)$ and $(\sum_i 2^i x_i)$. For non prime numbers MinxEnt should have more than 2 solutions.
- The problem (98) can be formulated for any basis, say for a decimal rather than binary one.
- To have a unique solution we can impose in addition the following constraint $\sum_i 2^i x_i > \sum_j 2^j y_j$.

We shall show how to apply both programs (25) and (40) with their corresponding solutions

$$p_j = \frac{\mathbb{E}_{\mathbf{u}}[X_j \exp\{-S(\mathbf{X})\lambda\}]}{\mathbb{E}_{\mathbf{u}}[\exp\{-S(\mathbf{X})\lambda\}]}, \quad j = 1, \dots, 2n \quad (101)$$

and

$$p_j = \frac{\mathbb{E}_{\mathbf{u}}[X_j \exp\{-I_{\{S(\mathbf{X})=b\}}\lambda\}]}{\mathbb{E}_{\mathbf{u}}[\exp\{-I_{\{S(\mathbf{X})=b\}}\lambda\}]}, \quad j = 1, \dots, 2n, \quad (102)$$

respectively.

Substituting $S(\mathbf{x}, \mathbf{y}) = (\sum_i 2^i x_i)(\sum_j 2^j y_j)$ into (101) and (102) we obtain

$$p_j^{\mathbf{X}} = \frac{\mathbb{E}_{\mathbf{u}}[X_j \exp\{-(\sum_i 2^i X_i)(\sum_k 2^k Y_k)\lambda\}]}{\mathbb{E}_{\mathbf{u}}[\exp\{-(\sum_i 2^i X_i)(\sum_k 2^k Y_k)\lambda\}]}, \quad j = 1, \dots, n. \quad (103)$$

and

$$p_j^{\mathbf{X}} = \frac{\mathbb{E}_{\mathbf{u}}[X_j \exp\{-I_{\{(\sum_i 2^i X_i)(\sum_j 2^j Y_j)=b\}}\lambda\}]}{\mathbb{E}_{\mathbf{u}}[\exp\{-I_{\{(\sum_i 2^i X_i)(\sum_j 2^j Y_j)=b\}}\lambda\}]}, \quad j = 1, \dots, n, \quad (104)$$

respectively and similarly for their $p_j^{\mathbf{Y}}$ counterparts. Note that

1. Formula(103) can be written as

$$p_j^{\mathbf{X}} = \frac{\frac{1}{2}\mathbb{E}_{\mathbf{u}}[\exp\{-(\sum_{i, i \neq j} 2^i X_i + 2^j)(\sum_k 2^k Y_k)\lambda | X_j = 1\}]}{\mathbb{E}_{\mathbf{u}}[\exp\{-(\sum_i 2^i X_i)(\sum_k 2^k Y_k)\lambda\}]}, \quad j = 1, \dots, n \quad (105)$$

and similarly its (104) counterpart.

2. $Z^{(1)} = \sum_{i=0}^n 2^i X_i$ has a discrete uniform distribution over the points $\{0, 1, \dots, \sum_{i=0}^n 2^i\}$ and similarly $Z^{(2)} = \sum_{i=0}^n 2^i Y_i$.
3. An alternative way of getting the p_j 's in (101) and (102) is to find the marginals of (99) and (100) by integrating out the corresponding joint pdf's. By doing so we automatically obtain that the j -th marginal $g_j(x_j)$ will have a Ber(p_j) distribution with p_j given in (101) and (102), respectively.
4. To get the pair-wise marginal pdf's $g_{j,k}(x_j, y_k)$, we need again to integrate out (99) and (100) with respect to $x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_n, y_1, \dots, y_{k-1}, y_{k+1}, \dots, y_n$. By doing so we automatically obtain that the (j, k) -th marginal $g_{j,k}(x_j, y_k)$ will have a two-parametric Ber($p_j^{\mathbf{X}}, p_k^{\mathbf{Y}}$) distribution. The parameter vector with the components $(p_j^{\mathbf{X}}, p_k^{\mathbf{Y}})$ can be calculated similar to (73), that is, directly (without resorting to $g_{j,k}(x_j, y_k)$).

We shall show how to calculate the vector (104) and the pair-wise dependent marginals $g_{i_1 i_2}(x_{i_1}, x_{i_2})$ (see (75) and the associated parameters $p_{x_i x_j}$ in (73)) using IME for the particular case $b = 77$. In this case there are two symmetric solutions: $(\mathbf{x}, \mathbf{y}) = ((0, 1, 1, 1), (1, 0, 1, 1))$ and $(\mathbf{x}, \mathbf{y}) = ((1, 0, 1, 1), (0, 1, 1, 1))$.

Denote by $C(\mathbf{x}, \mathbf{y}) = I_{\{\sum_{i,j} 2^{i+j} x_i y_j = 77\}}$. According to Lemma (3.1), $\lambda = -\infty$. Further more, according to Lemma 3.2, the optimal nonparametric IS distribution $g(\mathbf{x}, \mathbf{y})$ is uniformly distributed on the feasible set That is $g((0, 1, 1, 1), (1, 0, 1, 1)) = g((1, 0, 1, 1), (0, 1, 1, 1)) = 0.5$ and 0 otherwise.

The components of the vectors $\mathbf{p}^{\mathbf{X}}$ and $\mathbf{p}^{\mathbf{Y}}$ can be written as

$$p_j^{\mathbf{X}} = \frac{\mathbb{E}_h [X_j \exp \{-\lambda C(\mathbf{X}, \mathbf{Y})\}]}{\mathbb{E}_h [\exp \{-\lambda C(\mathbf{X}, \mathbf{Y})\}]} = \frac{0.5 \cdot \mathbb{E}_h [\exp \{-\lambda C(\mathbf{X}, \mathbf{Y})\} | X_j = 1]}{\mathbb{E}_h [\exp \{-\lambda C(\mathbf{X}, \mathbf{Y})\}]}, \quad k = 0, 1, 2, 3$$

$$p_j^{\mathbf{Y}} = \frac{\mathbb{E}_h [Y_j \exp \{-\lambda C(\mathbf{X}, \mathbf{Y})\}]}{\mathbb{E}_h [\exp \{-\lambda C(\mathbf{X}, \mathbf{Y})\}]} = \frac{0.5 \cdot \mathbb{E}_h [\exp \{-\lambda C(\mathbf{X}, \mathbf{Y})\} | Y_j = 1]}{\mathbb{E}_h [\exp \{-\lambda C(\mathbf{X}, \mathbf{Y})\}]}, \quad k = 0, 1, 2, 3$$

Calculating these expressions for $\lambda = -100$ yields

$$p_j^{\mathbf{X}} = p_j^{\mathbf{Y}} = \begin{cases} 1, & j=0 \\ 1, & j=1 \\ 0.5, & j=2 \\ 0.5, & j=3 \end{cases}$$

The number of feasible solutions can be estimated using IS as

$$|\widehat{\mathcal{X}^*}| = \frac{1}{N} \sum_{k=1}^N I_{\{(\mathbf{X}_k, \mathbf{Y}_k) \text{ feasible}\}} \frac{1}{f(\mathbf{X}_k, \mathbf{p}^{\mathbf{X}})f(\mathbf{Y}_k, \mathbf{p}^{\mathbf{Y}})},$$

where

$$(\mathbf{X}_k, \mathbf{Y}_k) \sim f(\mathbf{x}, \mathbf{p}^{\mathbf{X}})f(\mathbf{y}, \mathbf{p}^{\mathbf{Y}}) = \prod_{i=0}^3 \text{Ber}(x_i; p_i^x) \text{Ber}(y_i; p_i^y).$$

Using a sample $N = 1000$ it produces a 95%-confidence interval (2.032 ± 0.3304) . Although the method required here more computations than the exhaustive search (which required only 256 feasibility evaluations), the variance of its estimate is reduced as compared with the CMC method. (Using the same $N = 1000$, CMC produced a 95%-confidence interval (3.072 ± 1.7286)).

8 Applications

Here we present several well known counting and optimization problems, for which our algorithms can be useful.

Knapsack Problem

We consider here two well known knapsack problems, the so-called *bounded* knapsack problem, which reads as

$$\begin{aligned} & \max \sum_{k=0}^r c_k x_k \\ \text{s.t.} \quad & \sum_{k=1}^n a_k x_k \leq b \\ & x_k \in \{0, 1, \dots, r\}, \forall k = 1, \dots, n \end{aligned} \tag{106}$$

and the so-called *multiple knapsack* problem, which reads as

$$\begin{aligned} & \max \sum_{i=1}^m \sum_{k=1}^n c_k x_{ik} \\ \text{s.t.} \quad & \sum_{k=1}^n a_k x_{ik} \leq b_i, \quad \forall i=1, \dots, m \\ & \sum_{i=1}^m x_{ik} \leq 1, \quad \forall k = 1, \dots, n \\ & x_{ik} \in \{0, 1\}, \quad \forall i = 1, \dots, m; \quad k = 1, \dots, n. \end{aligned} \tag{107}$$

Note that here all a_k, b_k, c_k are fixed constants.

Set Covering, Set Packing and Set Partitioning

Note that set partition program reduces to the program (78), provided A is a 0-1 matrix, $x_i \in \{0, 1\}$, $\forall i = 1, \dots, n$ and the vector $\mathbf{b} = \mathbf{1}$, provided $m_2 = 0$ and the minimization is replaced by maximization. The set covering and set packing problems are similar to the set partition one, provided the equality constraints $\mathbf{Ax} = \mathbf{1}$ is replaced by $\mathbf{Ax} \geq \mathbf{1}$ and $\mathbf{Ax} \leq \mathbf{1}$, respectively.

