A Stochastic Minimum Cross-Entropy Method for Combinatorial Optimization and Rare-event Estimation

R.Y. Rubinstein

Faculty of Industrial Engineering and Management,
Technion, Haifa, Israel;
ierrr01@ie.technion.ac.il;
iew3.technion.ac.il:8080/ierrr01.phtml

March 29, 2005

Abstract

We present a new method, called the minimum cross-entropy (MCE) method for approximating the optimal solution of NP-hard combinatorial optimization problems and rare-event probability estimation, which can be viewed as an alternative to the standard cross entropy (CE) method. The MCE method presents a generic adaptive stochastic version of Kullback's classic MinxEnt method. We discuss its similarities and differences with the standard cross-entropy (CE) method and prove its convergence. We show numerically that MCE is a little more accurate than CE, but at the same time a little slower than CE. We also present a new method for trajectory generation for TSP and some related problems. We finally give some numerical results using MCE for rare-events probability estimation for simple static models, the maximal cut problem and the TSP, and point out some new areas of possible applications.

[0†] This research was supported by the Israel Science Foundation (grant No 191-565)

1 Introduction

In this paper we introduce a new generic method, called the *minimum cross-entropy* (MCE) method for approximating the optimal solution of NP-hard combinatorial optimization problems and estimation of probability of rare-event. The MCE method presents an adaptive stochastic version of Kullback's classic MinxEnt method and can be viewed as an alternative to the standard *cross entropy* (CE) method [3], [10]. We discuss similarities and differences between the standard CE and MCE. In particular, we show numerically that MCE is little bit more accurate than CE, but at the same time a little slower than CE. We also present a new method for trajectory generation for TSP and some related problems. We give numerical results with MCE for the maximal cut problem and the TSP, discuss its convergence, and point out some new areas of possible applications.

The main emphasis in this work will be made on application of MCE to combinatorial optimization problems (COP's). In particular, we shall deal with the following discrete optimization program

$$\max_{\boldsymbol{x}\in\mathcal{X}} S(\boldsymbol{x}), \tag{1}$$

where $\mathcal{X}$ is a set of states, which is typically assumed to be huge. For example, in the TSP (travelling salesman problem) with n cities the size of the space is $|\mathcal{X}| = (n-1)!$

Similar to the standard CE [10], which will be called simply *CE*, we shall

1. Associate with the original program (1) an *estimation problem*, called the *associated stochastic program* (ASP) introducing the randomness (distribution) in either
   (a) the *nodes* of the graph, in which case we speak of a *stochastic node network* (SNN), or
   (b) the *edges* of the graph, in which case we speak of a *stochastic edge network* (SEN).
2. Define for the ASP the following two phases:
   (a) **Trajectory Generation**: Trajectory generation for the MCE method is similar to the CE method.
   (b) **Parameter Updating**: Updating the parameters of the distribution in the ASP is different from CE. It is based on solving a sequence of stochastic Kullback's MinxEnt (minimum cross-entropy) programs [6], rather than on solving a sequence of cross-entropy programs involving estimation of rare-event probability [10].

There are many alternative generic randomized algorithm which successfully solve COPS, among them is simulated annealing [1] and genetic algorithm [4].

A closely related to MCE is the method on *Probability Collectives* (PC) pioneered by David Wolpert. The interested reader is refereed to [11] and references therein.

Sections 2 and 3 present some background on the CE and the MinxEnt methods, respectively. We also present in Section 3 a simple motivating MinxEnt program associated with die tossing. We extend this simple program in Section 4 to solve iteratively a sequence of associated *deterministic* MinxEnt programs. In Section 4 we also outline a related algorithm, which provides insight to our main MCE algorithm. In Section 5 we consider a sequence of *stochastic* MinxEnt programs and present our main MCE algorithm. We also discuss its relation to the CE algorithm. Section 6 deals with rare-event probability estimation using MCE. In Section 7 we present numerical results of the MCE algorithm for the max-cut problem and TSP. We show that MCE is little bit more accurate than CE, but at the same time a little slower than CE. We also present some preliminary numerical results with MCE for probability of rare-events estimation in static models. In Section 8 concluding remarks and some areas for further investigation are given. Finally, in Appendix we present a new method for trajectory generation for SEN models, some additional results on the MinxEnt program and prove the convergence of a variant of the MCE algorithm.

2 Background on the CE Method

Here we recapitulate briefly the CE method from [10]. For more details see the home page www.cemethod.org

Suppose we wish to solve (1). We first randomize the deterministic problem (1) by defining a family of probability density functions (pdf's) $\{f(\boldsymbol{x}, \boldsymbol{p}), \boldsymbol{p} \in \mathcal{V}\}$ on the set $\mathcal{X}$, with $\boldsymbol{p}$ being the pdf's parameter. We next associate with (1) the following estimation problem

$$\ell(\gamma) = \mathbb{P}_{\boldsymbol{u}}\{S(\boldsymbol{X}) \geq \gamma\} = \mathbb{E}_{\boldsymbol{u}} I_{\{S(\boldsymbol{X}) \geq \gamma\}}, \tag{2}$$

which is the ASP. Here, γ is a parameter (known or unknown) and $\boldsymbol{X}$ is a random vector with pdf $f(\boldsymbol{x}, \boldsymbol{u})$, for some $\boldsymbol{u} \in \mathcal{V}$. For example, $\boldsymbol{X}$ could be a normal random vector or an n-dimensional vector with independent components each having a discrete m-point probability mass function (pmf).

Having defined an ASP the goal of the CE Algorithm is to generate a sequence of tuples $\{\hat{\gamma}_t, \hat{\boldsymbol{p}}_t\}$, which converges to a small neighborhood of the optimal tuple $(\gamma^*, \boldsymbol{p}^*)$. More specifically, we initialize by setting $\boldsymbol{p}_0 = \boldsymbol{u}$, choosing a not very small *rarity parameter*, ρ, say $\rho = 10^{-2}$, and then we proceed iteratively as follows:

1. **Adaptive updating of** γ_t. Let γ_t be the $(1-\rho)$-quantile of $S(\boldsymbol{X})$ under $\boldsymbol{p}_{t-1}$. A simple estimator, denoted $\widehat{\gamma}_t$, of γ_t can be obtained by drawing a random sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from $f(\boldsymbol{x}, \boldsymbol{p}_{t-1})$, calculating the associated function values $S(\boldsymbol{X}_1), \ldots, S(\boldsymbol{X}_N)$ and their order statistics $S_{(1)}, \ldots, S_{(N)}$

and assigning $\widehat{\gamma}_t$ to be these order statistics' $(1-\rho)$-quantile, that is,

$$\widehat{\gamma}_t = S_{(\lfloor (1-\rho)N \rfloor + 1)}. \tag{3}$$

2. **Adaptive updating of $\boldsymbol{p}_t$.** For fixed γ_t and $\boldsymbol{p}_{t-1}$, derive $\boldsymbol{p}_t$ from the solution of the program

$$\max_{\boldsymbol{p}_t} D(\boldsymbol{p}_t) = \max_{\boldsymbol{p}_t} \mathbb{E}_{\boldsymbol{p}_{t-1}} I_{\{S(\boldsymbol{X}) \geq \gamma_t\}} \log f(\boldsymbol{x}, \boldsymbol{p}_t) \; . \tag{4}$$

The stochastic counterpart of (4) is as follows: for fixed $\widehat{\gamma}_t$ and $\widetilde{\boldsymbol{p}}_{t-1}$, derive $\widetilde{\boldsymbol{p}}_t$ from the following program

$$\max_{\widetilde{\boldsymbol{p}}_t} \widehat{D}(\widetilde{\boldsymbol{p}}_t) = \max_{\widetilde{\boldsymbol{p}}_t} \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \widehat{\gamma}_t\}} \; \log f(\boldsymbol{X}_i; \widetilde{\boldsymbol{p}}_t) \; . \tag{5}$$

Note that in (3) and (5) the same sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from $f(\boldsymbol{x}, \widetilde{\boldsymbol{p}}_{t-1})$ is used.

Solving (4) for a discrete n-dimensional pdf with independent components we obtain the following simple analytic expression for updating the parameters $p_{t,ij}, \; i = 1, \ldots, n; \; j = 1, \ldots, m$

$$p_{t,ij} = \frac{\mathbb{E}_{\boldsymbol{p}_{t-1}} I_{\{\boldsymbol{X}_i = j\}} I_{\{S(\boldsymbol{X}) \geq \gamma_t\}}}{\mathbb{E}_{\boldsymbol{p}_{t-1}} I_{\{S(\boldsymbol{X}) \geq \gamma_t\}}}. \tag{6}$$

The stochastic version of (6) obtained from the solution of (5) is

$$\widetilde{p}_{t,ij} = \frac{\sum_{k=1}^{N} I_{\{\boldsymbol{X}_{ki} = j\}} I_{\{S(\boldsymbol{X}_k) \geq \widehat{\gamma}_t\}}}{\sum_{k=1}^{N} I_{\{S(\boldsymbol{X}_k) \geq \widehat{\gamma}_t\}}}. \tag{7}$$

Instead of using the parameter vector $\widetilde{\boldsymbol{p}}_t$ obtained directly from (7) we use its following *smoothed* version

$$\overline{\boldsymbol{p}}_t = \alpha \widetilde{\boldsymbol{p}}_t + (1-\alpha) \widetilde{\boldsymbol{p}}_{t-1}, \tag{8}$$

where α, $(0 < \alpha < 1)$ is called the *smoothing parameter*. Clearly, for $\alpha = 1$ we have our original updating rule. The reason for using smoothed updating is to reduce the probability that some component of $\widetilde{\boldsymbol{p}}_t$ will be zeros or unities at the first few iterations, i.e. to prevent a fast convergence to a local optimum. Note that if $0 < \alpha < 1$, then $0 < \widetilde{p}_{t,ij} < 1 \; \forall \; k$, while for $\alpha = 1$ some components of $\widetilde{\boldsymbol{p}}_t$ might be zero or one even after the first iteration. In such a case, the algorithm will converge to a wrong solution.

It is crucial to note that the program (4) is solved under the assumption that the components of $f(\boldsymbol{x}, \boldsymbol{p})$ are *independent*, that is $f(\boldsymbol{x}, \boldsymbol{p}) = \prod_{s=1}^{n} f_s(x_s, \boldsymbol{p}_s)$. The same goes for program (5) and the components of $f(\boldsymbol{x}, \widetilde{\boldsymbol{p}}_{t-1})$. Clearly, solving (4) and (5) using dependent components in $f(\boldsymbol{x}, \boldsymbol{p})$ and $f(\boldsymbol{x}, \widetilde{\boldsymbol{p}}_{t-1})$ would yield updating rules different from (6) and (7).

For example, consider the programs (4) and (5), but assume pair-wise dependence between the components of $f(\boldsymbol{x}, \boldsymbol{p})$ and of $f(\boldsymbol{x}, \widetilde{\boldsymbol{p}}_{t-1})$, and consider a max-cut problem with n nodes. In this case, one would obtain an $n \times n$ matrix $\boldsymbol{P}_t$, whose elements present the joint probabilities of choosing simultaneously pairs of nodes, rather than choosing single nodes. The efficiency of this CE modifications is under investigation.

In [7] the following modifications of (7) have also been suggested

$$\widetilde{p}_{t,ij} = \frac{\sum_{k=1}^{N} I_{\{\boldsymbol{X}_{ki}=j\}} \varphi[S(\boldsymbol{X}_k, \eta)]}{\sum_{k=1}^{N} \varphi[S(\boldsymbol{X}_k, \eta)]}, \tag{9}$$

$$\widetilde{p}_{t,ij} = \frac{\sum_{k=1}^{N} I_{\{\boldsymbol{X}_{ki}=j\}} I_{\{S(\boldsymbol{X}_k) \geq \widehat{\gamma}_t\}} \varphi[S(\boldsymbol{X}_k, \eta)]}{\sum_{k=1}^{N} I_{\{S(\boldsymbol{X}_k) \geq \widehat{\gamma}_t\}} \varphi[S(\boldsymbol{X}_k, \eta)]}, \tag{10}$$

where, for example

$$\varphi[S(\boldsymbol{X}_k, \eta)] = e^{S(\boldsymbol{X}_k)\eta} \tag{11}$$

and η is some constant. Whether η is positive or negative depends on whether one is solving a maximization or minimization problem. Note that (7) is a particular case of (10), where all weight factors $e^{S(\boldsymbol{X}_k)\eta}$ associated with the elite sample ($\boldsymbol{X}_k$ for which $S(\boldsymbol{X}_k) \geq \widehat{\gamma}_t$) equal unity.

It is shown numerically in [8] that when using (9) for updating, the CE converges quite slowly as compared to using (7) or (10). The reason is that in the latter the updating is based on the elite sample solely, while in the former — on the entire sample. Nevertheless, we mentioned (9) here because of its similarity to the MCE updating formula, (60).

The main CE optimization algorithm can be summarized as follows.

Algorithm 2.1 (Main CE Optimization Algorithm)

1. *Choose some* $\overline{\boldsymbol{p}}_0$. *Set* $t = 1$ *(level counter).*
2. *Generate a sample* $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ *from the density* $f(\boldsymbol{x}, \overline{\boldsymbol{p}}_{t-1})$ *and compute* $\widehat{\gamma}_t$, *the* $(1-\rho)$*-quantile of its performance, according to (3).*
3. *Use the* **same** *sample* $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ *and deliver the solution (7) of the stochastic program (5). Denote the solution by* $\widetilde{\boldsymbol{p}}_t$.
4. *Apply (8) to smooth out the vector* $\widetilde{\boldsymbol{p}}_t$ *and obtain* $\overline{\boldsymbol{p}}_t$.
5. *If for some* $t \geq d$, *say* $d = 5$,

$$\widehat{\gamma}_t = \widehat{\gamma}_{t-1} = \cdots = \widehat{\gamma}_{t-d}, \tag{12}$$

then **stop** *(let* T *denote the final iteration); otherwise set* $t = t + 1$ *and reiterate from step 2.*

2.1 A variant of CE

Note that the program (4) is obtained from the Kullback-Leibler divergence

$$\mathcal{D}(g,h) = \mathbb{E}_g \log \frac{g(\boldsymbol{X})}{h(\boldsymbol{X})} = \int g(\boldsymbol{x}) \log g(\boldsymbol{x})\, d\boldsymbol{x} - \int g(\boldsymbol{x}) \log h(\boldsymbol{x})\, d\boldsymbol{x}, \tag{13}$$

where it is assumed that $h(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{p})$ and that $g(\boldsymbol{x})$ is the optimal importance sampling density, that is [10]

$$g^*(\boldsymbol{x}) = \frac{I_{\{S(\boldsymbol{x}) \geq \gamma\}} f(\boldsymbol{x}, \boldsymbol{u})}{\ell} \tag{14}$$

and $f(\boldsymbol{x}, \boldsymbol{u})$ is the original pdf.

Minimizing the Kullback-Leibler divergence between g^* in (14) and $f(\boldsymbol{x}, \boldsymbol{p})$ is the same as choosing $\boldsymbol{p}$ such that

$$-\int g^*(\boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{p})\, d\boldsymbol{x})$$

is minimized, which is equivalent in turn to solving the maximization problem

$$\max_{\boldsymbol{p}} \int g^*(\boldsymbol{x}) \log f(\boldsymbol{x}, \boldsymbol{p})\, d\boldsymbol{x}\ . \tag{15}$$

Substituting g from (14) into (15) we obtain the maximization program

$$\max_{\boldsymbol{p}} \int \frac{I_{\{S(\boldsymbol{x}) \geq \gamma\}} f(\boldsymbol{x}, \boldsymbol{u})}{\ell} \log f(\boldsymbol{x}, \boldsymbol{p})\, d\boldsymbol{x}, \tag{16}$$

which is equivalent to the program

$$\max_{\boldsymbol{p}} D(\boldsymbol{p}) = \max_{\boldsymbol{p}} \mathbb{E}_{\boldsymbol{u}}\, I_{\{S(\boldsymbol{X}) \geq \gamma\}} \log f(\boldsymbol{X}, \boldsymbol{p}) \tag{17}$$

and coincides with (4) up to the notations.