Consider a finite set $\{M = 1, 2, \dots, m\}$ and let $M_j, j \in N$ be a collection of subsets of the set M where $N = \{1, 2, \dots, n\}$. A subset $F \subseteq N$ is called a *cover* of M if $\cup_{j \in F} M_j = M$. The subset $F \subseteq N$ is called a *packing* of M if $M_j \cap M_k = \emptyset$ for all $j, k \in F$ and $j \neq k$. If $F \subseteq N$ is both a cover and packing, then it is called a *partitioning*.

Suppose c_j is the cost associated with M_j . Then the set covering problem is to find a minimum cost cover. If c_j is the value or weight of M_j , then the set packing problem is to find a maximum weight or value packing. Similarly, the set partitioning problem is to find a partitioning with minimum cost. These problems can be formulated as zero-one linear integer programs as shown below. For all $i \in M$ and $j \in N$, let

$$a_{ij} = \begin{cases} 1 & \text{if } i \in M_j \\ 0 & \text{otherwise} \end{cases}$$

and

$$x_j = \begin{cases} 1 & \text{if } j \in F \\ 0 & \text{otherwise} \end{cases}$$

Then the set covering, set packing and set partitioning formulations are given by

$$\begin{aligned} \min \quad & \sum_{j=1}^n c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^n a_{ij} x_j \geq 1 \quad i = 1, 2, \dots, m \\ & x_j = 0, 1 \quad j = 1, 2, \dots, n, \end{aligned}$$

$$\begin{aligned} \max \quad & \sum_{j=1}^n c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^n a_{ij} x_j \leq 1 \quad i = 1, 2, \dots, m \\ & x_j = 0, 1 \quad j = 1, 2, \dots, n, \end{aligned}$$

and

$$\begin{aligned} \max \quad & \sum_{j=1}^n c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^n a_{ij} x_j = 1 \quad i = 1, 2, \dots, m \\ & x_j = 0, 1 \quad j = 1, 2, \dots, n, \end{aligned}$$

respectively.

Bipartite Matching and Permanent

From now on we will consider the case where all a_{ij} 's are 0-1. It is well known that calculating of a permanent can be reduced to calculation of *perfect matching* in a bipartite graph $G((V_1, V_2), E)$ with independent set of nodes $V_1 = (v_{11}, \dots, v_{1n})$ and $V_2 = (v_{21}, \dots, v_{2n})$. Note that in a bipartite graph every edge has one node in V_1 and another in V_2 and no two edges share a common vertex. Note also that a matching is a collection of edges $M \subseteq E$ such that each vertex occurs at most once in M . A perfect matching is a matching of size n . Let Q_i denotes the set of

matching of size i in G . Assume that Q_n is non-empty, so that G has a perfect matching of vertices V_1 and V_2 . It is well known [12] that the number of perfect matchings in G equals to the permanent A , that is, $|Q_n| = \text{per}(A)$, where $\text{per}(A)$ is defined as

$$\text{per}(A) = |\mathcal{X}^*| = \sum_{\mathbf{x} \in \mathcal{X}} \prod_{i=1}^n a_{ix_i}. \quad (108)$$

Here \mathcal{X} is the set of all permutations $\mathbf{x} = (x_1, \dots, x_n)$ of $(1, \dots, n)$ and the elements a_{ij} can be written as

$$a_{ij} = \begin{cases} 1, & \text{if the nodes } v_{1i} \text{ and } v_{2j} \text{ are in } E \\ 0, & \text{otherwise.} \end{cases}$$

The general matching problem can be written as

$$\begin{aligned} & \min \sum_{i=1}^n \sum_{j=1}^n a_{ij} x_{ij} \\ \text{s.t. } & \sum_{k=1}^n x_{ik} = 1, \quad \forall i = 1, \dots, n \\ & \sum_{j=1}^n x_{ji} = 1, \quad \forall i = 1, \dots, n \\ & x_{ij} \in \{0, 1\}, \quad 1 \leq i \leq j \leq n. \end{aligned} \quad (109)$$

The SAT Problem

The most common SAT problem comprises the following two components:

- A set of n Boolean variables $\{x_1, \dots, x_n\}$, representing statements that can either be **TRUE** ($=1$) or **FALSE** ($=0$). The negation (the logical **NOT**) of a variable x is denoted by \bar{x} . For example, $\overline{\text{TRUE}} = \text{FALSE}$. A variable or its negation is called a *literal*.
- A set of m distinct *clauses* $\{S_1, S_2, \dots, S_m\}$ of the form $S_i = z_{i1} \vee z_{i2} \vee \dots \vee z_{ik}$, where the z 's are literals and the \vee denotes the logical **OR** operator. For example, $0 \vee 1 = 1$.

The binary vector $\mathbf{x} = (x_1, \dots, x_n)$ is called a *truth assignment*, or simply an *assignment*. Thus, $x_i = 1$ assigns truth to x_i and $x_i = 0$ assigns truth to \bar{x}_i , for each $i = 1, \dots, n$. The simplest SAT problem can now be formulated as: *find a truth assignment \mathbf{x} such that all clauses are true.*

Denoting the logical **AND** operator by \wedge , we can represent the above SAT problem via a single *formula* as

$$F_1 = S_1 \wedge S_2 \wedge \dots \wedge S_m,$$

where the $\{S_k\}$ consist of literals connected with only \vee operators. The SAT formula is then said to be in *conjunctive normal form* (CNF).

The problem of deciding whether there *exists* a valid assignment, and, indeed, providing such a vector, is called the *SAT-assignment* problem [19].

It is shown in [19] that the SAT-assignment problem can be modeled via rare-events with ℓ given in (34), that is,

$$\ell = \mathbb{E}_{\mathbf{u}} \left[I_{\{\sum_{i=1}^m C_i(\mathbf{X})=m\}} \right],$$

where \mathbf{u} denotes the “uniform” probability vector $(1/2, \dots, 1/2)$. It is important to note that here each $C_i(\mathbf{X}) = I_{\{\sum_{k=1}^n a_{ik} X_k \geq b_i\}}$ can be also written alternatively as

$$C_i(\mathbf{x}) = \max_j \{0, (2x_j - 1) a_{ij}\}. \quad (110)$$

Here $C_i(\mathbf{x}) = 1$ if clause S_i is TRUE with truth assignment \mathbf{x} and $C_i(\mathbf{x}) = 0$ if it is FALSE, $A = (a_{ij})$ is a given clause matrix that indicate if the literal corresponds to the variable (+1), its negation (-1), or that neither appears in the clause (0). If for example $x_j = 0$ and $a_{ij} = -1$, then the literal \bar{x}_j is TRUE. The entire clause is TRUE if it contains at least one true literal. In other words, ℓ in (34) is the probability that a uniformly generated SAT assignment (trajectory) \mathbf{X} is valid, that is, all clauses are satisfied, which is typically very small. The SAT counting problem has been therefore reduced to a problem involving the estimation of a rare-event probability in the form (20) and one can proceed directly with the single constrained MinxEnt to find the optimal joint n dimensional pdf $g(\mathbf{x})$. A simple analyzes of (34), (110) (see also for details [19]) that such single constrained MinxEnt does not lead to decomposable $g(\mathbf{x})$. As results an iterative time consuming MinxEnt is used in [19]. To find a decomposable $g(\mathbf{x})$ counterpart of the SAT problem we take into account that each S_i is in the form $S_i = z_{i_1} \vee z_{i_2} \vee \dots \vee z_{i_k}$ and then based on the representation $S_1 \wedge S_2 \wedge \dots \wedge S_m$ define the following associated set of linear constraints.

$$z_{i_1} + z_{i_2} + \dots + z_{i_k} \geq 1, \quad i = 1, \dots, m. \quad (111)$$

Clearly the representation (111) fits (32) and the linearity of the constraints causes the decomposability.

Example 8.1 (SAT Example) As a simple example consider the following SAT assignment

$$(x_1 + \bar{x}_2)(\bar{x}_1 + \bar{x}_2 + x_3)(x_2 + x_3)$$

In this case the system of linear constraints (111) reduces to

$$\begin{aligned} x_1 + (1 - x_2) &\geq 1 \\ (1 - x_1) + (1 - x_2) + x_3 &\geq 1, \\ x_2 + x_3 &\geq 1, \end{aligned}$$

where each $x_1, x_2, x_3 \in \{0, 1\}$.

It is not difficult to see that for SAT problems SME reduces to IME. The reason is that in the right hand side of each constraint in SAT we have ≥ 1 . Clearly, if for any constraint the right hand would be not ≥ 1 , SME would be different from IME.

Proceeding with our example we can write ℓ as $\ell = \mathbb{P}_{\mathbf{u}}(C_1 + C_2 + C_3 = 3)$, where $C_1 = I_{\{X_1 - X_2 \geq 0\}}$, $C_2 = I_{\{X_1 + X_2 - X_3 \leq 1\}}$ and $C_3 = I_{\{X_2 + X_3 \geq 1\}}$.

Remark 8.1 DNF form If SAT is given in the DNF form, then one can simple construct a corresponding single-constrained MinxEnt and again find an alternative to (37) optimal pdf $g(\mathbf{x})$. Note that the DNF for of SAT corresponds to constrained LIP programs with the view that constraints given in the “OR”, rather than in the standard “AND” form.

9 Numerical Results

We present here numerical results with the proposed algorithms for counting and unconstrained optimization. Constrained optimization will be considered some where else. For counting we shall use mainly the CCE and the depth-1 SME algorithms.

As we mentioned we found that VM algorithms are more robust than its SME and CE counterparts. In contrast, for unconstrained optimization, like TSP we found that all proposed algorithms perform similarly. The main reason for that is, - *there is no need to use LR's in optimization*. If not stated otherwise we set the rarity parameter $\rho = 0.001$ and the smoothing parameter $\alpha = 0.7$. Note that $\rho = 0.001$ applies to the elite samples only for the intermediate states of our algorithms, that is when $\hat{m}_t < m$. For $\hat{m}_t = m$ we accumulate all elite samples.

A huge collection of instances (including real-world) is available on OR-LIB site :

<http://people.brunel.ac.uk/~mastjjb/jeb/orlib/scpinfo.html>;

for multiple-knapsack on <http://hces.bus.olemiss.edu/tools.html> and

<http://elib.zib.de/pub/Packages/mp-testdata/ip/sac94-suite/index.html>.

Knapsack instances generator is given on: <http://www.diku.dk/pisinger/codes.html>

To study the variability in the solutions we run each problem 10 times and report our statistics based on these 10 runs of our algorithms. In the following tables the quantities are defined as follows (for each iteration t):

1. “Mean, max and min $|\widehat{\mathcal{X}^*}|$ ” denote the sample mean, maximum and minimum and minimal values of the 10 estimates of $|\mathcal{X}^*|$.
2. “Mean, max and min Found” denote the sample mean, maximum and minimum of values found in each of the 10 samples of size N . Note that the maximum value can be viewed as the lower bound of the true unknown quantity $|\mathcal{X}^*|$.
3. PV denotes the the proportion of generated values, averaged over 10 replications.
4. RE denotes the mean relative error for $|\widehat{\mathcal{X}^*}|$, averaged over the 10 runs.
5. λ denotes the mean λ , averaged over the 10 runs.
6. \mathcal{S} denotes the mean entropy averaged over the 10 runs.
7. m denotes the mean number of satisfied constraints at t -th iteration and averaged over the 10 runs.

In all counting problems we compared the performance of the CE-based algorithms like the standard CE, CCE and SME Algorithm 4.3 using depth-1 (with fixed $\lambda = -100$) with their counterparts VM-based counterparts, like standard VM, with and without LR's . While running the algorithms we found that for some particular instances, all CE-based algorithms produce incorrect estimators, while their counterpart, the VM-based algorithm, always delivers correct (unbiased) ones. This undesirable phenomenon of CE-based algorithms has not been yet fully understood and it is under investigation. In Section 9.2.2 we present such (“pathological”) 3-SAT example and discuss the performance of CE-based and VM-based algorithms.

Note again that while using the depth-1 SME Algorithm 4.3 we apply weighted function approach, which is based on formulas (83), (84).

In all our numerical studies we generated the matrices $A = (a_{ij})$ randomly and made sure that they are sparse. The sparsity insures that the counting quantity $|\mathcal{X}^*|$ is small (is associated with rare-event probability, that is, the most difficult cases), while random matrices generation insures the diversity of the cases. All cases have been checked first on small randomly generated models, such that $|\mathcal{X}^*|$ is relatively small, say $0 \leq |\mathcal{X}^*| \leq 100$, and such that their exact solution via full enumeration is available. Only, after that larger models have been tested.

To speed up the convergence we implemented the following.

- To speed up the convergence of the SME Algorithm 4.3 we set $\lambda = -10$ for its first 2-3 iterations and for the remaining ones we set $\lambda = -100$.
- In many counting problems involving rare-events the elements of $\hat{\mathbf{p}}_t$ are approaching either 0's or 1's as t increases. We set them automatically either 0's or 1's as soon as they reach, say 0.01 and 0.99, respectively. By doing so, at iterations $t + 1, \dots, T$ one needs to generate and update only a very small portion of \mathbf{p} 's, namely these which remain in the interval (0.01, 0.99).
- If a particular constraint is satisfied simultaneously by several random variables, say by r random variables, then while updating the parameter vector \mathbf{p} the weight for the corresponding variables will be equal to $1/r$. For example, consider the following constraints $X_1 + X_2 + X_3 \geq 1$. Assume that a particular outcome is $X_1 = 0, X_2 = 1, X_3 = 1$. Then while updating the parameter vector $\mathbf{p} = p_1, p_2, p_3$ we put for this particular outcome the following weights $p_1 = 0, p_2 = 1/2, p_3 = 1/2$.

Below we consider separately decision making, counting, rare-event simulation and unconstrained optimization.

9.1 Decision Making

Since counting is typically a difficult problem in many cases decision making is used instead. For example, in a constrained program, one would like to know if there is a feasible solution, that is, all m constrained are satisfied, rather than to count the total number of such feasible solutions. In the nomenclature of our CE-based and VM-based counting algorithms, this is equivalent of saying: our algorithm passes all m constraints successfully (the decision is YES) or it gets stucked somewhere before reaching m (the decision is NO). Intuitively, it is clear that it will be typically easier for a CE-based based VM-based to give a correct (yes) solution if the number of valid trajectories $|\mathcal{X}^*|$ is not small relative to m . Indeed, in our numerical results with decision making we typically did not observe that any of CE-based and VM-based algorithms had a problem of reaching m , provided $n \leq m \leq 500$ and the ratio $|\mathcal{X}^*|/m \geq 0.1$. This means, for example, that for $m = 100$ and for $|\mathcal{X}^*| \geq 10$ all of our counting algorithms will deliver with very high probability the true answer - **yes**. It is crucial to note that like in optimization, no likelihood ratios are involved in decision making.

What happens when $|\mathcal{X}^*|$ is small relative to m , say when $|\mathcal{X}^*|/m \leq 0.02$? We found that this is quite problematic to give a meaningful answer in the sense of the relative error (RE) to any of our algorithms. We also believe that this could be problematic for any randomized algorithm, while requiring the RE of $|\mathcal{X}^*|$ to be within 1-2% from $|\mathcal{X}^*|$.

We address this problem differently by borrowing the notion of relative discrepancy (precision) we have been using for optimization problems [21]. Recall that in optimization, the relative discrepancy of 1-2% of the solution from the best know solution means that if, say the best known solution (maximum problem) is 100, so any solution of an algorithm within the range (98, 100) means that the discrepancy is within 2%.

To keep similarity with optimization we define in parallel to the relative error (RE) what we call the *relative discrepancy* (RD_d) of a decision making algorithm and the *relative discrepancy of a counting algorithm* (RD_c) as

$$RD_d = \begin{cases} \frac{m - \hat{m}_T}{m + |\mathcal{X}^*|}, & \text{if } \hat{m}_T = m, \\ \frac{m + |\mathcal{X}^*| - \hat{m}_T}{m}, & \text{if } \hat{m}_T < m, \end{cases} \quad (112)$$

and as

$$RD_c = \begin{cases} \frac{m+|\mathcal{X}^*|-(\widehat{m}_T+|\widehat{\mathcal{X}}^*|)}{m+|\mathcal{X}^*|}, & \text{if } \widehat{m}_T = m, \\ \frac{m+|\mathcal{X}^*|-\widehat{m}_T}{m}, & \text{if } \widehat{m}_T < m, \end{cases} \quad (113)$$

respectively. Note that we assume in both cases that the constrained set is not empty and thus m exists. To clarify, consider a set with $m=100$ constraints and let $|\mathcal{X}^*|=10$. Assume that in both (decision making and counting) cases we obtained $\widehat{m}_T = m$, that is, our algorithms went through all constraints. In this case we automatically obtain that $RD_d = 0$, since for the (yes, no) decision making problem, the answer will be yes. Assume further that the counting algorithm delivered $|\widehat{\mathcal{X}}^*|=5$ instead of the true $|\mathcal{X}^*|=10$. We have, in this case $RD_c = 5/110 \approx 5\%$. Assume now that in another run, we obtained in both (decision and counting) cases $\widehat{m}_T = 90$, then $RD_d = RD_s = 20\%$, that is, we deliver an error (discrepancy) of 20%. Using the criteria RD_s instead of RE, in all our experiments we performed with counting models (disregarding the ‘‘pathological’’ ones) we obtained that $RD_s < 5\%$, while the original RE could be 50% and even more, in particular for the instances where $\frac{|\mathcal{X}^*|}{m} \leq 0.1$.

9.2 Counting

Below we present performance of our SME and CCE algorithms for counting using independent marginals of $g(\mathbf{x})$. The improvement from using pairwise dependence will be addressed somewhere else. Recall that in all our experiments with the CCE Algorithm 6.1 and SME Algorithm 4.3 we set $\lambda = -100$ and used depth-1 policy.

We also present the performance of our algorithms to count the number of optimal solutions in some constrained optimization problems, where the optimal solution was either obtained via full enumeration (for small models) or (the best known solution) was taken from the web site.

Permanent

Below we present performance of the SME Algorithm 4.3 for the permanent problem with randomly generated matrix A . To proceed, we define the notion of the so-called *random K -permutation matrix*, denotes as *K -PERM matrix*, where K , ($K < n$) means the number of independent uniformly distributed Bernoulli random variables at each row of the permanent matrix A of the size $n \times n$. We found empirically that in order for the true permanent value $|\mathcal{X}^*$ to be very small relative to $|\mathcal{X}|$, (and, thus for $\ell = |\mathcal{X}^*|/|\mathcal{X}|$ to be very small) the parameter K should be chosen as $K \leq 0.2n$.

Table 9 present the performance of the depth-1 SME Algorithm 4.3 for a 4-PERM randomly generated 20×20 permanent matrix using $N = 100,000$ samples with $|\mathcal{X}^*| = 2$, which was obtained using full enumeration.

One can see that the SME Algorithm 4.3 performs quite accurately. For this instance we also run the indicator version of Algorithm 4.3, which is based on the indicators (80), (81). We found that the weighted SME version with $r = 3$ outperforms a little bit its indicator counterpart as far as the accuracy, (relative error (RE)) is concerned, while the remaining parameters are similar to Table 9.