An alternative expression to (17) (and (4)) can be obtained by assuming in (13) that $g(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{p})$ and $h(\boldsymbol{x}) = g^*(\boldsymbol{x})$, that is by interchanging in (13) f and g^*. In this case we have

$$\mathcal{D}(g,h) = \int f(\boldsymbol{x}, \boldsymbol{p}) \log f(\boldsymbol{x}, \boldsymbol{p})\, d\boldsymbol{x} - \int f(\boldsymbol{x}, \boldsymbol{p}) \log \frac{I_{\{S(\boldsymbol{x}) \geq \gamma\}} f(\boldsymbol{x}, \boldsymbol{u})}{\ell}\, d\boldsymbol{x}. \tag{18}$$

The associated maximization problem is thus

$$\max_{\boldsymbol{p}} D(\boldsymbol{p}) = \max_{\boldsymbol{p}} \left[\int f(\boldsymbol{x}, \boldsymbol{p}) \log f(\boldsymbol{x}, \boldsymbol{p})\, d\boldsymbol{x} - \int f(\boldsymbol{x}, \boldsymbol{p}) \log I_{\{S(\boldsymbol{x}) \geq \gamma\}} f(\boldsymbol{x}, \boldsymbol{u})\, d\boldsymbol{x} \right], \tag{19}$$

which can be also written as

$$\max_{\boldsymbol{p}} D(\boldsymbol{p}) = \max_{\boldsymbol{p}} [\mathbb{E}_{\boldsymbol{p}} \log f(\boldsymbol{X}, \boldsymbol{p}) - \mathbb{E}_{\boldsymbol{p}} \log \{ I_{\{S(\boldsymbol{x}) \geq \gamma\}} f(\boldsymbol{X}, \boldsymbol{u}) \}]. \tag{20}$$

The usefulness of the program (20) as compared to the CE program (17) is questionable, since in contrast to (17), the program (20) does not provide an analytic solution similar to (6) and, thus numerical solutions methods should be used.

3 Background on the MinxEnt Method

The MinxEnt program [6] reads as

$$\begin{aligned} & \min_{f(\boldsymbol{x})} \left\{ \mathcal{D}(f|h) = \int \ln \tfrac{f(\boldsymbol{x})}{h(\boldsymbol{x})} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_f \ln \tfrac{f(\boldsymbol{X})}{h(\boldsymbol{X})} \right\} \\ (\mathrm{P}_0) \qquad & \text{s.t.} \quad \int S_j(\boldsymbol{x}) f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_f S_j(\boldsymbol{X}) = \gamma_j, \ j = 1, \ldots, k, \\ & \int f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1, \ \int h(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1. \end{aligned} \tag{21}$$

Here f and h are *joint* n-dimensional pdf's or n-dimensional pmf's, $S_j(\boldsymbol{x}),\ j = 1, \ldots, k$, are well defined functions and $\boldsymbol{x}$ is an n-dimensional vector. Here h is assumed to be known and is called the *prior* pdf.

As we already mentioned, the *prior* pdf h is assumed to be known in MinxEnt. When h is *unknown*, it is taken as a uniform (continuous or discrete) pdf. In this case

$$\min_{f(\boldsymbol{x})} \left\{ \mathcal{D}(f|h) = \int \ln \frac{f(\boldsymbol{x})}{h(\boldsymbol{x})} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_f \ln \frac{f(\boldsymbol{x})}{h(\boldsymbol{x})} \right\} \tag{22}$$

reduces to

$$\max_{f(\boldsymbol{x})} \left\{ \mathcal{H}(f) = -\int f(\boldsymbol{x}) \ln f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = -\mathbb{E}_f \ln f(\boldsymbol{x}) \right\}. \tag{23}$$

The original program (P_0) is called the *Kullback MinxEnt* program, while the program (P_0) with (22) replaced by (23) is called the *Janes MaxEnt* program. Note that the former minimizes the Kullback-Leibler cross-entropy, while the latter maximizes the Shannon entropy [6].

Remark 3.1 The entropy of a random variable $\boldsymbol{X}$ in the MaxEnt program is its measure of uncertainty. According to MaxEnt, we maximize the uncertainty of the information not given to us, subject to the use of all information given to us in the constraints. On the other hand, the relative entropy in MinxEnt is a measure of the distance between two distributions. The goal is to find a distribution f, which out of all distributions satisfying the given constraints, is closest to h. In spite of this seemingly strong difference, there is a close relationship between MaxEnt and MinxEnt, since (see [6] p.165) if the prior h is the most uncertain distribution, that is a uniform distribution, then being *closest* to it in MinxEnt subject to some constraints means being *maximally uncertain* subject to the same constraints in MaxEnt.

In this paper we shall assume that at least some of the constraints

$$\int S_j(\boldsymbol{x}) f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_f S_j(\boldsymbol{X}) = \gamma_j, \quad j = 1, \ldots, k \tag{24}$$

are not available analytically, and thus we shall deal with the stochastic (sample average) version of the MinxEnt program (P_0). By "stochastic" we mean that

either some (all) values of γ_j are "noisy" or some (all) functions $S_j(\boldsymbol{x})$ are "noisy". In the latter case, we observe $\widehat{S}_j(\boldsymbol{x}) = S_j(\boldsymbol{x})+\zeta$ instead of $S_j(\boldsymbol{x})$, where ζ, called the "noise", is a random variable with zero mean and finite variance, and similarly for the noisy version of γ_j. An alternative "noisy" situation may occur when $S(\boldsymbol{x})$ is "deterministic", but takes an extremely large number of different values, like in COP's. In such a case, as in the CE, we introduce an ASP, which is driven by some pdf and then sample the values $S(\boldsymbol{X})$. A typical example is the TSP with $(n-1)!$ different sample points in the sample space $\mathcal{X}$ of the ASP. Clearly, in such a situation, employing the "deterministic" constraints (24) is meaningless.

If not otherwise stated, we consider below only the discrete case, that is when $f(\boldsymbol{x})$ and $h(\boldsymbol{x})$ are pmf's. We shall denote $f(\boldsymbol{x})$ and $h(\boldsymbol{x})$ as $f(\boldsymbol{x},\boldsymbol{p})$ and $h(\boldsymbol{x},\boldsymbol{u})$, respectively. Note that in the discrete case, the pmf's $f(\boldsymbol{x})$ and $h(\boldsymbol{x})$ are *solely* defined by their corresponding parameter vectors, $\boldsymbol{p}$ and $\boldsymbol{u}$ respectively. If no ambiguity arises we shall also use the notations $\boldsymbol{p}(\boldsymbol{x})$ and $\boldsymbol{u}(\boldsymbol{x})$ for $f(\boldsymbol{x},\boldsymbol{p})$ and $h(\boldsymbol{x},\boldsymbol{u})$ respectively. The continuous case will be treated somewhere else.

In view of the above, the discrete case of (P_0) with $\boldsymbol{p} = (p_1,\ldots,p_m)$ and $\boldsymbol{u} = (u_1,\ldots,u_m)$ can be written in the following compact way

$$
\begin{array}{lll}
 & \min_{\boldsymbol{p}} \mathcal{D}(\boldsymbol{p}|\boldsymbol{u}) = \min_{\boldsymbol{p}} \sum_{i=1}^m p_i \ln \frac{p_i}{u_i} & \\
(\mathrm{P}_1) \qquad \text{s. t.} & \sum_{i=1}^m S_j(x_i) p_i = \mathbb{E}_{\boldsymbol{p}} S_j(X) = \gamma_j,\ j = 1,\ldots,k, & (25)\\
 & \sum_{i=1}^m p_i = 1,\ \sum_{i=1}^m u_i = 1,\ p_i \geq 0, u_i > 0. &
\end{array}
$$

Here

1. X denotes a random variable with m possible outcomes $x_1,\ldots,x_m$.
2. $\boldsymbol{u} = (u_1,\ldots,u_m)$ denotes the prior probability vector of X.
3. $\boldsymbol{p} = (p_1,\ldots,p_m)$ is the decision probability vector. Its optimal value, $\boldsymbol{p}^*$, is unknown in advance and must be found by solving the program (P_1) for given γ_j and $\boldsymbol{u}$.

The solution of the program (P_1) (see [6]) is given by

$$
\begin{array}{rl}
p_i^* & = \frac{u_i \exp\{-\sum_{j=1}^k \lambda_j^* S_j(x_i)\}}{\sum_{r=1}^m u_r \exp\{-\sum_{j=1}^k \lambda_j^* S_j(x_r)\}} = \\
 & = \frac{\mathbb{E}_{\boldsymbol{u}} I_{\{X=x_i\}} \exp\{-\sum_{j=1}^k \lambda_j^* S_j(X)\}}{\mathbb{E}_{\boldsymbol{u}} \exp\{-\sum_{j=1}^k \lambda_j^* S_j(X)\}}, \quad i = 1,\ldots,m,
\end{array}
\tag{26}
$$

where the components λ_j^* of the optimal vector $\boldsymbol{\lambda}^* = (\lambda_1^*,\ldots,\lambda_k^*)$ are obtained from the the numerical solution of the following system of non-linear equations

$$
\begin{array}{rl}
\nabla_{\lambda_j} D(\boldsymbol{\lambda}) & = -\frac{\sum_{i=1}^m S_j(x_i) u_i \exp\left\{-\sum_{r=1}^k S_r(x_i)\lambda_r\right\}}{\sum_{i=1}^m u_i \exp\left\{-\sum_{r=1}^k S_r(x_i)\lambda_r\right\}} + \gamma_j \quad = \\
 & = \quad -\frac{\mathbb{E}_{\boldsymbol{u}} S_j(X) \exp\left\{-\sum_{r=1}^k S_r(X)\lambda_r\right\}}{\mathbb{E}_{\boldsymbol{u}} \exp\left\{-\sum_{r=1}^k S_r(X)\lambda_r\right\}} + \gamma_j \qquad = 0.
\end{array}
\tag{27}
$$

The proof of (26) and (27) is given in section 9.3.

Notice (see [6]) that

- If only the normalization constraint $\sum_{i=1}^{m} p_i = 1$ is left in the program (P_1), then $\boldsymbol{p}^* = \boldsymbol{u}$.
- If $\gamma_j = \mathbb{E}_{\boldsymbol{u}} S_j(X)\ \forall\ j = 1, \ldots, k$, then solving (P_1) we obtain again $\boldsymbol{p}^* = \boldsymbol{u}$.

In addition to the natural normalization constraint for $\boldsymbol{p}(\boldsymbol{x})$ in (P_0), we shall impose below only a single constraint of the form (24), that is we shall set $k = 1$ in (P_0).

Example 3.1 Consider the following MaxEnt program

$$\begin{aligned} & \max_{f(x)} \left\{ \mathcal{H}(f) = \int f(x) \ln f(x) \mathrm{d}x = \mathbb{E}_f \ln f(X) \right\} \\ \text{s.t.} \quad & \mathbb{P}\{X \geq \eta\} = \mathbb{E}_f I_{\{X \geq \eta\}} = \gamma, \\ & \int f(x) \mathrm{d}x = 1. \end{aligned} \tag{28}$$

In this case it is not difficult to see that the solution $f^*(x)$ of (28) is

$$f^*(x) = \frac{1}{\gamma} \exp(-\frac{x}{\gamma}),\ x \geq \eta, \tag{29}$$

that is $f^*(x)$ presents a shifted (to the point η) exponential pdf with mean γ. It is also interesting to note that the optimal importance sampling density $g^*(x)$ (see (14)) for the estimate of the probability $\mathbb{P}\{X \geq \eta\}$, where $X \sim \exp(\frac{1}{\gamma})$ coincides with the MaxEnt one $f^*(x)$.

Example 3.2 To obtain better insight of the program (P_1), consider its particular case associated with die tossing.

$$\begin{aligned} & \min_{\boldsymbol{p}} \mathcal{D}(\boldsymbol{p}|\boldsymbol{u}) = \min_{\boldsymbol{p}} \textstyle\sum_{i=1}^{6} p_i \ln \frac{p_i}{u_i} \\ \text{s. t.} \quad & \textstyle\sum_{i=1}^{6} i p_i = \gamma, \\ & \textstyle\sum_{i=1}^{6} p_i = 1, \sum_{i=1}^{6} u_i = 1,\ p_i \geq 0,\ u_i > 0. \end{aligned} \tag{30}$$

Here, the components p_i^* of the optimal parameter vector $\boldsymbol{p}^*$ are equal to

$$p_i^* = \frac{u_i \exp\{-i\lambda^*\}}{\sum_{r=1}^{6} u_r \exp\{-r\lambda^*\}} = \frac{\mathbb{E}_{\boldsymbol{u}} I_{\{X=i\}} \exp\{-X\lambda^*\}}{\mathbb{E}_{\boldsymbol{u}} \exp\{-X\lambda^*\}},\ i = 1, \ldots, 6, \tag{31}$$

where λ^* (see (27)) is derived from the numerical solution of

$$\frac{\sum_{i=1}^{6} i u_i \exp\{-i\lambda\}}{\sum_{i=1}^{6} u_i \exp\{-i\lambda\}} = \gamma. \tag{32}$$

γ	λ^*	p_1^*	p_2^*	p_3^*	p_4^*	p_5^*	p_5^*	$\mathcal{H}(\boldsymbol{p}^*)$
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	0.9544	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

Table 3.1 λ^*, $\boldsymbol{p}^*$ and $\mathcal{H}(\boldsymbol{p}^*)$ as function of γ for a fair die.

Table 3.1, which is an exact replica of Table 4.1 of [6], presents λ^*, $\boldsymbol{p}^*$ and $\mathcal{H}(\boldsymbol{p}^*)$ as functions of γ for a fair die, that is with prior $(u_1 = \frac{1}{6}, \ldots, u_6 = \frac{1}{6})$. The table is self-explanatory.

Note that

- For the extreme values of γ, that is for $\gamma = 6$ and $\gamma = 1$, the corresponding optimal solutions are $\boldsymbol{p}^* = (0, 0, \ldots, 1)$ and $\boldsymbol{p}^* = (1, 0, \ldots, 0)$ respectively. It can also be shown that $\boldsymbol{p}^*$ is degenerated regardless of the prior $\boldsymbol{u}$ and while using instead of $S(x) = x$ an arbitrary function $S(x),\ x = 1, \ldots, 6$, with a single optimum. This important observation will play a crucial role while solving COP's with the MCE method.

- For both degenerated vectors, $\boldsymbol{p}^* = (0, 0, \ldots, 1)$ and $\boldsymbol{p}^* = (1, 0, \ldots, 0)$, the entropy is $\mathcal{H}(\boldsymbol{p}^*) = 0$, and thus there is no uncertainty.

- $\mathcal{H}(\gamma)$ is strictly concave in γ and the maximal entropy $\max_\gamma \mathcal{H}(\gamma) = \mathcal{H}(3.5) = 1.79176$.

- $\boldsymbol{p}^*(\gamma = 3.5) = \boldsymbol{u} = (\frac{1}{6}, \ldots, \frac{1}{6})$.

Remark 3.2 Application of Sanov's Theorem. The MinxEnt program (30) can be useful for finding rare-event probabilities associated with die tossing. In particular, assuming that we throw a fair die, what is the probability of the event A, that the average of N throws equals to γ? From Sanov's theorem [2] it follows that for large N

$$\mathbb{P}_{\boldsymbol{u}}(A) = \mathbb{P}_{\boldsymbol{u}}\left\{\frac{1}{N}\sum_{k=1}^{N} X_i \geq \gamma\right\} \approx 2^{-N\mathcal{D}(\boldsymbol{p}^*|\boldsymbol{u})}, \tag{33}$$

where $\boldsymbol{p}^*$ is the solution of the MinxEnt program (30) and we use $\log_2$ in (30) instead of ln.

For example, if we set $\gamma = 4$ then we have from Table 3.1 that $\boldsymbol{p}^* = (0.1031,\ 0.1227,\ 0.1461,\ 0.1740,\ 0.2072,\ 0.2468)$, and using using $\log_2$ instead of ln, we obtain for $N = 10,000$ that $\mathcal{D}(\boldsymbol{p}^*|\boldsymbol{u}) = 0.0623$. The resulting probability is therefore $\mathbb{P}(A) \approx 2^{-623}$. Note that $\boldsymbol{p}^*$ corresponds to the optimal change of measure in the relative entropy sense. Note also that since we use $\log_2$ instead of ln, we have that $\mathcal{D}(\boldsymbol{p}^*|\boldsymbol{u}) = 0.0623$ instead of $\mathcal{H}(\boldsymbol{p}^*, \gamma = 4) = 1.74843$ as in Table 3.1.