Table 9: Performance of the SME Algorithm 4.3 for a 4-PERM randomly generated 20×20 permanent matrix using $N = 100,000$ samples.

t	Mean	Max	Min	PV	RE	\mathcal{S}	m
0	0.0	0.0	0.0	0.00	NaN	6.58	16
1	2.0	2.8	1.4	0.00	0.180	5.56	18
2	2.0	2.1	1.9	0.01	0.025	3.80	20
3	2.0	2.1	2.0	0.02	0.025	2.78	20
4	2.0	2.1	2.0	0.03	0.012	2.35	20
5	2.0	2.0	1.9	0.03	0.017	2.18	20
6	2.0	2.0	2.0	0.03	0.016	2.11	20

We also compared the performance of the SME Algorithm 4.3 with that of the MinxEnt one introduced in [18], where (similar to the TSP trajectories [21]) the permanent trajectories were generated using an auxiliary probability matrix $\mathbf{P} = (p_{ij})$, which is associated with the permanent matrix A . We did not find any advantage of using Algorithm 4.3 as compared to its the MinxEnt counterpart.

Counting Hamiltonian Cycles

Below we present performance of the CCE Algorithm 4.3 for counting Hamiltonian cycles with a randomly generated matrix A . Similar, to the *random K-PERM* we define the so-called *random K-Hamiltonian matrix*, denoted as *K-HAM matrix*, where, as before, K , ($K < n$) denotes the number of independent uniformly distributed Bernoulli random variables at each row of the matrix A . We found empirically that in order for $|\mathcal{X}^*$ to be very small relative to $|\mathcal{X}|$, the parameter K should be chosen as $K \leq 0.15n$.

Table 10 present the performance the CCE Algorithm 6.1 for a 4-HAM randomly generated (30×30) matrix using $N = 100,000$ samples. The trajectories (tours) were generated using the *node transition algorithm* (see Algorithm 4.7.1 of [21]). The results are self explanatory.

Table 10: Performance of the CCE algorithm for the HC problem for a 4-HAM matrix $A = (30 \times 30)$ and $N = 100,000$.

t	$ \mathcal{X}^* $			Found			PV	RE
	Mean	Max	Min	Mean	Max	Min		
0	36.27	283	0	0.20	1	0	0.0000	2.3608
1	62.84	109	32	22.10	26	17	0.0005	0.3272
2	66.09	76	55	55.10	60	49	0.0065	0.0885
3	62.75	68	56	55.30	62	49	0.0344	0.0496

The SAT Problem

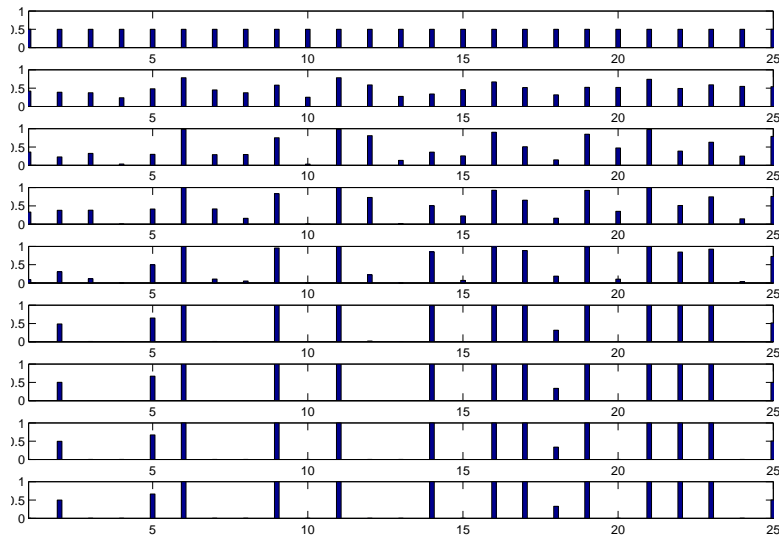
Table 11 presents the performance of the CCE Algorithm 6.1 for a random 3-SAT problem with an instance matrix $A = (25 \times 100)$ for $N = 50,000$. The results are self-explanatory. We also run this problem using the depth-1 SME Algorithm 4.3 and got very similar results. However, running it with the standard CE Algorithm 4.2 the results were worse in particular in terms of relative error.

Table 11: Performance of the CCE algorithm for the random 3-SAT with an instance matrix $A = (25 \times 100)$ and $N = 50,000$.

t	$ \mathcal{X}^* $			Found			PV	RE	S	m	RD_c
	Mean	Max	Min	Mean	Max	Min					
0	0.0	0.0	0.0	0	0	0	0.00	NaN	8.66	96	0.10
1	3.6	21.4	0.0	0	1	0	0.00	2.045	8.16	98	0.08
2	10.8	38.7	0.0	1	3	0	0.00	1.113	7.10	99	0.07
3	6.8	26.8	0.8	4	6	1	0.00	1.027	6.23	99	0.07
4	6.2	8.5	4.2	5	6	3	0.00	0.201	5.74	99	0.07
5	5.0	8.5	2.0	4	6	2	0.03	0.355	5.43	99	0.07
6	5.3	8.0	3.1	5	6	3	0.10	0.265	5.24	100	0.02
7	5.9	9.2	4.1	5	6	3	0.17	0.217	3.53	100	0.00
8	5.3	8.7	1.2	5	6	2	0.23	0.382	2.32	100	0.00
9	5.6	6.0	4.1	6	6	3	0.27	0.122	1.72	100	0.00
10	6.5	11.2	5.6	6	6	3	0.32	0.242	1.47	100	0.00
11	6.2	8.4	5.8	6	6	4	0.34	0.119	1.38	100	0.00
12	5.9	6.1	5.5	6	6	4	0.35	0.032	1.35	100	0.00
13	6.0	6.3	6.0	6	6	6	0.38	0.015	1.33	100	0.00
14	6.0	6.0	5.9	6	6	6	0.41	0.005	1.33	100	0.00
15	6.0	6.0	5.9	6	6	6	0.43	0.004	1.33	100	0.00

Figure 3 presents a typical dynamics of the CCE Algorithm 6.1 for the random 3-SAT with an instance matrix $A = (25 \times 100)$ and $N = 50,000$.

Figure 3: Typical dynamics of the CCE Algorithm 6.1.



Tables 12 and 13 presents the performance of the CCE Algorithm 6.1 for random 3-SAT problems with an instance matrices $A = (40 \times 160)$ and $A = (75 \times 325)$, respectively for $N = 100,000$. The last table was taken from the SATLIB website www.satlib.org. The results are self-explanatory again. We also run both problems using the depth-1 SME Algorithm 4.3 and got again very similar results. The standard CE have failed to produce meaningful results in both cases.

Table 12: Performance of the CCE Algorithm 6.1 for the random 3-SAT for the instance matrix $A = (40 \times 160)$ and $N = 100,000$.

t	$ \mathcal{X}^* $			<i>Found</i>			PV	RE	S	m
	Mean	Max	Min	Mean	Max	Min				
0	0.0	0.0	0.0	0	0	0	0.00	NaN	13.86	151
1	0.0	0.0	0.0	0	0	0	0.00	NaN	13.25	153
2	96.0	960.3	0.0	0	1	0	0.00	3.000	12.28	155
3	88.9	328.6	0.0	2	9	0	0.00	1.164	11.13	157
4	93.9	120.1	0.0	42	106	0	0.00	0.387	8.46	159
5	111.0	134.6	45.9	98	113	12	0.05	0.207	6.52	160
6	113.0	123.1	105.9	109	113	98	0.22	0.038	4.49	160
7	109.5	113.3	104.5	109	111	105	0.38	0.025	3.55	160
8	109.7	113.5	105.1	109	111	105	0.49	0.021	3.16	160
9	109.9	114.6	104.5	109	111	105	0.53	0.025	3.04	160
10	111.7	116.9	104.8	109	111	105	0.54	0.027	3.00	160

Table 13: Performance of the CCE Algorithm 6.1 for the random 3-SAT for the instance matrix $A = (75 \times 325)$ with $N = 100,000$

t	$ \mathcal{X}^* $			<i>Found</i>			PV	RE	S	m
	Mean	Max	Min	Mean	Max	Min				
0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	NaN	25.99	301
2	0.0	0.0	0.0	0.0	0.0	0.0	0.00	NaN	25.06	307
4	0.0	0.0	0.0	0.0	0.0	0.0	0.00	NaN	24.70	308
6	0.0	0.0	0.0	0.0	0.0	0.0	0.00	NaN	23.68	309
8	0.0	0.0	0.0	0.0	0.0	0.0	0.00	NaN	22.65	312
10	0.0	0.0	0.0	0.0	0.0	0.0	0.00	NaN	21.22	314
12	0.0	0.0	0.0	0.0	0.0	0.0	0.00	NaN	19.89	316
14	773.8	7738.3	0.0	0.1	1.0	0.0	0.00	3.000	17.59	320
16	359.9	2243.4	0.0	0.5	3.0	0.0	0.00	2.075	13.60	323
18	787.9	2601.6	0.0	127.0	988.0	0.0	0.01	1.054	11.14	324
20	1583.2	7507.5	0.0	383.2	1265.0	0.0	0.05	1.285	9.43	325
22	1752.5	6306.7	560.6	720.6	1278.0	3.0	0.11	0.888	6.16	325
24	1424.1	1685.0	1320.1	1106.5	1305.0	79.0	0.20	0.071	5.62	325
26	1403.5	1470.5	1326.5	1252.7	1285.0	1099.0	0.26	0.030	5.26	325
28	1388.4	1457.5	1351.4	1271.1	1281.0	1258.0	0.29	0.020	5.24	325
30	1397.4	1467.2	1347.4	1275.1	1288.0	1266.0	0.29	0.027	5.16	325

Figure 4 presents a typical dynamics of the CCE Algorithm 6.1 with the instance matrices $A = (40 \times 160)$.

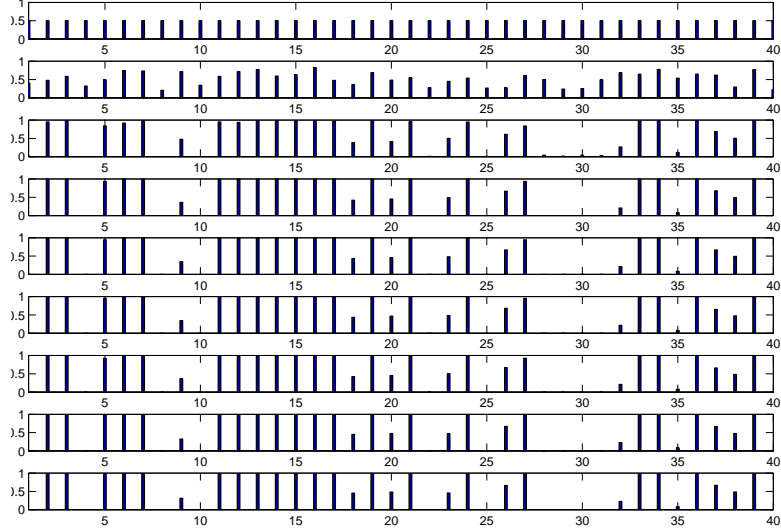
9.2.1 Counting the Number of Feasible Solutions in Constrained Optimization Problems

Set Covering, Set Packing and Set Partition

Table 15 presents the performance of the depth-1 SME Algorithm 4.3 for a benchmark set covering optimization problem with $N = 50,000$ using the weighted function approach with $r_1 = r_2 = 3$ for both constraints and the objective function. We set

1. The objective function being simply a sum of all variables.

Figure 4: Typical dynamics of the CCE Algorithm 6.1 for the random 3-SAT with an instance matrix $A = (40 \times 160)$ and $N = 100,000$



2. The constraints as $\mathbf{Ax} \geq \mathbf{b}$, where \mathbf{A} is a (20×20) matrix given in Table 14, each row of which contains 3 randomly generated 1's and the rest are 0's.
3. All elements of the vector \mathbf{b} equal to 2.

Figure 5 presents a typical dynamics of the SME Algorithm 4.3 for the random 3-SET problem with $N = 50,000$.

We call this model, the random 3-SET model to distinguish it from the random 3-SAT model. Using full enumeration we found that the total number of multiple extrema in our random 3-SET model equals 31. The results are self-explanatory again. We also run this problem using the CCE Algorithm 6.1 and got very similar results.

Table 14: (20×20) matrix

```

00100001000010000000
00000100001001001000
00100000010000010010
00010100100100000000
00000001000000000010
00010000000000000100
00101000100100000000
00001000000000000000
01001000000000000000
10000001001001000010
00000000000000011000
11000000001010000000
00000000000000001100
00000010000000000001
00000000000010011000
00000100010000000000
00000000000000000000
00000010000000000101
01010010010000000000
00000000100011101001

```

Table 15: Performance of the depth-1 SME Algorithm 4.3 for the set covering problem with $N = 50,000$ using the weighted function approach with $r = 3$.

t	$ \mathcal{X}^* $			<i>Found</i>			PV	RE	S	m
	Mean	Max	Min	Mean	Max	Min				
0	23.1	62.9	0.0	0	0	0	0.00	1.033	6.93	19
1	28.7	31.7	26.1	19	25	11	0.02	0.064	5.12	20
2	31.4	49.2	17.8	28	31	16	0.03	0.266	4.29	21
3	30.0	32.5	26.6	30	31	23	0.03	0.060	3.86	21
4	32.0	34.0	30.5	31	31	30	0.03	0.036	3.66	21
5	30.8	32.5	29.7	31	31	31	0.03	0.027	3.56	21
6	30.4	31.7	29.3	31	31	30	0.03	0.028	3.53	21

Knapsack Problem

Table 16 presents the performance of the CCE Algorithm 6.1 for the knapsack problem with the instance matrix $A = (20 \times 11)$ and $N = 10,000$ using the weighted function approach with $r = 3$. This problem was taken from the website <http://elib.zib.de>. Using full enumeration we found that the total number of multiple extrema is 612. One can see that Algorithm 6.1 performs quite well.

Figure 5: Typical dynamics of the SME Algorithm 4.3 for the random 3-SET problem with $N = 50,000$.

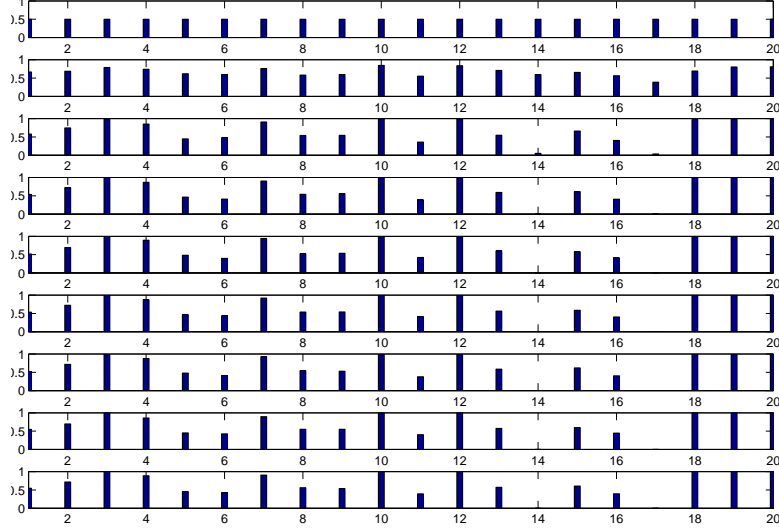
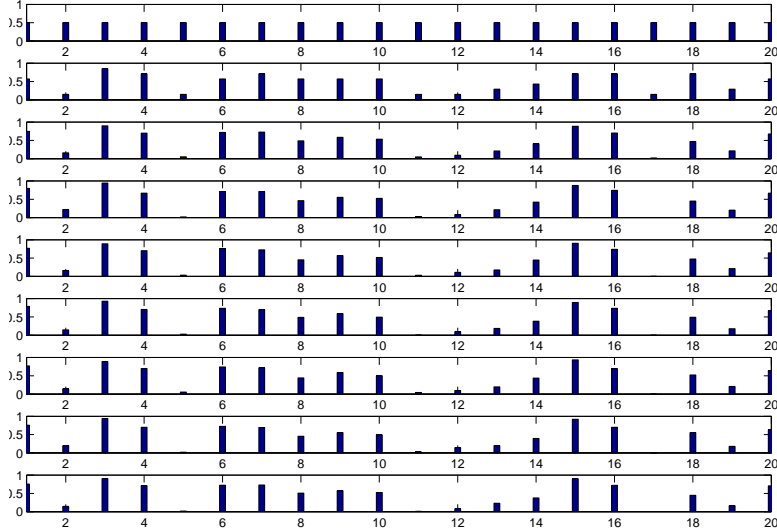


Table 16: Performance of the CCE Algorithm 6.1 for the knapsack problem with the instance matrix $A = (20 \times 11)$ and $N = 10,000$

t	Mean	Max	Min	PV	RE	S	m	RD_c
0	639.6	943.7	419.4	0.00	0.225	6.93	10	55.73
1	619.2	697.6	564.8	0.03	0.072	5.78	11	0.02
2	630.8	706.5	557.0	0.07	0.059	5.18	11	0.03
3	628.1	698.1	533.2	0.08	0.083	4.95	11	0.03
4	573.7	671.2	504.9	0.09	0.083	4.88	11	0.06
5	599.3	719.6	525.7	0.09	0.100	4.72	11	0.03
6	576.9	646.4	508.0	0.09	0.071	4.76	11	0.06
7	611.4	802.2	531.0	0.10	0.119	4.72	11	0.13
8	628.1	862.0	557.1	0.09	0.139	4.62	11	0.09
9	591.6	707.3	507.6	0.09	0.086	4.57	11	0.01
10	637.8	754.8	562.1	0.09	0.102	4.66	11	0.10
11	602.5	702.7	536.1	0.09	0.082	4.79	11	0.12
12	600.3	683.9	543.9	0.09	0.064	4.71	11	0.02
13	602.5	729.1	494.8	0.09	0.126	4.75	11	0.06
14	583.3	613.0	515.9	0.10	0.047	4.66	11	0.02
15	596.7	676.9	540.0	0.10	0.075	4.57	11	0.12

Figure 6 presents a typical dynamics of the CCE Algorithm 6.1 for the knapsack problem with the instance matrix $A = (20 \times 11)$ and $N = 10,000$.

Figure 6: Typical dynamics of the CCE Algorithm 6.1 for the knapsack problem with the instance matrix $A = (20 \times 11)$ and $N = 10,000$



9.2.2 “Honesty” of the Algorithms

While performing different simulation experiments with the proposed algorithms for counting and rare-event we found that the CE-based ones, like SME, CCE and CE generate from time to time (in about 5-10%) incorrect estimators. For some instances the error might reach 100% or even more. We call such instances, the “pathological” ones. Further more, while performing experiments with such “pathological” models, we found that the CE-based algorithms produce seemingly “stable, low variance”, estimators. But as we mentioned such seemingly “stable, low variance” estimators might be misleading. Based on this we came to conclusion that there exist quite a large set of randomly generated instances (models) for which cross-entropy-based algorithms are not robust, or simply fail. Unfortunately, we do not have yet a clear mechanism to distinguish between a “good” and a “pathological” models.