For general random functions $S(\boldsymbol{X})$, formula (33) can can be written as

$$\mathcal{L} = \mathbb{P}_h \left\{ \frac{1}{N} \sum_{k=1}^{N} S(\boldsymbol{X}_i) \geq \gamma \right\} = \mathbb{E}_h \left\{ I_{\frac{1}{N}\sum_{k=1}^{N} S(\boldsymbol{X}_k) \geq \gamma} \right\} \approx 2^{-N\mathcal{D}(f^*(\boldsymbol{X})|h(\boldsymbol{X}))}, \tag{34}$$

where we denote $\mathcal{L} = \mathbb{P}\{A\}$.

So far we have dealt with (P_1), in which X is a single-dimensional variate. The extension to the multi-dimensional case is straight forward. We have

$$p^*_{i_1 \cdots i_n} = \frac{\mathbb{E}_{\boldsymbol{u}} I_{\{\boldsymbol{X}=(i_1,\cdots,i_n)\}} \exp\{-\lambda^* S(\boldsymbol{X})\}}{\mathbb{E}_{\boldsymbol{u}} \exp\{-\lambda^* S(\boldsymbol{X})\}},\ i_k = 1, \ldots, m;\ k = 1, \ldots, n, \tag{35}$$

where $\boldsymbol{X} \sim h(\boldsymbol{x}, \boldsymbol{u})$, $p_{i_1 \cdots i_n}$ denotes the probability of a particular outcome, and λ^* is obtained from the solution the following non-linear equation

$$\nabla_\lambda D(\lambda) = -\frac{\mathbb{E}_{\boldsymbol{u}} S(\boldsymbol{X}) \exp\{-\lambda S(\boldsymbol{X})\}}{\mathbb{E}_{\boldsymbol{u}} \exp\{-\lambda S(\boldsymbol{X})\}} + \gamma = 0. \tag{36}$$

Note that (36) has a solution, provided that

$$\min_{\boldsymbol{x}} S(\boldsymbol{x}) < \gamma < \max_{\boldsymbol{x}} S(\boldsymbol{x}).$$

Equations (35) and (36) will serve as the basic formulas for updating the vector $\boldsymbol{p}$ and the scalar λ in the dynamic version of MinxEnt, (see (48) and (49)).

Example 3.3 Consider the following two-dimensional MinxEnt program

$$\begin{aligned} & \min_{\boldsymbol{p}} \mathcal{D}(\boldsymbol{p}|\boldsymbol{u}) = \min_{\boldsymbol{p}} \textstyle\sum_{i=1}^{m} \sum_{j=1}^{m} p_{ij} \ln \frac{p_{ij}}{u_{ij}} \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{m} \sum_{j=1}^{m} S(x_{ij}) p_{ij} = \mathbb{E}_{\boldsymbol{p}} S(\boldsymbol{X}) = \gamma, \\ & \textstyle\sum_{i=1}^{m} \sum_{j=1}^{m} p_{ij} = 1,\ \sum_{i=1}^{m} \sum_{j=1}^{m} u_{ij} = 1,\ p_{ij} \geq 0,\ u_{ij} > 0, \end{aligned} \tag{37}$$

where $S(\boldsymbol{x})$ is an arbitrary function associated, say with tossing two dies. The optimal solution of (37) (see (26)) is

$$p^*_{ij} = \frac{\mathbb{E}_{\boldsymbol{u}} I_{\{\boldsymbol{X}=\boldsymbol{x}_{ij}\}} \exp\{-\lambda^* S(\boldsymbol{X})\}}{\mathbb{E}_{\boldsymbol{u}} \exp\{-\lambda^* S(\boldsymbol{X})\}};\ i = 1, \ldots m;\ j = 1, \ldots m, \tag{38}$$

where $\boldsymbol{X} \sim h(\boldsymbol{x}, \boldsymbol{u})$, and λ^* is derived from the solution of (36).

4 Solving a Sequence of Deterministic MinxEnt Problems

In this section we solve iteratively a sequence of deterministic MinxEnt programs and give an associated MCE algorithm, Algorithm 4.1. Although Algorithm 4.1 is not directly applicable for solving COP's, it, however, provides good insight to our main stochastic MCE algorithm, Algorithm 5.1, since it presents the deterministic replica of the latter.

The goal of MCE is similar to that of CE, namely starting at some $\{\gamma_0, \boldsymbol{p}_0\}$, to generate iteratively a sequence of tuples $\{\gamma_t, \boldsymbol{p}_t\}$, with the view that after some finite number of iterations, say T iterations, $\{\gamma_T, \boldsymbol{p}_T\}$ will approximate quite accurately the optimal tuple $\{\gamma^*, \boldsymbol{p}^*\}$. Here $\gamma^* = \max_{\boldsymbol{x}\in\mathcal{X}} S(\boldsymbol{x})$ and $\boldsymbol{p}^*$ denotes the optimal degenerated solution defined similarly to (35) and (36).

To provide better insight, we start with a sequence of MCE programs associated with die tossing (see program (30)) and show how starting, say at $\boldsymbol{p}_0 = (\frac{1}{6}, \ldots, \frac{1}{6})$, to generate a sequence $\{\gamma_t, \boldsymbol{p}_t\}$, which converges to one of the two extremes: (i) $(\gamma_T, \boldsymbol{p}_T) = (6,\ (0,\ 0, \ldots, 1))$ or (ii) $(\gamma_T, \boldsymbol{p}_T) = (1,\ (1,\ 0, \ldots, 0))$. We proceed with quite general sequences of MinxEnt programs, which will provide insight and shall motivate our main MCE Algorithm of Section 5. If not otherwise stated we shall have in mind a maximization problem.

Example 4.1 Let $n = 1$ and consider the following *sequence* of MinxEnt programs.

$$\begin{aligned} &\min_{\boldsymbol{p}_t} \mathcal{D}(\boldsymbol{p}_t|\boldsymbol{p}_{t-1}) = \min_{\boldsymbol{p}_t} \textstyle\sum_{i=1}^m p_{t,i} \ln \frac{p_{t,i}}{p_{t-1,i}} \\ \text{s. t.}\quad &\textstyle\sum_{i=1}^m S(i) p_{t,i} = \gamma_t, \\ &\textstyle\sum_{i=1}^m p_{t,i} = 1. \end{aligned} \tag{39}$$

If $S(i) = i$ and $m = 6$, then we have the die tossing program similar to (30). Also, in n analogy to (26), the $j-th$ component ($j = 1, \ldots, 6$) of the optimal vector $\boldsymbol{p}_t$ of the program (39), (43) can be written as

$$p_{t,j} = \frac{p_{t-1,j} \exp\{-j\lambda_t\}}{\sum_{r=1}^6 p_{t-1,r} \exp\{-r\lambda_t\}} = \frac{\mathbb{E}_{\boldsymbol{p}_{t-1}} I_{\{X=j\}} \exp\{-X\lambda_t\}}{\mathbb{E}_{\boldsymbol{p}_{t-1}} \exp\{-X\lambda_t\}}, \tag{40}$$

where $X \sim f(x, \boldsymbol{p}_{t-1})$ and λ_t is the optimal solution of the dual program at iteration t, which in analogy to (32) can be derived from the numerical solution of

$$\frac{\sum_{i=1}^6 i p_{t-1,i} \exp\{-i\lambda\}}{\sum_{i=1}^6 p_{t-1,i} \exp\{-i\lambda\}} = \gamma_t. \tag{41}$$

For fixed $(\gamma_t, \boldsymbol{p}_{t-1})$, denote by $\boldsymbol{p}_t = (p_{t1}, \ldots, p_{tm}),\ \ t = 1, 2, \ldots$, the optimal solution of program (39) at iteration t. In order to generate from (39) a sequence $\{\gamma_t, \boldsymbol{p}_t\}$ we shall fix, in analogy to CE, some initial parameters $\boldsymbol{p}_0$ and the *rarity parameter*, ρ $(0 < \rho < 1)$. Similarly to CE, we order the values $S(x),\ x =$

$1, \ldots, m$, from smallest to largest and denote them $S_{(1)} \leq \ldots \leq S_{(m)}$. Further denote by $p_{t-1,(i)}$ the element of $\boldsymbol{p}_{t-1}$ corresponding to $S_{(i)}$

We present below two different approaches for generating the sequence $\{\gamma_t\}$. Both approaches make use of the $(1-\rho)$ quantile of the distribution of $S(x)$ at iteration t. Denote the index of this quantile by c_t, then

$$c_t = \max\Big\{c:\ \sum_{i=c}^{m} p_{t-1,(i)} \geq \rho\Big\}. \tag{42}$$

According to the first approach, γ_t is the weighted average of the largest (elite) values of $S(x)$ starting from c_t, that is

$$\gamma_t = \frac{\sum_{i=c_t}^{m} p_{t-1,(i)} S_{(i)}}{\sum_{i=c_t}^{m} p_{t-1,(i)}}. \tag{43}$$

Clearly, $\gamma_t > \mathbb{E}_{\boldsymbol{p}_{t-1}} S(X)$. If in the die example $\rho = 0.5$ and $\boldsymbol{p}_0 = (\frac{1}{6}, \ldots, \frac{1}{6})$, then we obtain $c_1 = 4$ and $\gamma_1 = 5 > \mathbb{E}_{\boldsymbol{p}_0} X = 3.5$.

According to the second approach, γ_t presents the worst value of $S(X)$ among the elite values, that is (see also (3) for CE)

$$\gamma_t = S_{(c_t)}, \tag{44}$$

where $X \sim f(x, \boldsymbol{p}_{t-1})$.

If not otherwise stated we shall use (43).

Solving iteratively (39), (43), we generate a sequence of scalars $\gamma_1, \gamma_2, \ldots$ and a sequence of vectors $\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots$ with the view that for some T (big enough), $\gamma_T \approx \gamma^*$ and $\boldsymbol{p}_T \approx \boldsymbol{p}^*$, where $\boldsymbol{p}^*$ is the true optimal degenerated vector (see the corresponding data in the lower and the upper rows of Table 3.1).

With this in mind, one can devise in analogy to the CE method [3], an iterative procedure for approximating the true optimal tuple $(\gamma^*, \boldsymbol{p}^*)$. Each iteration of such a procedure consists of two main phases: in the first phase γ_t is updated, and in the second — $\boldsymbol{p}_t$ is updated. Specifically, setting $\boldsymbol{p}_0 = \boldsymbol{u}$, where $\boldsymbol{u}$ is, say, a parameter vector with equal components, we generate the subsequent γ_t and $\boldsymbol{p}_t$ as follows:

1. **Adaptive updating of γ_t.** For a fixed $\boldsymbol{p}_{t-1}$, calculate γ_t according to (43). For example, if in the die tossing example, we set $\rho = 0.5$, then $\gamma_1 = 5$.

2. **Adaptive updating of $\boldsymbol{p}_t$.** For fixed γ_t and $\boldsymbol{p}_{t-1}$, derive $\boldsymbol{p}_t$ from the solution of the MinxEnt program (39).

Table 4.1 presents data similar to Table 3.1 for $\boldsymbol{p}_0 = \boldsymbol{u} = (\frac{1}{6}, \ldots, \frac{1}{6})$ and $\rho = 0.5$. In particular, it presents $\gamma_t,\ \lambda_t,\ \boldsymbol{p}_t, \mathcal{D}(\boldsymbol{p}_t|\boldsymbol{p}_{t-1})$ and $\mathcal{H}(\boldsymbol{p}_t)$ as functions of t while solving (39), (43) iteratively. Clearly, in order to speed up the convergence of $\boldsymbol{p}_t$ to $\boldsymbol{p}^*$ we shall decrease ρ. Note that for $0 < \rho \leq \frac{1}{6}$ the convergence to $\boldsymbol{p}^* = (0, 0, 0, 0, 0, 1)$ takes place in a single iteration. In addition, it is readily

t	γ_t	λ_t	p_{t1}	p_{t2}	p_{t3}	p_{t4}	p_{t5}	p_{t6}	$\mathcal{D}(\cdot)$	$\mathcal{H}(\boldsymbol{p}_t)$
0			0.166	0.166	0.166	0.166	0.166	0.166		1.792
1	5.000	-0.629	0.020	0.038	0.0723	0.136	0.255	0.478	0.424	1.367
2	5.652	-1.351	0.001	0.003	0.013	0.050	0.192	0.741	0.187	0.769
3	6.000	$-\infty$	0.000	0.000	0.000	0.000	0.000	1.000	0.299	0.000

Table 4.1 The behavior of γ_t, λ_t, $\boldsymbol{p}_t$, $\mathcal{D}(\boldsymbol{p}_t|\boldsymbol{p}_{t-1})$ and $\mathcal{H}(\boldsymbol{p}_t)$ as functions of t for $\boldsymbol{p}_0 = \boldsymbol{u} = (\frac{1}{6}, \ldots, \frac{1}{6})$ and $\rho = 0.5$.

seen, that if we would use in (43) the *smallest* values of $S(x)$ instead of its *largest* values, then the process would converge after exactly the same number of iterations to the tuple $(\gamma^*,\ \boldsymbol{p}^*) = (1, (1, 0, 0, 0, 0, 0))$.

The proof of the convergence of the solutions of the sequence of programs (39) to the optimal solution $(\gamma^*, \boldsymbol{p}^*)$ for the single-dimensional case with an arbitrary $S(x)$ and γ_t defined as per (43) is given in Proposition 9.1 of Section 9.3. In particular, we prove the following:

- The sequence $\{\boldsymbol{p}_t\}$ obtained recursively from the solution of the program (39), (43) converges to the degenerated case $\boldsymbol{p}^* = (0, 0, \ldots, 1)$.
- The sequence $\{\mathcal{H}(\boldsymbol{p}_t)\}$ monotonically decreases in t and for some T big enough $\mathcal{H}(\boldsymbol{p}_T) = \mathcal{H}(\boldsymbol{p}^*) = 0$.
- The sequence $\{\gamma_t\}$ monotonically increases in t and for some T big enough, $\gamma_T = \gamma^*$.

The single-dimensional MinxEnt program (39) naturally extends to the multi-dimensional case. Indeed, let $\boldsymbol{X}$ be an n-dimensional random vector distributed $\boldsymbol{p}_{t-1}(\boldsymbol{x})$, then the multi-dimensional version of (39) reads as

$$\begin{aligned} &\min_{\boldsymbol{p}_t} \left\{ \mathcal{D}(\boldsymbol{p}_t|\boldsymbol{p}_{t-1}) = \textstyle\sum_{\boldsymbol{x}} \boldsymbol{p}_t(\boldsymbol{x}) \ln \frac{\boldsymbol{p}_t(\boldsymbol{x})}{\boldsymbol{p}_{t-1}(\boldsymbol{x})} \right\} \\ \text{s. t.} \quad &\textstyle\sum_{\boldsymbol{x}} S(\boldsymbol{x}) \boldsymbol{p}_t(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{p}_t} S(\boldsymbol{X}) \ = \gamma_t, \\ &\textstyle\sum_{\boldsymbol{x}} \boldsymbol{p}_t(\boldsymbol{x}) = 1. \end{aligned} \tag{45}$$

We call (45), the *dynamic MinxEnt program* to distinguish it from the program (P_0), which we call, the *static MinxEnt program*.

In this case formulas (42) and (43) become

$$c_t = \max \left\{ c : \sum_{i=c}^{|\mathcal{X}|} p_{t-1,(i)} \geq \rho \right\} \tag{46}$$

and

$$\gamma_t = \frac{\sum_{i=c_t}^{|\mathcal{X}|} p_{t-1,(i)} S_{(i)}}{\sum_{i=c_t}^{|\mathcal{X}|} p_{t-1,(i)}}, \tag{47}$$

respectively, while formula (44) remains unchanged. Here $|\mathcal{X}|$ is the cardinality of the sample space. For example, for the max-cut problem $|\mathcal{X}| = 2^{n-1}$.