In contrast, the VM-based counting algorithms always produces a stable estimator.

Returning to the issue of robustness of the CE-based and VM-based counting algorithms we found based on extensive numerical results the following:

1. The lack of robustness of CE-based algorithms occurs for instances where the true counting quantity $|\mathcal{X}^*|$ is very small, like $1 \leq |\mathcal{X}^*| \leq 20$. To observe this phenomenon we generated a set of random 3-SAT models with the instance matrices 20×80 and selected (using full enumeration) only these for which $1 \leq |\mathcal{X}^*| \leq 20$. We find that there are about 5-10% such “pathological” models, for which all CE-based algorithms generates “bad” estimators and thus, fail.
2. Although VM-based algorithms always produce statistically sound estimators, the resulting estimators for “pathological” models have intractable (very high) relative error. In particular, we found that it does not matter how many

iteration one performs with VM, the resulting \mathbf{p} will be close to the original \mathbf{u} . In short, for “pathological” models the VM-based algorithms perform similar to CMC ones. Thus, in order to get a relative error say of 10% for a “pathological” model using VM, one needs to take a sample size of order of the size of the entire space $|\mathcal{X}|$. This is useless, of course.

To provide more insight on this, consider, for example, counting 0-1-Tables with a $m \times n$ matrix. Assume that $n = m$ and that the margins in all rows and columns are equal to $m/2$. It is not difficult to see that in this case the optimal $\mathbf{p} = \mathbf{u}$ and that any VM-based algorithm will be not able to move from \mathbf{u} , which is exactly what it should be. In contrast the CE-based algorithms will typically move from \mathbf{u} to some incorrect value of $\mathbf{p} \neq \mathbf{u}$ and thus will deliver a wrong final estimate of $|\mathcal{X}^*|$.

It follows from the above that the crucial difference between VM-based and CE-based counting algorithms is that the former can identify a “pathological” model, while the latter can not. Such identification by VM can be done by several approaches, say based on the dynamics of the relative error or based on the dynamics of $\hat{\mathbf{p}}_t$. For instance, VM can declare a “pathological” model if the relative error remains very high say during first 10-15 iterations, or alternatively, if most of the components of the optimal $\hat{\mathbf{p}}_t$ are remain close to \mathbf{u} , as t decreases. We call such property of VM, the *honesty of the VM algorithm*.

Below we provide more details on the honesty of the VM-based algorithms and the “dishonesty” of CE-based ones by discussing a specific example.

Example 9.1 The following data concerns one of our “pathological” models we have generated (among many good ones). In particular it concerns a random 3-SAT 20×80 instance matrix for which using full enumeration we found $|\mathcal{X}^*| = 15$. Also, using the deterministic CE updating formula (55) (via full enumeration) we obtained for that particular case the optimal 20-dimensional vector \mathbf{p} , most of the components of which were close to 0.5. This is exactly a type of a “pathological” model, since $\mathbf{p} \approx \mathbf{u}$ and no parametric version is able to move “far away” from $\mathbf{u} = (0.5, \dots, 0.5)$.

Indeed, as expected, taking a huge sample of $N = 10,000,000$ we obtained after 10 iteration with VM algorithm an estimator $\hat{\mathbf{p}}$ very close to the true \mathbf{p} . Taking next another sample of $N = 10,000,000$, the resulting estimator (24) delivered $|\hat{\mathcal{X}}^*| = 15$. Smaller samples, like $N = 1,000,000$ produced estimators with high relative error of both, $|\hat{\mathcal{X}}^*|$ and of $\hat{\mathbf{p}}$. Note also that in the last case most of the components of $\hat{\mathbf{p}}$ have been fluctuating around the value 0.5. Clearly, because the true \mathbf{p} is close to \mathbf{u} , VM performs similar to CMC, that is, both are useless. On the positive side of VM: *it is, at least, “honest”*.

While running CE-based algorithms, like CE and SME, we obtained a completely different picture as far as the estimators $\hat{\mathbf{p}}$ and $|\hat{\mathcal{X}}^*|$ are concerned. In particular, starting from $N = 10,000$ and up, most of the components of $\hat{\mathbf{p}}$ have degenerated either to 0 or to 1. As result, we obtained a “stable” (low variance, but wrong) estimator $|\hat{\mathcal{X}}^*| = 9$ instead of the true one $|\mathcal{X}^*| = 15$.

As a final remark, note that we found numerically (for small models), that if most of the components of the optimal vector \mathbf{p} are quite different from \mathbf{u} (close to degenerated ones, 0’s or 1’s), both CE-based and VM-based algorithms produce stable and accurate estimators (good model), but if most of the components of \mathbf{p} are close to \mathbf{u} , we have a “pathological” model with all the consequences as per above. Clearly, since the optimal vector \mathbf{p} we are looking for is unknown, we have to rely on the “honesty” of the VM-based and not on the CE-based algorithms.

We shall explain now why the CE-based algorithms might behave differently from the VM-based one by arguing that the resulting \mathbf{p} updating can be quite different.

Indeed, consider the following two basic CE and VM optimization programs [21]

$$\max_{\mathbf{p}} \mathcal{K}(\mathbf{p}) = \max_{\mathbf{p}} \sum_{k=1}^N I_{\{S(\mathbf{x}_k) \geq b\}} W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1}) \ln f(\mathbf{X}_k, \mathbf{p}) \quad (114)$$

and

$$\min_{\mathbf{p}} \mathcal{V}(\mathbf{p}) = \min_{\mathbf{p}} \sum_{k=1}^N I_{\{S(\mathbf{x}_k) \geq b\}} W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1}) W(\mathbf{X}_k, \mathbf{u}, \mathbf{p}), \quad (115)$$

respectively.

The corresponding gradient of $\mathcal{K}(\mathbf{p})$ and $\mathcal{V}(\mathbf{p})$ are

$$\nabla \mathcal{K}(\mathbf{p}) = \sum_{k=1}^N I_{\{S(\mathbf{x}_k) \geq b\}} W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1}) \nabla \ln f(\mathbf{X}_k, \mathbf{p}) \quad (116)$$

and

$$\nabla \mathcal{V}(\mathbf{p}) = \sum_{k=1}^N I_{\{S(\mathbf{x}_k) \geq b\}} W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1}) W(\mathbf{X}_k, \mathbf{u}, \mathbf{p}) \nabla \ln f(\mathbf{X}_k, \mathbf{p}), \quad (117)$$

respectively. Comparing (116) and (117) it readily follows that the LR part $W(\mathbf{X}_k, \mathbf{u}, \mathbf{p})$ is *missing* in the gradient $\nabla \mathcal{K}(\mathbf{p})$. Thus, the gradient $\nabla \mathcal{K}(\mathbf{p})$ based on the cross-entropy presents only an approximation of gradient counterpart based on variance minimization. It is well known that the latter one is the true one, presenting the core of most Monte Carlo experiments. Since the LR $W(\mathbf{X}_k, \mathbf{u}, \mathbf{p})$ is an essential part in $\mathcal{V}(\mathbf{p})$ and $\nabla \mathcal{V}(\mathbf{p})$, the CE and VM updatings of \mathbf{p} (based on (116) and (117)) can be quite different, and this is the reason that CE-based algorithms might generate misleading result.

Note finally that our numerical results suggests that the VM algorithm based on the program (115), while *omitting* the LR term $W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1})$ in (117), that is, using the VM algorithm based on the program

$$\min_{\mathbf{p}} \mathcal{V}^{(-)}(\mathbf{p}) = \min_{\mathbf{p}} \sum_{k=1}^N I_{\{S(\mathbf{x}_k) \geq b\}} W(\mathbf{X}_k, \mathbf{u}, \mathbf{p}), \quad (118)$$

performs similarly (and some time even better) than the standard CE based on the original program (114), that is, the one *containing* the LR term $W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1})$. This phenomenon might be explained by comparing the gradients $\nabla \mathcal{K}(\mathbf{p})$ in (116) and $\nabla \mathcal{V}^{(-)}(\mathbf{p})$, that is,

$$\nabla \mathcal{V}^{(-)}(\mathbf{p}) = \sum_{k=1}^N I_{\{S(\mathbf{x}_k) \geq b\}} W(\mathbf{X}_k, \mathbf{u}, \mathbf{p}) \nabla \ln f(\mathbf{X}_k, \mathbf{p}). \quad (119)$$

Indeed, it is readily seen that both gradients $\nabla \mathcal{K}(\mathbf{p})$ and $\nabla \mathcal{V}^{(-)}(\mathbf{p})$ are similar in the sense that they differ only in their LR terms; the former contains $W(\mathbf{X}_k; \mathbf{u}, \hat{\mathbf{p}}_{t-1})$, but it misses the part $W(\mathbf{X}_k, \mathbf{u}, \mathbf{p})$, while the latter vice-versa. Note again that the bias introduced while omitting a LR terms is often less than having a large variance which is based on both LR terms.

Note also that our numerical experiments suggest that even the standard VM algorithm based on the truncated gradient (119) is still typically "honest" as its original VM program (115) is.

Note finally that more research is needed in order to understand why without the LR term $W(\mathbf{X}_k, \mathbf{u}, \mathbf{p})$ the CE-based methods still perform well for "good" (not "pathological") models.

9.3 Rare Events

Here we present comparative studies of the standard CE and VM algorithms with the view to show higher efficiency (accuracy) of VM. We consider a network with 20 nodes from [9] depicted in Figure 9.3. In particular, we consider estimation of the rare-event probability $\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X}) \geq \gamma\}}]$ with the performance $S(\mathbf{X})$ being the shortest path from node 1 to 20, $\mathbf{X} \sim \exp(\mathbf{u})$, where \mathbf{u} and γ are fixed in advance. Note that using full enumeration we found that the total number of feasible path in the network equals 830.