Denote the optimal solution of program (45) by

$$\boldsymbol{p}_t = \{p_{t,i_1\cdots i_n},\ i_k = 1,\ldots,m;\ k = 1,\ldots,n\}.$$

In analogy to (40) we have that

$$p_{t,i_1\cdots i_n} = \frac{\mathbb{E}_{\boldsymbol{p}_{t-1}} I_{\{\boldsymbol{X}=(i_1,\cdots,i_n)\}} \exp\{-\lambda_t S(\boldsymbol{X})\}}{\mathbb{E}_{\boldsymbol{p}_{t-1}} \exp\{-\lambda_t S(\boldsymbol{X})\}},\ i_k = 1,\ldots,m;\ k = 1,\ldots,n, \tag{48}$$

and in analogy to (36), λ_t can be derived (numerically) from the solution of the following non-linear equation

$$\frac{\mathbb{E}_{\boldsymbol{p}_{t-1}} S(\boldsymbol{X}) \exp\{-\lambda S(\boldsymbol{X})\}}{\mathbb{E}_{\boldsymbol{p}_{t-1}} \exp\{-\lambda S(\boldsymbol{X})\}} = \gamma_t. \tag{49}$$

Note that the elements of the vector $\boldsymbol{p}_t = \{p_{t,i_1\cdots i_n}, i_k = 1,\ldots,m;\ k = 1,\ldots,n\}$ in (48) are updated *analytically*, while λ_t is updated *numerically*. Also note that in the MaxEnt version of (45) all the components of the vector $\boldsymbol{p}_{t-1}$ are equal to m^{-n}. We present below the deterministic MCE Algorithm for solving the program (45) iteratively, while generating a sequence of tuples $\{\gamma_t, \boldsymbol{p}_t\}$.

Algorithm 4.1 (Deterministic MCE Algorithm)

1. *Choose some* $\boldsymbol{p}_0$*, say* $\boldsymbol{p}_0 = \boldsymbol{u}$ *(with equal elements). Set* $t = 1$ *(level counter).*

2. *For a given pdf* $\boldsymbol{p}_{t-1}(\boldsymbol{x})$ *calculate* γ_t *according to (47).*

3. *Deliver the optimal* λ_t *from the numerical solution of (49). Deliver the optimal* $\boldsymbol{p}_t = \{p_{t,i_1\cdots i_n}, i_k = 1,\ldots,m;\ k = 1,\ldots,n\}$ *of the program (45) according to (48).*

4. *If for some* $t \geq d$*, say* $d = 5$*,*

$$\gamma_t = \gamma_{t-1} = \cdots = \gamma_{t-d}, \tag{50}$$

then **stop** *(let* T *denote the final iteration); otherwise set* $t = t + 1$ *and reiterate from step 2.*

Let us discuss trajectory generation for the MinxEnt program (45). It is crucial to notice the following difference between CE and MinxEnt. In CE, the probabilities p_{t,i_k} are obtained from the solution of the CE program (4), and are associated *directly* with either the *nodes* or with the *edges* of the graph (for SNN and SEN respectively). In contrast, solving the MCE program (45) we derive a joint pdf $p_{t,i_1\cdots i_n}$, which represents the probability of choosing the *trajectory* $(i_1,\cdots,i_n)$ in the graph.

For real COP's with the size of hundreds of nodes, the joint pdf $p_{t,i_1\cdots i_n}$ contains an enormous number of elements. Randomly sampling from such a pdf is *impractical*. Therefore we need to devise a mechanism, which based on $p_{t,i_1\cdots i_n}$, will generate trajectories for SNN and SEN similarly to CE. The easiest way of doing so is simply to find first all n marginal pdf's p_{t,i_k} from the n-dimensional joint pdf $p_{t,i_1\cdots i_n}$, then associate each marginal to its corresponding node or edge, and finally, generate the trajectories exactly as in CE. For example, the marginal pdf's can be simply calculated from the joint one $p_{t,i_1\cdots i_n}$ as

$$\widehat{p}_{t,i_k} = \sum_{i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_n} p_{t,i_1\cdots i_n},\ i_k = 1,\ldots,m;\ k = 1,\ldots,n. \tag{51}$$

Denote the n-dimensional vector of these marginal pdf's (probabilities) corresponding to $\boldsymbol{x} = (i_1,\ldots,i_n)$ as

$$\boldsymbol{p}_t^{(m)}(\boldsymbol{x}) = \{\widehat{p}_{t,i_k}\ k = 1,\ldots,n\}. \tag{52}$$

Alternatively, based on $p_{t,i_1\cdots i_n}$, one can generate trajectories using the conditional distributions

$$p_{t,i_k|i_1\cdots i_{k-1}},\ k = 1,\ldots,n,$$

which can be calculated using the following formula

$$p_{t,i_k|i_1\cdots i_{k-1}} = \frac{p_{t,i_1\cdots i_k}}{p_{t,i_1\cdots i_{k-1}}},\ k = 1,\ldots,n. \tag{53}$$

Here it is assumed that $p_{t,i_1|i_0} \equiv p_{t,i_1}$. Note that the marginal pdf's

$$p_{t,i_1\cdots i_{n-1}}, p_{t,i_1\cdots i_{n-2}}, \ldots, p_{t,i_1}$$

could be calculated recursively from their joint one $p_{t,i_1\cdots i_n}$.

We proceed with $\boldsymbol{p}_t^{(m)}$. As mentioned, trajectory generation using $\boldsymbol{p}_t^{(m)}$ is quite simple, i. e., identical to the standard CE method. It should be noted, however, that using $\boldsymbol{p}_t^{(m)}$ we generate a trajectory $\boldsymbol{x} = (i_1,\ldots,i_n)$ based on the associated marginal probabilities $\boldsymbol{p}_t^{(m)}(\boldsymbol{x})$ rather than on the true pdf $\boldsymbol{p}(\boldsymbol{x}) = p_{t,i_1,\ldots i_n}$. In short, we shall generate trajectories using the probability vector

$$\widehat{p}_{t,i_1\ldots,i_n} = \widehat{p}_{t,i_1}\cdots\widehat{p}_{t,i_n}, \tag{54}$$

which equals to the product of the marginal probabilities. One might think that such straightforward trajectory generation could affect the efficiency of the MCE method. This is, however, not the case, since for typical COP's, like TSP and max-cut the function $S(\boldsymbol{x})$ is " close" to an additive one; and for the later case we clearly have that $p_{t,i_1,\ldots i_n} = \widehat{p}_{t,i_1\ldots,i_n}$. This is one of the main reason that MCE works so well for COP's!

With this at hand, denote

$$\widehat{\boldsymbol{p}}_t = \{\widehat{p}_{t,i_1\cdots i_n},\ i_k = 1,\ldots,m;\ k = 1,\ldots,n\}$$

and consider the following modification of the original MinxEnt program (45)

$$\begin{aligned} & \min_{\boldsymbol{p}_t} \left\{ \mathcal{D}(\boldsymbol{p}_t | \widehat{\boldsymbol{p}}_{t-1}) = \sum_{\boldsymbol{x}} \boldsymbol{p}_t(\boldsymbol{x}) \ln \frac{\boldsymbol{p}_t(\boldsymbol{x})}{\widehat{\boldsymbol{p}}_{t-1}(\boldsymbol{x})} \right\} \\ \text{s. t.} \quad & \sum_{\boldsymbol{x}} S(\boldsymbol{x}) \boldsymbol{p}_t(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{p}_t} S(\boldsymbol{X}) \ = \gamma_t, \\ & \sum_{\boldsymbol{x}} \boldsymbol{p}_t(\boldsymbol{x}) = 1. \end{aligned} \tag{55}$$

Recall that $\boldsymbol{p}_t^{(m)}(\boldsymbol{x}) = \{\widehat{p}_{t,i_k} \ k = 1, \ldots, n\}$ presents an n-dimensional vector of marginal pdf's, while $\widehat{\boldsymbol{p}}_t = \{\widehat{p}_{t,i_1 \cdots i_n}, \ i_k = 1, \ldots, m; \ k = 1, \ldots, n\}$ presents the product of the above marginal pdf's.

Note that the program (55) coincides with the program (45), provided $\widehat{\boldsymbol{p}}_{t-1}$ in (55) is replaced by $\boldsymbol{p}_{t-1}$.

The MaxEnt counterpart of both (45) and (55) can be written as

$$\begin{aligned} & \max_{\boldsymbol{p}_t} \left\{ \mathcal{H}(\boldsymbol{p}_t) = \sum_{\boldsymbol{x}} \boldsymbol{p}_t(\boldsymbol{x}) \ln \boldsymbol{p}_t(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{p}_t} \ln \boldsymbol{p}_t(\boldsymbol{X}) \right\} \\ \text{s. t.} \quad & \sum_{\boldsymbol{x}} S(\boldsymbol{x}) \boldsymbol{p}_t(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{p}_t} S(\boldsymbol{X}) \ = \gamma_t, \\ & \sum_{\boldsymbol{x}} \boldsymbol{p}_t(\boldsymbol{x}) = 1. \end{aligned} \tag{56}$$

Remark 4.1 In our numerical results in Section 7 we show that MCE, (and in the particular its MaxEnt variant) is typically a little more accurate than its counterpart CE. Our explanation for that is based on the following facts. Consider the particular case of the Kullback-Leibler divergence $\mathcal{D}(g, h)$ in (13), namely where $g(\boldsymbol{x}) = g(x_1, x_2)$ and $h(\boldsymbol{x}) = g_1(x_1) g_2(x_2)$, that is

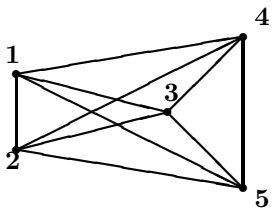
$$\mathcal{D}(g, h) = \mathbb{E}_g \log \frac{g(\boldsymbol{X})}{h(\boldsymbol{X})} \ = \mathbb{E}_g \log \frac{g(X_1, X_2)}{g_1(X_1) g_2(X_2)}. \tag{57}$$

Such divergence is called the *mutual information* [2]. It presents the relative entropy between the joint pdf $g(\boldsymbol{x}) = g(x_1, x_2)$ and the product of marginal distributions $g_1(x_1) g_2(x_2)$. It is well known [2] that among all possible *product distributions* associated with the joint one $g(\boldsymbol{x}) = g(x_1, x_2)$, the distribution $g_1(x_1) g_2(x_2)$ has the *maximal* relative entropy and, thus $g_1(x_1) g_2(x_2)$ is the "closest" to $g(x_1, x_2)$ in the relative entropy sense. The extension to the case of more than two variables is straightforward.

It follows from the above that among all possible product distributions associated with the true optimal joint distribution $p_{t,i_1 \cdots i_n}(\boldsymbol{x})$, the distribution $\widehat{\boldsymbol{p}}_{t-1}(\boldsymbol{x})$, which presents the product of the marginal distributions obtained from $p_{t,i_1 \cdots i_n}(\boldsymbol{x})$ and which is used in the MinxEnt program (55) is the best in the relative entropy sense. In contrast, the pdf $f(\boldsymbol{x}, \boldsymbol{p})$, which is used in the CE program (17) also presents a product of marginal pdf's, it is *not optimal* (in the relative entropy sense) with respect to the true optimal importance sampling pdf $g^*(\boldsymbol{x})$ (see (14)). Note finally, that although in MaxEnt a uniform pdf is used instead of $\widehat{\boldsymbol{p}}_{t-1}(\boldsymbol{x})$, the trajectory generation is also performed as in the MinxEnt, namely using the marginal pdf's of $p_{t,i_1 \cdots i_n}(\boldsymbol{x})$

The following toy example illustrates step by step the performance of the deterministic MinxEnt Algorithm 4.1 and some of its modifications, while solving a simple 5-node max-cut problem. The small size of the problem allows us to present all the calculations.

Example 4.2 (Illustration of Algorithm 4.1 and its modifications) Consider the 5-node graph with cost matrix C presented in Figure 4.1. The $|\mathcal{X}| = 2^{n-1} = 16$ possible cuts and the corresponding cut values are given in Table 4.2. It follows that in this case the optimal cut vector is $\boldsymbol{x}^* = (1, 1, 0, 0, 0)$ with $S(\boldsymbol{x}^*) = \gamma^* = 28$.



$$C = \begin{pmatrix} 0 & 1 & 3 & 5 & 6 \\ 1 & 0 & 3 & 6 & 5 \\ 3 & 3 & 0 & 2 & 2 \\ 5 & 6 & 2 & 0 & 2 \\ 6 & 5 & 2 & 2 & 0 \end{pmatrix} \qquad (58)$$

Figure 4.1 A 5-node graph

$\boldsymbol{X}$	V_1	V_2	$S(\boldsymbol{X})$
(1,0,0,0,0)	$\{1\}$	$\{2,3,4,5\}$	15
(1,1,0,0,0)	$\{1,2\}$	$\{3,4,5\}$	28
(1,0,1,0,0)	$\{1,3\}$	$\{2,4,5\}$	19
(1,0,0,1,0)	$\{1,4\}$	$\{2,3,5\}$	20
(1,0,0,0,1)	$\{1,5\}$	$\{2,3,4\}$	18
(1,1,1,0,0)	$\{1,2,3\}$	$\{4,5\}$	26
(1,1,0,1,0)	$\{1,2,4\}$	$\{3,5\}$	21
(1,1,0,0,1)	$\{1,2,5\}$	$\{3,4\}$	21
(1,0,1,1,0)	$\{1,3,4\}$	$\{2,5\}$	20
(1,0,1,0,1)	$\{1,3,5\}$	$\{2,4\}$	18
(1,0,0,1,1)	$\{1,4,5\}$	$\{2,3\}$	19
(1,1,1,1,0)	$\{1,2,3,4\}$	$\{5\}$	15
(1,1,1,0,1)	$\{1,2,3,5\}$	$\{4\}$	15
(1,1,0,1,1)	$\{1,2,4,5\}$	$\{3\}$	10
(1,0,1,1,1)	$\{1,3,4,5\}$	$\{2\}$	15
(1,1,1,1,1)	$\{1,2,3,4,5\}$	$\emptyset$	0

Table 4.2 The 16 possible cuts of Example 4.2.

To proceed we set $\rho = 0.5$ and $\boldsymbol{p}_0 = (\frac{1}{16}, ..., \frac{1}{16})$. We shall show that $\boldsymbol{p}_t$ converges fast to the optimal vector $\boldsymbol{p}^* = (1, 1, 0, 0, 0)$, provided γ_t is calculated according to (47). We consider the following cases:

(a) The original Algorithm 4.1.

(b) The MaxEnt modification of Algorithm 4.1, where the components of the vector $\boldsymbol{p}_{t-1}$ in (45) are equal to $\frac{1}{|\mathcal{X}|}$.

(c) Modification (55) of Algorithm 4.1, where the vector $\boldsymbol{p}_{t-1}$ in (45) is replaced by $\widehat{\boldsymbol{p}}_{t-1}$ and the trajectories (cuts) are generated using the marginal pdf $\boldsymbol{p}_{t-1}^{(m)}(\boldsymbol{x})$ instead of $\boldsymbol{p}_{t-1}(\boldsymbol{x})$.

(a) Original Algorithm 4.1
Iteration 1

It is readily seen that for $\rho = 0.5$ and $\boldsymbol{p}_0 = (\frac{1}{16}, ..., \frac{1}{16})$ the elite sample corresponds to the best eight cuts. Calculating γ_1 according to (47) yields $\gamma_1 = 21.75$. Applying the bisection method to (49) we obtain $\lambda_1 = -0.1522$. The corresponding vector $\boldsymbol{p}_1 = \{p_{1,i_2\cdots i_5},\ i_k = 0, 1;\ k = 2, \ldots, 5\}$ obtained from (48) is presented in Table 4.3.

$\boldsymbol{X}$					$S(\boldsymbol{X})$	$\boldsymbol{p}_1$
1	1	0	0	0	28	0.2176
1	1	1	0	0	26	0.1605
1	1	0	0	1	21	0.0750
1	1	0	1	0	21	0.0750
1	0	0	1	0	20	0.0644
1	0	1	1	0	20	0.0644
1	0	0	1	1	19	0.0553
1	0	1	0	0	19	0.0553
1	0	0	0	1	18	0.0475
1	0	1	0	1	18	0.0475
1	0	0	0	0	15	0.0301
1	0	1	1	1	15	0.0301
1	1	1	0	1	15	0.0301
1	1	1	1	0	15	0.0301
1	1	0	1	1	10	0.0141
1	1	1	1	1	0	0.0031

Table 4.3 : Results for iteration 1 of the original Algorithm 4.1.

Iteration 2

It follows from Table 4.3 that the total probability of the best (elite) four cuts is $0.527 > 0.5 = \rho$. Hence, we obtain from (47) and Table 4.3 that the average $S(\boldsymbol{X})$ of those four cuts is $\gamma_2 = 25.4$. Proceeding similarly to Iteration 1 we obtain from (49) and (48) that $\lambda_2 = -0.1975$ and the vector $\boldsymbol{p}_2$ is as presented in Table 4.4.

$\boldsymbol{X}$					$S(\boldsymbol{X})$	$\boldsymbol{p}_2$
1	1	0	0	0	28	0.5043
1	1	1	0	0	26	0.2506
1	1	0	0	1	21	0.0436
1	1	0	1	0	21	0.0436
1	0	0	1	0	20	0.0308
1	0	1	1	0	20	0.0308
1	0	0	1	1	19	0.0217
1	0	1	0	0	19	0.0217
1	0	0	0	1	18	0.0153
1	0	1	0	1	18	0.0153
1	0	0	0	0	15	0.0054
1	0	1	1	1	15	0.0054
1	1	1	0	1	15	0.0054
1	1	1	1	0	15	0.0054
1	1	0	1	1	10	0.0009
1	1	1	1	1	0	0.0000

Table 4.4 : Results for iteration 2 of the original Algorithm 4.1.

Iteration 3

It follows from Table 4.4 that the probability of the first cut is $0.5043 > 0.5 = \rho$. Hence, $\gamma_3 = 28$, which also corresponds to $S(\boldsymbol{X}) = S\big((1,1,0,0,0)\big)$. Proceeding as before we obtain from (49) and (48) $\lambda_3 = -12.8$ and the $\boldsymbol{p}_3$ which is given in Table 4.5.

Note that since the elite sample contains only a single trajectory, (49) can be calculated analytically. Its resulting value is $\lambda_3^* = -\infty$. Apparently, the value $\lambda_3 = -12.8$ already leads to the optimal solution.

Table 4.5 shows that the vector $\boldsymbol{p}_3$ assigns zero probability to all cuts except the first one, which has probability 1 (up to four significant digits). This implies convergence of Algorithm 4.1 to the optimal solution.

(b) The MaxEnt Modification of Algorithm 4.1

We found that the MaxEnt Modification of Algorithm 4.1 converges to the optimal solution after three iterations. The tables are similar to those of the original version of Algorithm 4.1.

(c) Algorithm 4.1 with the modification (55).

Iteration 1

Clearly, this iteration coincides with that of the original Algorithm 4.1.

Iteration 2

: Since the marginal distributions in this case are Bernoulli, we shall denote below $\boldsymbol{p}_t^{(m)}\big((1,1,1,1,1)\big)$ by $\boldsymbol{p}_t^{(m)}$ for brevity.

Here, for the $\boldsymbol{p}_1$ given in table 4.3, we first apply equation (51) to find $\boldsymbol{p}_1^{(m)}$,

$\boldsymbol{X}$					$S(\boldsymbol{X})$	$\boldsymbol{p}_3$
1	1	0	0	0	28	1.0000
1	1	1	0	0	26	0.0000
1	1	0	0	1	21	0.0000
1	1	0	1	0	21	0.0000
1	0	0	1	0	20	0.0000
1	0	1	1	0	20	0.0000
1	0	0	1	1	19	0.0000
1	0	1	0	0	19	0.0000
1	0	0	0	1	18	0.0000
1	0	1	0	1	18	0.0000
1	0	0	0	0	15	0.0000
1	0	1	1	1	15	0.0000
1	1	1	0	1	15	0.0000
1	1	1	1	0	15	0.0000
1	1	0	1	1	10	0.0000
1	1	1	1	1	0	0.0000

Table 4.5 : Results for iteration 3 of the original Algorithm 4.1.

which is $\boldsymbol{p}_1^{(m)} = (0.6054, 0.4210, 0.3364, 0.3026)$. Next we use $\boldsymbol{p}_1^{(m)}$ in (54) to obtain $\widehat{\boldsymbol{p}}_1$. For example, $\widehat{p}_{1,(1,1,0,0,0)}$ is

$$\widehat{p}_{1,(1,1,0,0,0)} = 1 \cdot 0.6054 \cdot (1 - 0.4210) \cdot (1 - 0.3364) \cdot (1 - 0.3026) = 0.1622.$$

The entire vector $\widehat{\boldsymbol{p}}_1$ is given in Table 4.6.

Arguing as in Iteration 2 of the original Algorithm 4.1 we obtain that the total probability of the best (elite) six cuts is $0.525 > 0.5 = \rho$. Hence, we obtain from (47) that the average $S(\boldsymbol{X})$ of the six cuts is $\gamma_2 = 24.11$. Finally, from (49) and (48) we obtain that $\lambda_2 = -0.1584$ and the components of the vector $\boldsymbol{p}_2$ are as given in Table 4.6.

Iteration 3 and Iteration 4

The Results for iteration 3 and Iteration 4 are summarized in Table 4.7 and Table 4.8, respectively.

It follows that after four iterations modification (55) of Algorithm 4.1 has converged to the optimal solution.

$\boldsymbol{p}_1^{(m)}$					$S(\boldsymbol{X})$	$\widehat{\boldsymbol{p}}_1$	$\boldsymbol{p}_2$
1	0.6054	0.4210	0.3364	0.3026			
$\boldsymbol{X}$							
1	1	0	0	0	28	0.1622	0.4085
1	1	1	0	0	26	0.1180	0.2164
1	1	0	0	1	21	0.0704	0.0585
1	1	0	1	0	21	0.0822	0.0683
1	0	0	1	0	20	0.0536	0.0380
1	0	1	1	0	20	0.0390	0.0276
1	0	0	1	1	19	0.0233	0.0141
1	0	1	0	0	19	0.0769	0.0466
1	0	0	0	1	18	0.0459	0.0237
1	0	1	0	1	18	0.0334	0.0172
1	0	0	0	0	15	0.1057	0.0340
1	0	1	1	1	15	0.0169	0.0054
1	1	1	0	1	15	0.0512	0.0164
1	1	1	1	0	15	0.0598	0.0192
1	1	0	1	1	10	0.0357	0.0052
1	1	1	1	1	0	0.0259	0.0008

Table 4.6 : Results for iteration 2 of Algorithm 4.1 with the modification (55).

$\boldsymbol{p}_2^{(m)}$					$S(\boldsymbol{X})$	$\widehat{\boldsymbol{p}}_2$	$\boldsymbol{p}_3$
1	0.7934	0.3497	0.1787	0.1414			
$\boldsymbol{X}$							
1	1	0	0	0	28	0.3638	0.7623
1	1	1	0	0	26	0.1957	0.2006
1	1	0	0	1	21	0.0599	0.0103
1	1	0	1	0	21	0.0792	0.0136
1	0	0	1	0	20	0.0206	0.0025
1	0	1	1	0	20	0.0111	0.0013
1	0	0	1	1	19	0.0034	0.0003
1	0	1	0	0	19	0.0510	0.0043
1	0	0	0	1	18	0.0156	0.0009
1	0	1	0	1	18	0.0084	0.0005
1	0	0	0	0	15	0.0948	0.0019
1	0	1	1	1	15	0.0018	0.0000
1	1	1	0	1	15	0.0322	0.0006
1	1	1	1	0	15	0.0426	0.0009
1	1	0	1	1	10	0.0130	0.0000
1	1	1	1	1	0	0.0070	0.0000

Table 4.7 : Results for iteration 3 of Algorithm 4.1 with the modification (55).

$\boldsymbol{p}_3^{(m)}$					$S(\boldsymbol{X})$	$\widehat{\boldsymbol{p}}_3$	$\boldsymbol{p}_4$
1	0.9883	0.2082	0.0186	0.0127			
$\boldsymbol{X}$							
1	1	0	0	0	28	0.7582	1.0000
1	1	1	0	0	26	0.1994	0.0000
1	1	0	0	1	21	0.0098	0.0000
1	1	0	1	0	21	0.0144	0.0000
1	0	0	1	0	20	0.0002	0.0000
1	0	1	1	0	20	0.0000	0.0000
1	0	0	1	1	19	0.0000	0.0000
1	0	1	0	0	19	0.0024	0.0000
1	0	0	0	1	18	0.0001	0.0000
1	0	1	0	1	18	0.0000	0.0000
1	0	0	0	0	15	0.0090	0.0000
1	0	1	1	1	15	0.0000	0.0000
1	1	1	0	1	15	0.0026	0.0000
1	1	1	1	0	15	0.0038	0.0000
1	1	0	1	1	10	0.0002	0.0000
1	1	1	1	1	0	0.0000	0.0000

Table 4.8 : Results for iteration 4 of Algorithm 4.1 with modification (55).

5 The Stochastic MCE method

As mentioned earlier, in the stochastic MaxEnt and MinxEnt programs we shall assume that

$$\mathbb{E}_{\boldsymbol{p}} S(\boldsymbol{X}), \quad \boldsymbol{X} \in \mathcal{X}$$

(see (24)) is not available analytically because of the huge size of $\mathcal{X}$.

Consider first the stochastic (sample average) version of the MaxEnt program (56). It can be written as

$$\begin{aligned} & \max_{\widetilde{\boldsymbol{p}}_t} \widehat{\mathcal{H}}(\widetilde{\boldsymbol{p}}_t) = \min_{\widetilde{\boldsymbol{p}}_t} \textstyle\sum_{k=1}^N \widetilde{\boldsymbol{p}}_t(\boldsymbol{X}_k) \ln \widetilde{\boldsymbol{p}}_t(\boldsymbol{X}_k) \\ \text{s. t.} \quad & \textstyle\sum_{k=1}^N S(\boldsymbol{X}_k) \widetilde{\boldsymbol{p}}_t(\boldsymbol{X}_k) = \widehat{\gamma}_t, \\ & \textstyle\sum_{k=1}^N \widetilde{\boldsymbol{p}}_t(\boldsymbol{X}_k) = 1, \end{aligned} \tag{59}$$

where $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ is a sample taken from the pdf $\widetilde{\boldsymbol{p}}_{t-1} = \{\widetilde{p}_{t,i_1 \cdots i_n};\ i_r = 1, \ldots, m;\ r = 1, \ldots, n\}$, and $\widetilde{\boldsymbol{p}}_t$ denotes the estimate of $\boldsymbol{p}_t$ in (56).

It is readily seen that the solution of (59) is

$$\widetilde{p}_{t,i_1 \cdots i_n} = \frac{\sum_{k=1}^N I_{\{\boldsymbol{X}_k = i_1 \cdots i_n\}} \exp\{-\widehat{\lambda}_t S(\boldsymbol{X}_k)\}}{\sum_{k=1}^N \exp\{-\widehat{\lambda}_t S(\boldsymbol{X}_k)\}}, \tag{60}$$

where $\widehat{\lambda}_t$, the estimate of λ_t (see (49)) can be obtained from the solution of the following non-linear equation

$$\frac{\sum_{k=1}^{N} S(\boldsymbol{X}_k) \exp\{-\lambda S(\boldsymbol{X}_k)\}}{\sum_{k=1}^{N} \exp\{-\lambda S(\boldsymbol{X}_k)\}} = \widehat{\gamma}_t. \tag{61}$$

Consider next the stochastic counterpart of the MinxEnt program (55), which can be written as

$$\begin{aligned} &\min_{\widetilde{\boldsymbol{p}}_t} \widehat{\mathcal{D}}(\widetilde{\boldsymbol{p}}_t|\widetilde{\boldsymbol{p}}_{t-1}) = \min_{\widetilde{\boldsymbol{p}}_t} \sum_{k=1}^{N} \widetilde{\boldsymbol{p}}_t(\boldsymbol{X}_k) \ln \frac{\widetilde{\boldsymbol{p}}_t(\boldsymbol{X}_k)}{\widetilde{\boldsymbol{p}}_{t-1}(\boldsymbol{X}_k)} \\ \text{s. t.} \quad &\sum_{k=1}^{N} S(\boldsymbol{X}_k)\widetilde{\boldsymbol{p}}_t(\boldsymbol{X}_k) = \widehat{\gamma}_t, \\ &\sum_{k=1}^{N} \widetilde{\boldsymbol{p}}_t(\boldsymbol{X}_k) = 1. \end{aligned} \tag{62}$$

Its solution is

$$\widetilde{p}_{t,i_1\cdots i_n} = \frac{\sum_{k=1}^{N} I_{\{\boldsymbol{X}_k = i_1\cdots i_n\}} \widetilde{\boldsymbol{p}}_{t-1}(\boldsymbol{X}_k) \exp\{-\widehat{\lambda}_t S(\boldsymbol{X}_k)\}}{\sum_{k=1}^{N} \widetilde{\boldsymbol{p}}_{t-1}(\boldsymbol{X}_k) \exp\{-\widehat{\lambda}_t S(\boldsymbol{X}_k)\}}, \tag{63}$$

where $\widehat{\lambda}_t$ is the solution of

$$\frac{\sum_{k=1}^{N} \widetilde{\boldsymbol{p}}_{t-1}(\boldsymbol{X}_k) S(\boldsymbol{X}_k) \exp\{-\lambda S(\boldsymbol{X}_k)\}}{\sum_{k=1}^{N} \widetilde{\boldsymbol{p}}_{t-1}(\boldsymbol{X}_k) \exp\{-\lambda S(\boldsymbol{X}_k)\}} = \widehat{\gamma}_t. \tag{64}$$

Note that the MaxEnt updating formulas (60) and (61) can be viewed as particular cases of their MinxEnt counterparts (63) and (64), respectively, namely with $\widetilde{\boldsymbol{p}}_{t-1}(\boldsymbol{x})$ being uniformly distributed.

We shall use below the MaxEnt updating formulas (60) and (61). We found that such a MaxEnt based Algorithm is superior than its MinxEnt counterpart using updating formulas (63) and (64). This can be explained by the fact that the MinxEnt program (62) contains a high dimensional likelihood ratio term

$$\frac{\widetilde{\boldsymbol{p}}_t(\boldsymbol{X}_k)}{\widetilde{\boldsymbol{p}}_{t-1}(\boldsymbol{X}_k)},$$

and it is well known [10] that using high dimensional likelihood ratio terms one obtains *inaccurate* estimates, that is, the optimal solution (63) and (64) of the program (62) could be quite *unreliable*.

Note also that in the analogy to the simulated annealing algorithm,

$$T_t = \frac{1}{\widehat{\lambda}_t}$$

may be called the "annealing temperature". *In contrast to simulated annealing, where $\widehat{\lambda}_t$ is chosen in advance, here, in MCE it is updated adaptively.*

To generate trajectories for COP's we shall use the marginal pdf's of $\widetilde{\boldsymbol{p}}_t(\boldsymbol{x})$, similar as we did with the marginal pdf's of $\boldsymbol{p}_t(\boldsymbol{x})$. In particular, in analogy to (51), the marginal probabilities, denoted as $\breve{p}_{t,i_k}$, can be calculated as

$$\breve{p}_{t,i_k} = \sum_{i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_n} \widetilde{p}_{t,i_1\cdots i_n};\ i_k = 1,\ldots,m;\ k = 1,\ldots,n, \tag{65}$$

and similarly for the stochastic version of (53).