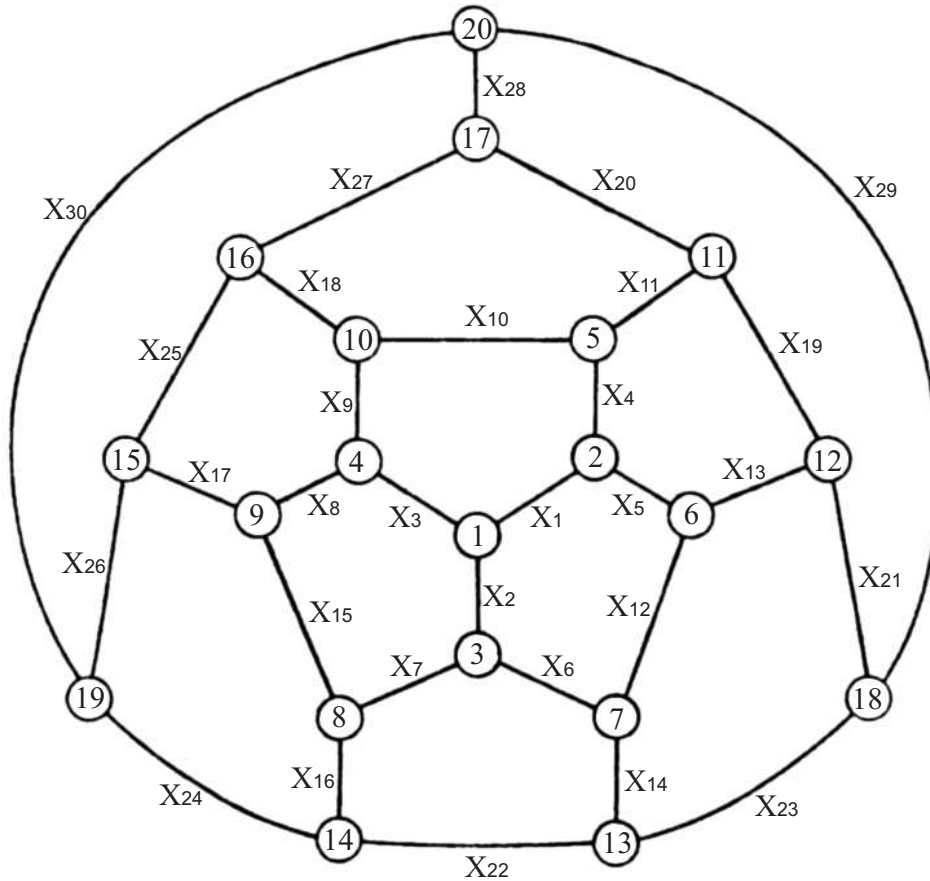


Figure 7: A network.

Table 17 presents the performance of the standard CE and VM algorithms the model in Fig.9.3 with $\gamma = 10$, $N = 50,000$ and $N_1 = 100,000$. Here Mean T denotes the mean number of iterations averaged over 10 replications, N and N_1 denote the sample size for estimating the optimal parameter vector \mathbf{v} in $\exp(\mathbf{v})$. We set all 30 parameters equal to 1 and selected $\rho = 0.01$ and $\alpha = 0.7$.

Table 17: Performance of CE and VM with equal initial parameters, $\gamma = 10$, $N = 50,000$, $N_1 = 100,000$.

Method		CE	VM
$\hat{\ell}$	Mean	1.233E-08	1.234E-08
	Max	2.076E-08	1.629E-08
	Min	8.640E-09	8.997E-09
RE		0.33293	0.20606
Mean T		7.0	8.1
Mean CPU		36.32	45.44

It follows from above that both approaches perform similarly. While updating the parameter vector \mathbf{v} we found that 6 among 30 elements of \mathbf{v} have change the most. These elements corresponds to the parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$ According to [22], such elements are called *the bottleneck* ones. Taking this into consideration we kept the 6 bottlenecks parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$ equal unity, while we increased the remaining (non-bottleneck) ones.

Table 18 presents data for such a case. In particular it presents data similar to Table 17 but with Weib($\alpha, u^{-1/\alpha}$) pdf instead of $\exp(u)$ pdf. As before we assume that only u is controllable, while all α 's are equal to $1/4$. In addition, we set $u_1 = u_2 = u_3 = u_{28} = u_{29} = u_{30} = 1$, while the remaining 24 ones we set equal to 4.

Table 18: Performance of CE and VM with bottleneck elements, $\gamma = 2000$, $N = 100,000$, $N_1 = 300,000$.

Method		CE	VM
$\hat{\ell}$	Mean	7.415E-09	4.013E-09
	Max	4.628E-08	6.138E-09
	Min	9.744E-11	2.784E-09
RE		1.85189	0.24994
Mean T		9.6	12.6
Mean CPU		109.85	260.44

One can clearly see that for this case (with bottleneck parameters) VM outperforms substantially CE.

We next consider the some model in Figure 9.3 but with Bernoulli random variables.

Table 19 presents performance of CE and VM algorithms with all equal initial parameters $u = 0.999$. We set $N = 10,000$, $N_1 = 50,000$, $\alpha = 0.7$, $\rho = 0.01$.

Table 19: Performance of CE and with equal initial Bernoulli parameters $u = 0.999$, $N = 10,000$, $N_1 = 50,000$.

Method		CE	VM
$\hat{\ell}$	Mean	2.232E-6	2.073E-6
	Max	3.858E-6	2.184E-6
	Min	1.008E-6	1.996E-6
RE		0.45903	0.03720
Mean T		8.0	8.0
Mean CPU		8.18	27.96

Table 20 presents data similar to Table 19, where we set the 6 bottleneck parameters $u_1, u_2, u_3, u_{28}, u_{29}, u_{30}$ to 0.97, while we keep the remaining 24 parameters equal to 0.999.

Table 20: Performance of CE and VM with bottlenecks Bernoulli elements $N = 10,000, N_1 = 50,000$.

Method		CE	VM
$\hat{\ell}$	Mean	4.353E-5	5.436E-5
	Max	5.527E-5	5.593E-5
	Min	2.700E-5	5.233E-5
RE		0.32695	0.02323
Mean T		8.0	8.0 0
Mean CPU		11.93	16.18

It follows from the results of Table 19 and Table 20 that in both cases VM outperforms CE.

9.4 Optimization

In this section we present performance of CE, VM, MinxEnt and IME for unconstrained optimization. Constrained optimization will be considered somewhere else.

Table (21) presents comparative studies of the following 4 methods: CE, VM, MinxEnt and IME for a couple of TSP models taken from <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/atsp/>

In all numerical results we use the same CE parameters as for the `ft53` problem, that is, $\rho = 10^{-2}$, $N = 10n^2$, $\alpha = 0.7$ and $d = 5$ (see (96)). To study the variability in the solutions, each problem was repeated 10 times. In Table 21, n denotes the number of nodes of the graph, \bar{T} denotes the average total number of iterations needed before stopping, \hat{b}_1 and \hat{b}_T denote the average initial and final estimates of the optimal solution, b^* denotes the best known solution, $\bar{\varepsilon}$ denotes the average relative experimental error based on 10 replications, ε_* and ε^* denote the smallest and the largest relative error among the 10 generated shortest paths, and finally CPU denotes the average CPU time in seconds.

Table 21: Comparative studies for TSP.

file	n	b^*	Alg.	\hat{b}_1	\hat{b}_T	$\bar{\varepsilon}$	ε_*	ε^*	\bar{T}	CPU
ftv33	34	1286	CE	3248.2	1333	0.0365	0.0000	0.0684	17.8	56.59
			VM	3366.4	1286	0.0000	0.0000	0.0000	23.8	127.60
			PME	3296.7	1308	0.0171	0.0000	0.0412	19.8	173.83
			IME		1305.8	0.0154	0.0000	0.0435	18.30	76.77
ry48p	48	14422	CE	40254.9	14840	0.0289	0.0133	0.0579	31.2	424.88
			VM	42208.1	14960.7	0.0373	0.0162	0.0597	61.7	935.29
			PME	41041.7	14952	0.0367	0.0228	0.0537	34.0	992.51
			IME		14888.7	0.0323	0.0160	0.0461	30.60	731.60

It follows that all 4 methods work reasonable well and it is difficult to give priority to any of them.

The RSA problem

Since the RSA problem has only one optimal solution we regard as an optimization one in the sense that no LR term are involved while updating the parameter vector \mathbf{p} . Like in TSP we run the RSA problem with CE, VM, MinxEnt and IME. We found that all four methods, produce accurate estimators (find both prime numbers) for the RSA models containing up to 100 binary digits. More research is under way with the view to find the prime numbers for large RSA models while using pair wise dependent and some other enhancements.

10 Conclusion and Further Research

In this paper we presented a new generic minimum cross-entropy method, called SME for rare-event probability estimation, counting, and optimization. The main idea of SME approach is to associate with each original problem an auxiliary *single-constrained* convex MinxEnt program of special type, which has a closed form solution. We proved that the optimal pdf $g(\mathbf{x})$ obtained from the solution of this MinxEnt program is a zero variance pdf, provided the "temperature" parameter λ is set to minus infinity. For some particular instances we showed how to approximate the optimal zero variance pdf by a normal pdf using a central limit theorem. In addition we proved that the parametric pdf $f(\mathbf{x}, \mathbf{p})$ based on the product of marginals of the optimal zero variance pdf $g(\mathbf{x})$ coincide with the parametric pdf of the standard CE method. A remarkable feature, which we discovered in this paper is that originally designed at the end of 1990-th as a heuristics for estimation of rare-events and COP's, CE has strong connection with the proposed MinxEnt and, thus strong mathematical foundation.