In analogy to (52) we denote the n-dimensional vector of marginal pdf's (probabilities) at iteration t by $\widetilde{\boldsymbol{p}}_t^{(m)}$, that is

$$\widetilde{\boldsymbol{p}}_t^{(m)} = \{\breve{p}_{t,i_k};\ i_k = 1,\ldots,m;\ k = 1,\ldots,n\}$$

and in analogy to (54) we define the induced joint pdf corresponding to $\boldsymbol{x} = (i_1,\ldots,i_n)$ as

$$\breve{p}_{t,i_1\ldots,i_n} = \breve{p}_{t,i_1}\cdots\breve{p}_{t,i_n}. \tag{66}$$

As mentioned, the starting point in the methodology of the MCE method is similar to that in the CE, namely we associate with the optimization problem (1) a meaningful stochastic program, called the ASP, and we introduce randomness either at the *nodes* of the graph or at its *edges*, depending on whether we deal with a stochastic node network (SNN) or a stochastic edge network (SEN). Similarly to the deterministic MCE we update the sequence $\{\widehat{\gamma}_t, \widetilde{\boldsymbol{p}}_t\}$ in the stochastic MCE method as follows:

1. **Adaptive updating of $\widehat{\gamma}_t$.** Let $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_N$ be a random sample from $\widetilde{\boldsymbol{p}}_{t-1}(\boldsymbol{x})$ and let $S(\boldsymbol{X}_i)$, $i = 1,\ldots,N$ be the associated function values. As in the CE method we order them from smallest to largest as $S_{(1)} \le \ldots \le S_{(N)}$ and in analogy to (47), calculate $\widehat{\gamma}_t$ by

$$\widehat{\gamma}_t = \frac{1}{\lceil \rho N \rceil} \sum_{i=\lfloor (1-\rho)N \rfloor + 1}^{N} S_{(i)}, \tag{67}$$

   where typically $0.01 \le \rho \le 0.1$. Note that $\widehat{\gamma}_t$ presents the sample average of the $\lceil \rho N \rceil$ largest (elite) values of $S(\boldsymbol{X}_i),\ i = 1,\ldots,N$.

   As an alternative to $\widehat{\gamma}_t$ one can consider the estimator of the $(1-\rho)$-*quantile* of $S(\boldsymbol{X}_k),\ k = 1,\ldots,N$, which is given by (3). If not otherwise stated we shall use (67) for COP's.

2. **Adaptive updating of $\widetilde{\boldsymbol{p}}_t$.** For fixed $\widehat{\gamma}_t$ and $\widetilde{\boldsymbol{p}}_{t-1}$, derive $\widehat{\lambda}_t$ from the numerical solution of (61) and deliver $\widetilde{\boldsymbol{p}}_t$ according to (60). Calculate $\widetilde{\boldsymbol{p}}_t^{(m)}$ according to (65) and $\breve{\boldsymbol{p}}_t$ according to (66) and use the former for trajectory generation.

As in CE, instead of using the parameter vector $\widetilde{\boldsymbol{p}}_t^{(m)}$ obtained from (65) directly we shall use its following *smoothed* version

$$\overline{\boldsymbol{p}}_t^{(m)} = \alpha\widetilde{\boldsymbol{p}}_t^{(m)} + (1-\alpha)\widetilde{\boldsymbol{p}}_{t-1}^{(m)}, \tag{68}$$

where as before α ($\alpha < 1$) is called the *smoothing parameter*.

The resulting MCE algorithm, which is is similar to the CE algorithm 2.1 and which is based on the and the updating formulas (60) and (61) of the MaxEnt program (59) can be summarized as follows.

Algorithm 5.1 (Main MCE Algorithm for COP's)

1. *Choose some* $\overline{\boldsymbol{p}}_0^{(m)}$, *say* $\overline{\boldsymbol{p}}_0^{(m)} = \boldsymbol{u}$ *(with equal elements). Set* $t = 1$ *(level counter).*

2. *Generate a sample* $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ *from the pdf* $\overline{\boldsymbol{p}}_{t-1}^{(m)}$, *calculate the associated function values* $S(\boldsymbol{X}_1), \ldots, S(\boldsymbol{X}_N)$ *and use their* $\lceil \rho N \rceil$ *largest (elite) values,* $S_{(i)},\ i = \lfloor (1-\rho)N \rfloor + 1, \ldots, N$ *to calculate* $\widehat{\gamma}_t$ *according to (67).*

3. *Use the* **same** *sample* $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$. *Deliver* $\widehat{\lambda}_t$ *from the numerical solution of (61) and* $\widetilde{\boldsymbol{p}}_t = \{\widetilde{p}_{t,i_1 \cdots i_n}, i_k = 1, \ldots, m;\ k = 1, \ldots, n\}$ *according to (60).*

4. *Apply (68) to smooth out the vector* $\widetilde{\boldsymbol{p}}_t^{(m)}$ *and obtain* $\overline{\boldsymbol{p}}_t^{(m)}$.

5. *If for some* $t \geq d$, *say* $d = 5$,

$$\widehat{\gamma}_t = \widehat{\gamma}_{t-1} = \cdots = \widehat{\gamma}_{t-d}, \tag{69}$$

   then **stop** *(let* T *denote the final iteration); otherwise set* $t = t + 1$ *and reiterate from step 2.*

Note that similar to CE, as soon as the sample size N, the stopping parameter d, the rarity parameter ρ and the smoothing parameter α are specified in advance, the MCE Algorithm 5.1 is "self-tuning". Note also that similarly to the FACE (fully adaptive CE) algorithm (see [3]) one can easily design a fully adaptive version of MinxEnt.

Remark 5.1 Comparing (60) with the CE updating (9) we note that they coincide, provided $\varphi[S(\boldsymbol{X}_k, \eta)] = e^{-\eta S(\boldsymbol{X}_k)}$ and $\eta = \widehat{\lambda}_t$. Moreover, recall that using (9) for updating in the CE method entails quite slow convergence. In contrast, using either (7) or (10) (both contain $I_{\{S(\boldsymbol{X}_k) \geq \widehat{\gamma}_t\}}$, i.e. both use only the elite sample for updating) results in a convergence which is quite quick and accurate.

In analogy to CE (see (4)), we can modify (60) such that only the elite sample will be used instead of the entire sample. Thus we obtain

$$\widetilde{p}_{t,i_1 \cdots i_n} = \frac{\sum_{k=1}^N I_{\{\boldsymbol{X}_k = i_1 \cdots i_n\}} I_{\{S(\boldsymbol{X}_k) \geq \widehat{\gamma}_t\}} \exp\{-\widehat{\lambda}_t S(\boldsymbol{X}_k)\}}{\sum_{k=1}^N I_{\{S(\boldsymbol{X}_k) \geq \widehat{\gamma}_t\}} \exp\{-\widehat{\lambda}_t S(\boldsymbol{X}_k)\}}, \tag{70}$$

where as before $\widehat{\lambda}_t$ is derived from (61).

Note also that (70) resembles (10) for CE. There is, however, a crucial difference between the two, since in the former, $\widehat{\lambda}_t$ is derived from the *optimal solution* of a well-defined dual program, while in the latter, the parameter η is chosen *heuristically*.

6 Rare-Event Probability Estimation

6.1 Light-tailed distribution

Consider estimation of a rare-event probability ℓ given as

$$\ell = \mathbb{E}_h \left\{ I_{\{S(\boldsymbol{X}) \geq \gamma\}} \right\}. \tag{71}$$

Using the importance sampling (IS) density f^* obtained from the MinxEnt program (P_0) we can write ℓ as

$$\ell = \mathbb{E}_{f^*} \left\{ I_{\{S(\boldsymbol{X}) \geq \gamma\}} \frac{h(\boldsymbol{X})}{f^*(\boldsymbol{X})} \right\}. \tag{72}$$

The corresponding likelihood ratio (LR) estimate is

$$\widetilde{\ell} = \frac{1}{N} \sum_{i=1}^{N} \left\{ I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} \frac{h(\boldsymbol{X}_i)}{f^*(\boldsymbol{X}_i)} \right\}, \tag{73}$$

where $\boldsymbol{X}_i \sim f^*(\boldsymbol{x})$.

Clearly the accuracy of $\widetilde{\ell}$ depends on f^*. We consider below only the case where f^* is derived from the MinxEnt program (P_0) with a *single* constrained, namely for

$$\mathbb{E}_f\{S(\boldsymbol{X})\} = \gamma, \tag{74}$$

where γ is given in (72). In this case the optimal solution is

$$f^*(\boldsymbol{x}, \lambda^*) = \frac{h(\boldsymbol{x}) \exp\{-S(\boldsymbol{x})\lambda^*\}}{\mathbb{E}_h \exp\{-S(\boldsymbol{X})\lambda^*\}}, \tag{75}$$

where λ satisfies

$$\frac{\mathbb{E}_h S(\boldsymbol{X}) \exp\{-\lambda S(\boldsymbol{X})\}}{\mathbb{E}_h \exp\{-\lambda S(\boldsymbol{X})\}} = \gamma. \tag{76}$$

The procedure for estimating ℓ in (71) can be summarized is as follows:

- Solve the MinxEnt program (P_0) with the single constraint (74) and deliver $f^*(\boldsymbol{x})$ as in (75).
- Take a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from $f^*(\boldsymbol{x})$ and estimate ℓ using (73).

As an example, let $S(\boldsymbol{X}) = X$, and assume that $X \sim \mathcal{N}(\mu, \sigma^2)$. In this case ($\mathrm{P}_0$) can be solved analytically. It is easy to show that the optimal pdf is $f^*(x) = \mathcal{N}(\gamma, \sigma^2)$ and the SCV (squared coefficient of variation) of $\widetilde{\ell}(\gamma)$, denoted $\kappa^2(\gamma)$ is $\kappa^2(\gamma) = O(\gamma)$.

It crucial to realize that unlike the standard CE method for rare-event simulation, which presents a simulation-based adaptive procedure for finding the optimal parameters in the IS pdf, here solving MinxEnt program we find *analytically* (without iterations) the optimal joint IS pdf f^*. Note, however, that in

contrast to CE, such joint pdf f^* is typically a complex one. So, generating random variables from f^* in high dimensions might be a difficult task, in particular when f^* is multidimensional continuous pdf. Note also that in same particular case the Gibs sampler and the *sampling importance resampling* (SIR) methods [5], can be used to generate from $f^*(\boldsymbol{x})$, even in high dimension, while in low dimension, say for $n \leq 5$, one can generate from $f^*(\boldsymbol{x})$ using the acceptance-rejection method. Note finally that in high dimension one can use instead of the joint pdf $f^*(\boldsymbol{x})$ its approximation, $f^*_{(m)}(\boldsymbol{x})$ equal to the product the marginal pdf's $f^*_{(m)}(x_i),\ i = 1, \ldots, n$ of $f^*(\boldsymbol{x})$, that is

$$f^*_{(m)}(\boldsymbol{x}) = \prod_{i=1}^{n} f^*_{(m)}(x_i),$$

and thus, estimate ℓ as

$$\widehat{\ell} = \frac{1}{N} \sum_{i=1}^{N} \left\{ I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} \frac{h(\boldsymbol{X}_i)}{\prod_{j=1}^{n} f^*_{(m)}(X_j)} \right\}. \tag{77}$$

Here the sample $\boldsymbol{X}_i$ is taken from $\prod_{j=1}^{n} f^*_{(m)}(x_j)$. Clearly, the estimate $\widetilde{\ell}$ based on $f^*(\boldsymbol{x})$ is typically more accurate than $\widehat{\ell}$ based on $f^*_{(m)}(\boldsymbol{x})$. It is readily seen that in the particular case, where $S(\boldsymbol{x})$ is separable with respect to each component of the vector $\boldsymbol{x}$, one has that $f^*(\boldsymbol{x}) = f^*_{(m)}(\boldsymbol{x})$ and, thus both estimates $\widetilde{\ell}$ and $\widehat{\ell}$ coincide.

6.2 Heavy-tailed distribution

Consider again the MinxEnt program (P_0) with the single constraint (74). Note that for heavy-tailed pdf such original MinxEnt program (P_0) fails, since (76) has no solution (the moment generating function for a random variable with heavy-tailed pdf does not exist). To overcome this difficulty one can use several alternatives. We shall consider the following two approaches (a) based on Csiszer's ϕ-divergence family and (b) the so-called *transformed* MinxEnt program (P_0).

(a) Csiszer's family of ϕ-divergence is defined as [6]

$$D(f|h) = \mathbb{E}_h \phi \left\{ \frac{f(\boldsymbol{X})}{h(\boldsymbol{X})} \right\}, \tag{78}$$

where ϕ is twice-differentiable convex function, for which $\phi(1) = 0$. As particular cases we consider

1. $\phi(x) = x \ln x$. We have $D(f|h) = \mathbb{E}_f \left\{ \ln \frac{f(\boldsymbol{X})}{h(\boldsymbol{X})} \right\}$, which is Kullback-Leibler's divergence.
2. $\phi(x) = x^k, k \geq 2$. We have

$$D(f|h) = \mathbb{E}_h \left\{ \frac{f(\boldsymbol{X})}{h(\boldsymbol{X})} \right\}^k = \mathbb{E}_f \left\{ \frac{f(\boldsymbol{X})}{h(\boldsymbol{X})} \right\}^{k-1}.$$

3. $\phi(x) = x^{-k}, k \geq 1$. We have

$$D(f|h) = \mathbb{E}_h \left\{ \frac{h(\boldsymbol{X})}{f(\boldsymbol{X})} \right\}^k = \mathbb{E}_f \left\{ \frac{h(\boldsymbol{X})}{f(\boldsymbol{X})} \right\}^{k+1}.$$

   Note that for $k = 1$, $\mathbb{E}_f \left\{ \frac{h(\boldsymbol{X})}{f(\boldsymbol{X})} \right\}^2$ corresponds to variance minimization under f.

4. $\phi(x) = \frac{x^\alpha - x}{\alpha - 1}, \alpha > 0, \alpha \neq 1$. We have $D(f|h) = \frac{1}{\alpha - 1} \left\{ \mathbb{E}_f f(\boldsymbol{X})^{\alpha-1} h(\boldsymbol{X})^{1-\alpha} - 1 \right\}$, which is Havrada-Charvat divergence. Note that as $\alpha \to 1$, Havrada-Charvat divergence $D(f|h)$ approaches to Kullback-Leibler's divergence.

5. $\phi(x) = 1 - x^{1/2}$. We have $D(f|h) = \mathbb{E}_f \left\{ \frac{f(\boldsymbol{X})^{1/2} - h(\boldsymbol{X})^{1/2}}{f(\boldsymbol{X})^{1/2}} \right\}^2$, which is Hellingers's divergence.

To proceed, note that a particular divergence measure should satisfy

1. $D(f|h) \geq 0$.
2. $D(f|h) = 0$ if and only if $f = h$.
3. When $D(f|h)$ is minimized subject to given linear constraints, using Lagrange's method, the resulting probability f^* should be non-negative, that is $f^* \geq 0$

As possible candidates for heavy tails satisfying the above conditions one can consider either Havrada-Charvat's or Hellinger's divergence.

(b) As for a motivating example consider estimation

$$\ell = \mathbb{E}_h \left\{ I_{\{X \geq \gamma\}} \right\}, \tag{79}$$

where $X \sim h(x) = \text{Weibull}(u, a)$, that is

$$h(x) = a u^{-1} (u^{-1} x)^{a-1} e^{(u^{-1}x)^a}. \tag{80}$$

Note that for $a < 1$, Weibull(u, a) is a heavy-tailed pdf. As mentioned, in this case the original MinxEnt program (P_0) fails.

To proceed we use the inverse-transform (IT) method, that is we write a random variable X as

$$X = H^{-1}(Y),$$

where Y is a uniform random variable, that is $Y \sim \mathcal{U}(0, 1)$. For $X \sim \text{Weibull}(u, a)$ we have

$$X = u(-\ln(1 - Y))^{1/a}. \tag{81}$$

We shall show next that using the IT method $X = H^{-1}(Y)$ one can make the single constraint MinxEnt program (P_0) applicable for estimating ℓ in (79).

Indeed, using (81), we can write (79) as

$$\ell = \mathbb{E}_{\mathcal{U}} \left\{ I_{\{u(-\ln(1-Y))^{1/a} \geq \gamma\}} \right\}, \tag{82}$$

which is the same as

$$\ell = \mathbb{E}_{\mathcal{U}} \left\{ I_{\{Y \geq e^{-(\frac{\gamma}{u})^a}\}} \right\}, \tag{83}$$

where $Y \sim \mathcal{U}(0,1)$. Note that (83) expresses ℓ in terms of the uniform $\mathcal{U}(0,1)$ pdf, for which the associated single constrained MinxEnt program (P_0) clearly *has a solution*. It can be written as

$$(\mathrm{P}_1) \qquad \begin{aligned} &\min_{f(x)} \left\{ \mathcal{D}(f|h) = \mathbb{E}_f \ln \tfrac{f(X)}{h(X)} \right\} \\ &\text{s.t.} \quad \mathbb{E}_f X = e^{-(\frac{\gamma}{u})^a}, \\ &\int f(x) \mathrm{d}x = 1, \end{aligned} \tag{84}$$

where $h(x) = \mathcal{U}(0,1)$. It is not difficult to see that the solution of (P_1), which we call, the *transformed* MinxEnt program, is

$$f^*(x, \lambda^*) = \frac{\exp(\lambda^* x)}{C(\lambda^*)}, \ \lambda^* = \frac{1}{\gamma}, \ 0 \leq x \leq 1, \tag{85}$$

where $C(\lambda^*)$ is the normalization constant. Note that $f^*(x, \lambda^*)$ presents a truncated exponential pdf (with a positive exponent).

More generally. Let $H(x)$ denotes the cdf of the random variable X, then (83) can be extended as

$$\ell = \mathbb{E}_{\mathcal{U}} \left\{ I_{\{Y \geq H(\gamma)\}} \right\} \tag{86}$$

and the program (P_1) generalizes as

$$(\mathrm{P}_2) \qquad \begin{aligned} &\min_{f(x)} \left\{ \mathcal{D}(f|h) = \mathbb{E}_f \ln \tfrac{f(X)}{h(X)} \right\} \\ &\text{s.t.} \quad \mathbb{E}_f X = H(\gamma), \\ &\int f(x) \mathrm{d}x = 1. \end{aligned} \tag{87}$$

Its solution $f^*(x, \lambda^*)$ extends (85) and presents a truncated exponential family pdf.

With this to hand we can define a general *transformed* multi-dimensional program (P_0) and use its solution $f^*(\boldsymbol{x})$ as an importance sampling pdf in (73). To proceed, we write $S(\boldsymbol{X}) = S(X_1, \ldots, X_n)$ as $S(\boldsymbol{X}) =$
$S[H_1^{-1}(X_1), \ldots, (H_n^{-1}(X_n)]$, where as before $X = H^{-1}(Y)$ and $Y \sim \mathcal{U}(0,1)$. The associated transformed single constrained program (P_0) can be written as

$$(\mathrm{P}_3) \qquad \begin{aligned} &\min_{f(\boldsymbol{x})} \left\{ \mathcal{D}(f|h) = \mathbb{E}_f \ln \tfrac{f(\boldsymbol{X})}{h(\boldsymbol{X})} \right\} \\ &\text{s.t.} \quad \mathbb{E}_f S[H_1^{-1}(X_1), \ldots, H_n^{-1}(X_n)] = \gamma, \\ &\int f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1, \end{aligned} \tag{88}$$

where $h(\boldsymbol{x})$ has independent components, each distributed $\mathcal{U}(0,1)$, that is $h(\boldsymbol{x}) = h_1(x_1)\cdots h_n(x_n)$, $h_1(x_1) = \cdots = h_n(x_n) = \mathcal{U}(0,1)$.

Note that solution $f^*(\boldsymbol{x})$ of the transformed MinxEnt program (P_3)

1. Is given again by (75), (76), where $S(\boldsymbol{X}) = S[H_1^{-1}(X_1), \ldots, (H_n^{-1}(X_n)]$.
2. It can be used in the (73) to estimate probabilities of rare events for both, *light-tailed and heavy-tailed* distributions.

As mentioned, for general non-separable sample functions $S(\boldsymbol{X})$, the optimal multi-dimensional $f^*(\boldsymbol{x})$ in (75), (76) typically has a complex form, so generating samples from such $f^*(\boldsymbol{x})$ might be a formidable task. For this reason we can use in parallel to the *non-parametric* $f^*(\boldsymbol{x})$ in (75), (76) its *parametric* MinxEnt version or the parametric Hellinger's entropy version. In both cases we assume that $f(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{v})$ and $h(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{u})$. In particular the parametric MinxEnt program can be written as

$$\begin{aligned} & \min_{\boldsymbol{v}} \left\{ \mathcal{D}(\boldsymbol{v}|\boldsymbol{u}) = \int \ln \tfrac{f(\boldsymbol{x},\boldsymbol{v})}{f(\boldsymbol{x},\boldsymbol{u})} f(\boldsymbol{x},\boldsymbol{v}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_{\boldsymbol{v}} \ln \tfrac{f(\boldsymbol{X},\boldsymbol{v})}{f(\boldsymbol{X},\boldsymbol{u})} \right\} \\ \text{s.t.} \quad & \int S_j(\boldsymbol{x}) f(\boldsymbol{x},\boldsymbol{v}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_{\boldsymbol{v}} S_j(\boldsymbol{X},\boldsymbol{v}) = \gamma_j, \;\; j = 1,\ldots,k, \\ & \int f(\boldsymbol{x},\boldsymbol{v}) \mathrm{d}\boldsymbol{x} = 1, \;\; \int f(\boldsymbol{x},\boldsymbol{u}) \mathrm{d}\boldsymbol{x} = 1. \end{aligned} \tag{89}$$

Here $f(\boldsymbol{x},\boldsymbol{u})$ can be viewed again as the prior pdf and $f(\boldsymbol{x},\boldsymbol{v}^*)$, where $\boldsymbol{v}^*$ is the optimal solution of (89), as the optimal parametric posterior pdf.

7 Numerical Results

In this section we present numerical results with the MCE method, for the maximal cut (max-cut) problem and the TSP. Note that the former presents a typical SNN-type problem, while the later - a typical SEN-type problem. For some instances of the TSP we compare the efficiency of MCE and CE. All runs were made on a 600MHz PIII with 256MB RAM using a Matlab code.

7.1 The Max-Cut Problem

The max-cut is known as an NP-hard. It can be formulated as follows. Given a weighted graph, $G(V,E)$, with a set of nodes $V = \{1,\ldots,n\}$ and a set of edges E, partition the nodes of the graph into two subsets V_1 and V_2 such that the sum of the weights of the edges going from one subset to the other is maximized. We denote by c_{ij} the weight going from node i to to node j. Formally, a *cut* is a partition $\{V_1, V_2\}$ of V. For example if $V = \{1,\ldots,6\}$, then $\{\{1,3,4\},\{2,5,6\}\}$ is a possible cut. The *cost* of a cut is sum of the weights across the cut. As an

example, consider the following 6×6 cost matrix

$$C = \begin{pmatrix} 0 & c_{12} & c_{13} & 0 & 0 & 0 \\ c_{21} & 0 & c_{23} & c_{24} & 0 & 0 \\ c_{31} & c_{32} & 0 & c_{34} & c_{35} & 0 \\ 0 & c_{42} & c_{43} & 0 & c_{45} & c_{46} \\ 0 & 0 & c_{53} & c_{54} & 0 & c_{56} \\ 0 & 0 & 0 & c_{64} & c_{65} & 0 \end{pmatrix} . \tag{90}$$

For example the cost of the cut $\{\{1,3,4\},\{2,5,6\}\}$ is

$$c_{12} + c_{32} + c_{42} + c_{45} + c_{53} + c_{46} .$$

It is convenient to represent a cut via a *cut vector* $\boldsymbol{x} = (x_1, \ldots, x_n)$, where $x_i = 1$ if node i belongs to same set as node 1, and $x_i = 0$ otherwise. By definition $x_1 = 1$. For example, the cut $\{\{1,3,4\},\{2,5,6\}\}$ can be represented via the cut vector $(1,0,1,1,0,0)$.

A Synthetic max-cut

In order to verify the accuracy of the MCE method we construct an artificial graph such that the solution is available in advance. In particular, for $m \in \{1,\ldots,n\}$ consider the following symmetric cost matrix:

$$C = \begin{pmatrix} Z_{11} & B_{12} \\ B_{21} & Z_{22} \end{pmatrix} , \tag{91}$$

where Z_{11} is an $m \times m$ symmetric matrix in which all elements above the diagonal are generated from an arbitrary distribution with support $[a,b]$ (and all elements below it follow by symmetry), Z_{22} is an $(n-m) \times (n-m)$ symmetric matrix which is generated similarly to Z_{11}, and all the other elements are c, apart from the diagonal elements, which are naturally 0.

It is readily seen that for $c > b(n-m)/m$ the optimal cut is given by

$$V_1^* = \{1,\ldots,m\} \quad \text{and} \quad V_2^* = \{m+1,\ldots,n\} , \tag{92}$$

and the optimal value of the cut is

$$\gamma^* = c\, m\, (n-m) . \tag{93}$$

We shall study two variants of Algorithm 5.1, namely with the updating formulas for $\boldsymbol{p}^*$ given by (60), and by (70), respectively. Note that in both variants $\widehat{\lambda}^*$ is updated according to (61). We call the first variant *regular MCE*, and the second — *elite MCE*.

Tables 7.1 and 7.2 present a typical evolutions of the regular MCE and the elite MCE, respectively for the above synthetic problem with $n = 100$, $m = 50$, $c = 1$ and $\boldsymbol{Z}_{11} = \boldsymbol{Z}_{22} = \boldsymbol{0}$, for which it is readily seen that the optimal cut is $\{V_1 = (1,2,\ldots,50), V_2 = (51,52,\ldots,100)\}$ and the optimal $\gamma^* =$

$50 \cdot (100-50) = 2,500$. Here t denotes the iteration number, $\hat{\gamma}_t$ is given by (47), $S_{t,(N)}$ is the best cut value sampled at iteration t, $p\uparrow_{min} = \min\{p_{t,i} \mid p_{t,i} \geq 0.5,\ i = 1,2,\ldots,n\}$ and $p\downarrow_{max} = \max\{p_{t,i} \mid p_{t,i} \leq 0.5,\ i = 1,2,\ldots,n\}$. For both variants we set $N = 5n = 500$, $\alpha = 0.7$, $\rho = 0.1$ and $d = 5$. The CPU time in both cases was about 7 seconds.

It follows that for both variants Algorithm 5.1 performs fast and accurate.

t	$\hat{\gamma}_t$	$S_{t,(N)}$	$p\uparrow_{min}$	$p\downarrow_{max}$
1	1296.84	1358	0.5044	0.4986
2	1328.88	1502	0.5015	0.4969
3	1385.00	1502	0.5005	0.4862
4	1478.56	1640	0.5519	0.4706
5	1608.64	1828	0.5106	0.4629
6	1738.68	1880	0.5230	0.4892
7	1875.84	2010	0.5099	0.4441
8	2026.80	2130	0.6741	0.4182
9	2167.64	2262	0.7552	0.2962
10	2301.20	2402	0.8300	0.2376
11	2406.32	2500	0.8666	0.1169
12	2475.00	2500	0.9372	0.0515
13	2500.00	2500	0.9812	0.0155
14	2500.00	2500	0.9944	0.0046
15	2500.00	2500	0.9983	0.0014
16	2500.00	2500	0.9995	0.0004

Table 7.1 : Typical evolution of the regular MCE Algorithm 5.1 for a synthetic max-cut problem with 100 nodes.

Tables 7.3 and 7.4 present performance statistics of the regular MCE and the elite MCE, respectively as function of the sample size N for the above synthetic problem with $n = 100$, $m = 50$, $c = 5$. Each experiment is based on 10 independent runs, with $\alpha = 0.7$ and varying ρ, such that the elite sample contains 30 trajectories. Here ε denotes the relative (average, minimum and maximum) accuracy, which is calculated as

$$\varepsilon = \frac{\gamma^* - \widehat{\gamma}_T}{\gamma^*}.$$

Here $\widehat{\gamma}_T$ denotes to the value of the function $S(\boldsymbol{x})$ (average, minimum and maximum, respectively) found by Algorithm 5.1 before stopping, T denotes the number of iterations needed before Algorithm 5.1 stops and CPU denotes the time in seconds. Both experiments were run for $\boldsymbol{Z}_{11}$ and $\boldsymbol{Z}_{22}$ being distributed $Beta(\alpha, \beta, a, b) = Beta(100, 1, 1, 5)$. It is readily seen that in this case the optimal cut is $\{V_1 = (1, 2, \ldots, 50), V_2 = (51, 52, \ldots, 100)\}$ and the optimal $\gamma^* = 5 \cdot 50 \cdot (100 - 50) = 12,500$. It follows again that for both variants *regular MCE*, and *elite MCE* Algorithm 5.1 performs quite quickly and accurately

t	$\hat{\gamma}_t$	$S_{t,(N)}$	$p\uparrow_{min}$	$p\downarrow_{max}$
1	1296.64	1362	0.5032	0.4929
2	1297.00	1362	0.5019	0.4924
3	1367.00	1474	0.5017	0.4746
4	1491.04	1610	0.5062	0.5000
5	1664.40	1810	0.5436	0.4779
6	1847.68	2010	0.5043	0.4951
7	2013.04	2174	0.5087	0.4093
8	2183.04	2306	0.7498	0.4018
9	2336.48	2450	0.8563	0.2506
10	2439.48	2500	0.9363	0.1702
11	2500.00	2500	0.9809	0.0511
12	2500.00	2500	0.9943	0.0153
13	2500.00	2500	0.9983	0.0046
14	2500.00	2500	0.9995	0.0014
15	2500.00	2500	0.9998	0.0004

Table 7.2 : Typical evolution of the elite MCE Algorithm 5.1 for a synthetic max-cut problem with 100 nodes.

(with the maximal relative error not exceeding 2.0%). Note also that both variants of Algorithm 5.1 perform well even with the sample size $N = n = 100$, provided the size of elite sample (at each run) contains about 30 trajectories. We finally performed similar experiment with different distributions of $\boldsymbol{Z}_{11}$ and $\boldsymbol{Z}_{22}$ and different synthetic max-cut problems of the above type and found that Algorithm 5.1 performs accurately for both variants.

N	ε			T			CPU time [sec.]		
	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
100	3.71E-3	3.59E-3	3.81E-3	13.7	6	25	1.6	1	3
200	3.70E-3	3.63E-3	3.79E-3	9.9	6	14	1.9	1	3
400	3.71E-3	3.60E-3	3.81E-3	10.8	7	21	3.7	2	7
600	3.66E-3	3.58E-3	3.74E-3	9.4	6	14	4.7	3	7
800	3.64E-3	3.51E-3	3.72E-3	11.9	6	17	8.2	4	11
1000	3.64E-3	3.57E-3	3.75E-3	10.1	6	18	9.3	5	16

Table 7.3 : The performance of the regular MCE Algorithm 5.1 for the synthetic max-cut problem with 100 nodes.

N	ε			T			CPU time [sec.]		
	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
100	3.76E-3	3.57E-3	3.91E-3	10.7	6	25	1.3	1	3
200	3.38E-3	5.32E-4	3.83E-3	15.2	8	46	3.0	2	9
400	2.23E-3	0	3.75E-3	19.9	7	42	7.4	3	16
600	1.49E-3	0	3.72E-3	22.7	6	33	13.0	3	19
800	2.19E-3	0	3.70E-3	17.7	6	33	13.6	5	25
1000	1.83E-3	0	3.70E-3	16.6	6	27	15.9	6	26

Table 7.4 : The performance of the elite MCE Algorithm 5.1 for the synthetic max-cut problem with 100 nodes.

7.2 The Travelling Salesman Problem

The travelling salesman problem (TSP) can be formulated as follows. Consider a weighted graph G with n nodes, labelled $1, 2, \ldots, n$. The nodes represent cities, and the edges represent the roads between the cities. Each edge from i to j has weight or cost c_{ij}, representing the length of the road. The problem is to find the shortest *tour* that visits all the cities exactly once (except the starting city, which is also the terminating city).