The crucial difference between the proposed SME method and its standard CE counterparts is in their simulation-based versions: in the latter we always require to generate (via Monte Carlo) a sequence of tuples including the temperature parameter and the parameter vector in the optimal marginal pdf's, while in the former we can **fix** in advance the temperature parameter (to be set a large negative number) and then generate (via Monte Carlo) a sequence of parameter vectors of the optimal marginal pdf's only. In addition, in contrast to CE, neither the elite sample nor the rarity parameter is needed in SME. As a result, the proposed SME Algorithm becomes simpler, faster and at least as accurate as the standard CE.

Motivated by the SME method we have introduced a new updating rule for the parameter vector \mathbf{p} in the standard cross-entropy method, called the CCE updating.

We showed numerically, that

1. Typically CCE and depth-1 SME algorithm produce quite accurate estimators of \mathbf{p} . In particular, we found numerically that they allow quite accurate approximation of counting problems with up to one hundred of decision variables and several hundreds of constraints.
2. In order to get accurate counting estimators with the SME and CCE Algorithms it is highly advisable to use its weighted functions $\mathcal{C}(\mathbf{x})$, version, that is, the ones defined in (68) and (70) instead of their indicator one $I_{\{\mathcal{C}(\mathbf{x})=m\}}$.
3. We found that the VM-based algorithm based on the solution of the variance minimization problem are more robust as compared to their CE-based counterparts. In spite of this fact we still recommend using (for large models) the CE-based algorithms. The main reason is that CE-based algorithms can typically update the vector \mathbf{p} analytically, rather than numerically and as result, they are approximately two times faster. Before applying a CE-based algorithm we do, however, suggest to make a short pilot run using the standard VM algorithm (say, even without using the LR term $W(\mathbf{X}, \mathbf{u}, \mathbf{p})$) to decide whether or not the model is “pathological”. If the answer is - YES, then apply the ”honesty” principle and quit (stop); if - NO, then run the model with CCE or SME.

Back to the Roots

While performing different simulation experiments with the proposed algorithms for counting and rare-event we found that all CE -based ones, like SME and CCE generate occasionally “bad“ estimators. For some instances the error reached 100% or even more. We call such instances, which present a non-neglegable set of problems, the “pathological” ones. Furthermore, while performing experiments with such “pathological” models, we found that the CE-based algorithms produce seemingly “stable, low variance”, estimators. But as we mentioned such estimators might be quite misleading. In contrast, the VM-based counting algorithms always produces statistically sound (unbiased) estimator. Our explanation for this follows from the comparison of formulas (114)-(117). It is due to the fact that the LR function $W(\mathbf{X}, \mathbf{u}, \mathbf{p})$, which is an essential part of the VM-based algorithms is ”missing” in the CE-based counterparts.

All this can be summarized as follows:

Introduced in 1997 [15] for rare-event simulation, the original VM algorithm was replaced almost immediately by its CE-counterpart. The main reason for that the CE program (114) can be solved analytically as compared to the VM program (115), which needs to be solved numerically and, thus is little bit more time consuming. Another reason is that CE appears to be a very efficient method in optimization, since in optimization there is no need to use LR terms at all.

The discovery of this paper on the ”honesty” of VM-based algorithms, (and in particular on the ”missing” LR term $W(\mathbf{X}, \mathbf{u}, \mathbf{p})$ in CE), brings us back to the origins, namely to the original VM algorithm [15]. However, as mentioned, in spite of this fact we still recommend using the convenient CE-based algorithms, by making first a simple pilot run with the standard VM algorithm to find out whether or not the underlying model is “pathological”.

Further Research

The entire area of counting using randomized algorithms and in particular the ones based on MinxEnt, CE and VM are still in its infancy.

As for further research for **counting** we intend to consider the following issues:

1. Develop fast VM-based algorithms for counting.
2. Use the large deviation theory to prove polynomial convergence and speed of convergence of the SME Algorithm 4.3 for rare-event probability estimation and thus, for estimation of the counting quantity $|\mathcal{X}^*|$ according to (24).
3. Apply the above counting algorithms to a broad variety of counting problems, like Hamiltonian cycles, counting 0-1-Tables, self-avoiding walks, counting problems associated with graph coloring, cliques and counting the number of multiple extrema in a multi-extremal function.
4. Although we obtained some preliminary encouraging results while using pairwise dependence, much more work is needed before it might be recommended in practice.
5. Investigate the issue of non robustness of CE with the view to generate robust CE algorithms for rare-events and counting, similar as VM does.
6. Apply the dynamic programming approach of [6] for efficient generation from the pdf $g(\mathbf{x})$ in (38).
7. Consider the program (problem of moments) (64) as an alternative to the MinxEnt program (4), (5).

Although CE and MinxEnt have been successfully applied to many unconstrained combinatorial optimization problems [21], their success to *constrained optimization* (both combinatorial and integer) is still very limited. To the best of our knowledge, the only alternative is the penalty function approach [21].

Finally

1. An interesting and challenging issue is to develop efficient CE and MCE based classification algorithms competitive with the well known *boosting* algorithms.
2. Find the relation ship between the MinxEnt based method for optimization and the method of the Arora et all [2] called, *The Multiplicative Weights Update Method*. The latter is based Lagrange relaxation and involves Kullback-Leibler's cross-entropy.

Acknowledgments

I would like to thank Ido Leichter for replacing the former proofs of Lemmas 3.1-3.3 by less cumbersome ones, Alexander Shapiro for introducing me to the problem of moments, Dirk Kroese, Thomas Taimre, Zdravko Botev and Add Ridder for insightful correspondence and valuable remarks on the earlier draft of the paper, and to Andrey Dolgin and Dmitry Lifshitz for performing the computational part of the paper.

References

- [1] Aarts E. H. L. and J. H. M. Korst, *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons, 1989.
- [2] Arora S., Hazan E. and S. Kale, "The Multiplicative Weights Update Method: A Meta Algorithm and Applications. Manuscript, Princeton University.
- [3] Ben-Tal A. and M. Taboule, "Penalty Functions and Duality in Stochastic Programming via ϕ -Divergence Functionals." *Mathematics of Operations Research*, Vol 12, No2, pp 224-240, 1987.

- [4] Botev, Z.I, Kroese, D.P., Taimre, T. Generalized Cross-Entropy Methods. Proceedings of RESIM06, 1-30, 2006.
- [5] Cover T.M. and Thomas J.A., *Elements of Information Theory*, John Wiley & Sons, inc, 1991.
- [6] Ghate A. and Smith R.L. “ A Dynamic Programming Approach to Efficient Sampling from Boltzmann Distribution”, (submitted for publication)
- [7] Gomes C. P. and Selman B., “Satisfied with Physics” *Science*, Vol. 297, pp 784-785, 2002.
- [8] Homem-de-Mello T., ”A Study on the Cross-Entropy Method for Rare Event Probability Estimation”, *INFORMS Journal on Computing*, Vol. 19, No. 3, 381-394, 2007.
- [9] G. S. Fishman, *Discrete Event Simulation: Modeling, Programming, and Analysis*, Springer-Verlag, 2001.
- [10] Kapur J.N. and H.K. Kesavan, *Entropy Optimization with Applications*, Academic Press, Inc., 1992.
- [11] M. Mitzenmacher and E. Upfal. *Probability and Computing : Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York (NY), 2005.
- [12] Motwani R. and R. Raghavan. *Randomized Algorithms* Cambridge University Press, 1997.
- [13] Mezard. M and M. Momtarnari. *Constrained Satisfaction Networks in Physics and Computations: Probabilistic Approaches*. Oxford Press, 2006.
- [14] Pincus, M. A., “ A closed form selection of certain programming problems” *Oper., Res*, **16**, 690-694, 1968.
- [15] Rubinstein, R. Y., “ Optimization of Computer simulation Models with Rare Events”, *European Journal of Operations Research*, **99**, 89-112, 1997.
- [16] Rubinstein, R.Y., “The cross-entropy method for combinatorial and continuous optimization”, *Methodology and Computing in Applied Probability* **2**, 127–190, 1999.
- [17] R. Y. Rubinstein “ A Stochastic Minimum Cross-Entropy Method for Combinatorial Optimization and Rare-event Estimation”, *Methodology and Computing in Applied Probability*, No 1, pp 1-46, 2005.
- [18] R. Y. Rubinstein “How Many Needles Are in a Haystack, or How to Solve #P-Complete Counting Problems ”, *Methodology and Computing in Applied Probability*, No 1, pp 1-42, 2007.
- [19] R. Y. Rubinstein, D. P. Kroese, Dolgin A., and P. W. Glynn, “Parametric Minimum Cross-Entropy Method for Counting the Number of Satisfiability Assignments”, Manuscript, Technion, Israel.
- [20] Rubinstein, R..Y. and Shapiro, A., *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization*, John Wiley and Sons Inc, 1993.
- [21] Rubinstein R.Y. and D.P. Kroese, *The Cross-Entropy Method: a Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*, Springer, 2004.

- [22] Rubinstein R.Y. and D.P. Kroese, *Simulation and the Monte Carlo Method; Second Edition*, Wiley, 2007.
- [23] Shapiro, A., *Stochastic Programming, Handbook in Operations Research and Management Science*, edited by Ruszczyński, A. and Shapiro, A., Elsevier, 2003.
- [24] Wolsey L. A., *Integer Programming*, Wiley, 1998.