Mathematically TSP reads as follows.

$$\min_{\boldsymbol{x}\in\mathcal{X}} S(\boldsymbol{x}) \quad = \quad \min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \sum_{i=1}^{n-1} c_{x_i,x_{i+1}} + c_{x_n,1} \right\} . \tag{94}$$

Note that the number of elements in $\mathcal{X}$ is

$$|\mathcal{X}| = (n-1)! \tag{95}$$

To demonstrate good performance of the MCE Algorithm 5.1 we provide a number of numerical examples. The first example concerns the benchmark TSP `ft53` taken from the URL
http://www.iwr.uni-heidelberg.de/groups/comopt/ software/TSPLIB95/atsp/

Tables 7.5 and 7.6 demonstrate typical evaluations of the regular MCE Algorithm 5.1 and its elite variant respectively, for the problem `ftv35`, which defines an asymmetric fully connected graph of size 36 with a given cost matrix c_{ij}. In Tables 7.5 and 7.6 $S_{t,(1)}$ denotes the length of smallest tour in iteration t and P_t^{mm} denotes the minimum of the maximum elements in each row of matrix $\hat{P}_t$. We set the following parameters: stopping parameter $d = 5$, rarity parameter $\rho = 0.01$, smoothing parameter $\alpha = 0.7$ and the sample size $N = 10\,n^2 = 12960$. We found that the relative errors for the regular and the elite variants are

$$\varepsilon^{regular} = \frac{\hat{\gamma}_T^{regular} - \hat{\gamma}^*}{\hat{\gamma}^*} = 0.013 \qquad \varepsilon^{elite} = \frac{\hat{\gamma}_T^{elite} - \hat{\gamma}^*}{\hat{\gamma}^*} = 0.018, \tag{96}$$

respectively. Here the best known solution $\hat{\gamma}^* = 1473$.

t	$\widehat{\gamma}_t$	$S_{t,(1)}$	P_t^{mm}
1	4066.7	3602	0.077
2	3540.2	3178	0.093
3	3092.3	2719	0.099
4	2752.7	2443	0.137
5	2480.0	2195	0.150
6	2278.4	2029	0.149
7	2103.4	1889	0.176
8	1979.9	1787	0.206
9	1878.2	1724	0.195
10	1793.6	1674	0.218
11	1712.0	1602	0.251
12	1640.3	1531	0.233
13	1540.5	1498	0.319
14	1506.8	1492	0.426
15	1495.7	1492	0.384
16	1492.0	1492	0.792
17	1492.0	1492	0.938
18	1492.0	1492	0.981
19	1492.0	1492	0.994

Table 7.5 : A typical evolution of the regular MCE Algorithm 5.1 for the TSP instance `ftv35` with $n = 36$ nodes.

t	$\widehat{\gamma}_t$	$S_{t,(1)}$	P_t^{mm}
1	4090.6	3730	0.073
2	3466.2	3039	0.133
3	2968.1	2679	0.123
4	2611.5	2448	0.180
5	2341.6	2070	0.191
6	2096.3	1881	0.200
7	1919.0	1764	0.199
8	1766.3	1650	0.189
9	1632.1	1555	0.220
10	1555.2	1537	0.428
11	1548.2	1536	0.417
12	1526.0	1520	0.743
13	1519.9	1505	0.715
14	1504.6	1499	0.714
15	1499.0	1499	0.740
16	1499.0	1499	0.922
17	1499.0	1499	0.977
18	1499.0	1499	0.993
19	1499.0	1499	0.998

Table 7.6 : A typical evolution of the elite MCE Algorithm 5.1 for the TSP instance `ftv35` with $n = 36$ nodes.

We also applied the regular and the elite MCE Algorithm 5.1 to some other instances of the TSP. Tables 7.7 and 7.8 present its performance on a selection of case studies from from the above URL
http://www.iwr.uni-heidelberg.de/groups/comopt/ software/TSPLIB95/atsp/.

In Tables 7.7 and 7.8, n denotes the number of nodes in the graph, and the other columns are as in Tables 7.3-7.4, respectively. All numerical results were obtained while using the same parameters as for the `ftv35` instance, that is $d = 5$, $\rho = 0.01$, $N = 10n^2$ and $\alpha = 0.7$. Each variant of the MCE Algorithm 5.1 was applied to each instance 10 times. It follows from Tables 7.7 and 7.8 that while the elite MCE Algorithm 5.1 is nearly twice as fast as the regular one, its accuracy is much worse. In fact, the relative error obtained by the regular MCE Algorithm 5.1 is about two times less compared to the elite one.

Instance	n	ε			T			CPU time [min.]		
		Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
`ftv35`	36	1.16%	0.14%	1.70%	19.1	17	21	14.7	13.2	16.1
`ft53`	53	3.08%	0.14%	6.24%	27.7	25	30	103	93	111
`ftv64`	65	1.30%	0.16%	2.34%	30.5	29	32	252	240	264

Table 7.7 : Performance of the regular MCE Algorithm 5.1 for various benchmark instances of the TSP.

Instance	n	ε			T			CPU time [min.]		
		Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
`ftv35`	36	2.49%	0.41%	5.02%	17.9	15	20	8.3	6.9	9.3
`ft53`	53	4.93%	2.52%	8.78%	25.2	21	30	55.8	46.6	66.3
`ftv64`	65	2.85%	0.16%	9.08%	26.5	24	29	131	118	143

Table 7.8 : Performance of the elite MCE Algorithm 5.1 for various benchmark instances of the TSP.

7.3 Comparison of CE and MCE

As mentioned earlier, formula (3) for updating γ in the CE method differs from that (47) in the MCE method. Albeit in our experiments above with the MCE Algorithm 5.1 we used $\rho = 0.01$, the value typically used in the CE method. Since γ is a function of ρ we compared the MCE Algorithm 5.1 using the updating formula (47) with both, the same MCE using the updating formula (44) and the CE Algorithm 2.1 using (3). To make a fair comparison, we used a fixed ρ, namely $\rho = 0.01$, for CE (with (3)) and MCE (with (44)), while we allowed different values of ρ for MCE with (47). Table 7.9 summarizes our experiments for the instance `ftv35` based on 10 independent replications for each case. It readily follows that MCE with (47) and $\rho = 0.05$ is more accurate than

its alternatives with $\rho = 0.01$ and $\rho = 0.1$ and it also more accurate (although very little) than CE with (3) (for $\rho = 0.01$). It finally follows that MCE with (44) and $\rho = 0.01$ is the most accurate, among the three. Note, however, that both MCE variants are about 1.5 time slower than its counterpart CE. We obtained similar results while running the instances ft53 and ftv64 (see Tables 7.7 and 7.8). Still more experimentation should be done before deciding about the superiority of MCE versus CE and vice versa, since, as follows, there exist a trade off between the accuracy and the CPU, while applying them.

	CE with (3)		MCE with (44)		MCE with (47)					
	$\rho = 0.01$		$\rho = 0.01$		$\rho = 0.01$		$\rho = 0.05$		$\rho = 0.1$	
	ε	CPU	ε	CPU	ε	CPU	ε	CPU	ε	CPU
Avg.	1.47 %	5.8	0.92 %	8.6	2.59%	7.5	0.98 %	9.1	2.21 %	9.3
Min.	0%	6.1	0 %	8.8	0.41%	6.8	0%	8.4	0%	9.0
Max.	2.26 %	5.3	1.71 %	8.3	8.28%	8.3	1.95 %	10.3	3.76 %	10.1

Table 7.9 : Comparison of the efficiency of CE using (3) with its counterparts, MCE using (44) and (47).

For an explanation of the superiority (in the sense of accuracy) of MCE versus CE see Remark 4.1.

7.4 Rare-Events Estimation

Here we present simulation studies with the MCE method for estimating rare event probability $\ell = \mathbb{P}(S(\boldsymbol{X}) \geq \gamma)$ for static models with light-tailed distributions. Its application to queuing models with both, light and heavy-tailed distribution will be reported some where else. Note that the IS estimator $\widetilde{\ell}$ in (73) is of the form $\widetilde{\ell} = N^{-1} \sum_{i=1}^{n} Z_i$ and similarly $\widehat{\ell}$ in (77). The SCV κ^2 of each Z_i is estimated as

$$\widehat{\kappa^2} = \frac{N^{-1} \sum_{i=1}^{N} Z_i^2 - (\widehat{\ell})^2}{(\widehat{\ell})^2} \; .$$

The relative error of the estimator is thus estimated by $\mathrm{RE} = \sqrt{\widehat{\kappa^2}/N}$. Assuming asymptotic normality of the estimator, the confidence intervals (CI) now follow in a standard way. For example, a 95% CI for ℓ is given by

$$\widehat{\ell} \pm 1.96 \, \widehat{\ell} \, \mathrm{RE} \; .$$

For a quite moderate probability like $\ell = 10^{-3}$, we also compare the MCE estimates $\widetilde{\ell}$ and $\widehat{\ell}$ with their corresponding CMC estimate, denoted as $\bar{\ell}$.

We consider two simple static models: Our first model is

$$S(\boldsymbol{X}) = \max\{X_1, \ldots, X_5\},$$

where $X_i,\ i = 1, \ldots, 5$ are iid each distributed with the following six point pdf

$$h(x) = 0.1998,\ x = 1, \ldots, 5 \text{ and } h(x) = 0,0001,\ x = 6. \tag{97}$$

Tables 7.10 presents the values $\widetilde{\ell}$, $\widehat{\ell}$ and $\bar{\ell}$ for $S(\boldsymbol{X}) = \max\{X_1, \ldots, X_5\}$ and their corresponding estimates of $\kappa(\gamma)$, denoted as $\widetilde{\kappa}$, $\widehat{\kappa}$ and $\bar{\kappa}$, respectively as function of the sample size N for $\gamma = 6$. Note that in this case $\ell = 4.99001 \cdot 10^{-3}$.

It follows from the results of Table 7.10 that both $\widetilde{\ell}$ and $\widehat{\ell}$ are in agreement with their CMC one $\bar{\ell}$, and that $\widetilde{\ell}$ outperforms $\widehat{\ell}$, which is what one would expect.

N	MCE with f^*		MCE with $f^*_{(m)}$		CMC	
	$\widetilde{\ell}$	$\widetilde{\kappa}$	$\widehat{\ell}$	$\widehat{\kappa}$	$\bar{\ell}$	$\bar{\kappa}$
1,000	4.99E-03	6.7E-15	5.26E-03	3.6E-02	4.0E-03	4.9 E-01
10,000	4.99E-03	1.2E-15	4.94E-03	1.2E-02	6.3 E-03	1.2 E-01
100,000	4.99E-03	3.6E-15	5.00E-03	3.7E-03	5.3E-03	4.3 E-02

Table 7.10 : The performance of the MCE and CMC estimates for the sample functions $S(\boldsymbol{X}) = \max\{X_1, \ldots, X_5\}$.

Our second model is a *stochastic shortest path* problem. That is, we have a weighted graph given in Figure 1, with random weights $X_1, \ldots, X_5$. Let $S(\boldsymbol{X})$

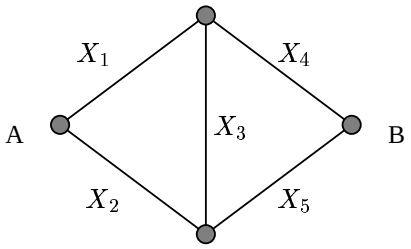


Figure 1: Shortest path from A to B.

be the length of the shortest path from node A to node B. Note that there are four possible paths. We wish to estimate from simulation the probability $\ell = \mathbb{P}(S(\boldsymbol{X}) \geq \gamma)$ that the length of the shortest path $S(\boldsymbol{X})$ exceeds some fixed γ. For this model we assume that the random variables $X_1, \ldots, X_5$ are iid, each distributed

$$h(x) = 0.50,\ 0.35,\ 0.12,\ 0,001,\ 0,001,\ 0,001,\ x = 1, \ldots, 6. \tag{98}$$

Table 7.11 presents the performance of both MCE estimates, for the shortest path model with $\gamma = 11$ and $\ell = 9.0 \cdot 10^{-8}$. Note that in this case the CMC estimates $\bar{\ell}$ are useless, since ℓ is very small.

N	MCE with $f^*_{(m)}$		MCE with f^*	
	$\widetilde{\ell}$	$\widetilde{\kappa}$	$\widehat{\ell}$	$\widehat{\kappa}$
1,000	9.91E-08	6.51E-2	9.13E-08	4.62E-2
10,000	9.02E-08	2.14E-2	8.95E-08	2.16E-2
100,000	9.09E-08	6.71E-3	9.01E-08	4.63E-3

Table 7.11 : The performance of the MCE estimates for the bridge network.

We also compared the accuracy of the MCE estimate $\hat{\ell}$ based on the marginal pdf's with its standard CE counterpart. We set for CE, the rarity parameter $\rho = 0.01$ and took a sample size $N = 10^4$ at each iteration. We found that $\widehat{\ell}$ typically has the same accuracy as its CE counterpart, while $\widetilde{\ell}$ outperforms CE.

8 Conclusions and Further Research

We presented a new cross-entropy method, called the *minimum cross-entropy* (MCE) method for solving NP-hard combinatorial optimization problems, which is based on the stochastic version of the classic Kullback's MinxEnt method and which involves probability of rare-event estimation. We discussed the similarities and the differences between the standard CE and MCE, its convergence and efficiency. Below we list some topics for further research.

1. **Convergence proof** To prove the convergence, speed of convergence of the MCE Algorithm 5.1 and its modifications.

2. **Rare-event Estimation** Investigate the complexity properties of MCE and Csiszer's ϕ-divergence family for probability of rare-event estimation in static and queueing models with both light-tailed and heavy-tailed distributions.

3. **Constrained Optimization with MCE** The MCE framework allows to tackle constrained combinatorial and multi-extremal problems, like the well known knapsack problem, with a volume constraint on the knapsack.

4. **Design an "optimal" system satisfying some constraints** Consider the following well known queueing decision problem: design an "optimal" queueing model, say a $GI/G/1$ queue, such that the variance of the number of customers in the queue is minimal subject to that the expected number of customers in the queue does not exceed a fixed number γ. When we say "design an optimal system" we mean select the underlying distributions, like the inter-arrival and the service time distributions.

   Rather than solving this typically difficult variance minimization problem, one can solve a simpler problem, namely a single iteration of the stochastic version of the Jaynes' MaxEnt problem (P_0) such as the MCE program (62) with $\boldsymbol{p}_{t-1} \equiv const$ and fixed $\widehat{\gamma}_t$, that is $\widehat{\gamma}_t = \gamma$. By doing so we shall

maximize Jaynes' cross-entropy subject to some constraints rather than we minimize the variance, subject to the some constraints. In short, the optimality in MCE is in the minimal cross-entropy sense, rather than in the minimal variance sense. Notice that in this case, no rare-events are involved and we have a single-stage MinxEnt program.

9 Appendix

9.1 A New Method for Trajectory Generation for SEN

Here we present a new method for trajectory generation for SEN, where similar to SNN we associate a probability distribution with each *node* rather than with each *edge*. We demonstrate our method for a TSP problem, assuming without loss of generality that the network is fully connected.

To proceed let us associate with each node $i,\ i = 1,\dots,n$ of the network a normal pdf $\mathcal{N}(\mu_i, \sigma_i^2),\ i = 1,\dots,n$. It is crucial to start the algorithm for trajectory generation using identical pdf's (identical initial conditions). We assume without loss of generality that at the first iteration all $\mu_i = 0,\ i = 1,\dots,n$ and all $\sigma_i^2 = 1,\ i = 1,\dots,n$ and then proceed as follows:

Algorithm 9.1 (A New Algorithm for Trajectory Generation for SEN)

1. *Take a sample*
$$\boldsymbol{X} = (X_1,\dots,X_n) \sim \{\mathcal{N}(\mu_1,\sigma_1^2),\dots,\mathcal{N}(\mu_n,\sigma_n^2)\}.$$
2. *Order the sample $X_1,\dots,X_n$ in the increasing order and denote the ordered statistic values as*
$$X_{(1)},\dots,X_{(n)}.$$
3. *Start from $X_{(1)}$ and mark the true location of each $X_{(k)},\ k = 1,\dots,n$ in the original sequence $X_1,\dots,X_n$ as $i_k,\ k = 1,\dots,n$.*
4. *Take the ordered sequence $i_1 \cdots i_n$ as the desired tour.*
5. *Calculate the tour length S based on the generated trajectory $i_1 \cdots i_n$.*

Example 9.1 For clarity, consider a TSP problem with four cities. Assume that $(X_1, X_2, X_3, X_4) = (-2, -4, 17, 1)$. We have $(X_{(1)}, X_{(2)}, X_{(3)}, X_{(4)} = (-4, -2, 1, 17)$ and the corresponding trajectory is: $i_1 i_2 i_3 i_4 = 2 \to 1 \to 4 \to 3 \to 2$.

Generating enough tours at each iteration, say cn tours (instead of cn^2 tours as required earlier) we can employ any of the two optimization Algorithms (either CE Algorithm 2.1, or the MCE Algorithm 5.1), and then update the parameters $\mu_i,\ i = 1,\dots,n$ and $\sigma_i^2,\ i = 1,\dots,n$ accordingly. We argue that

using Algorithm 9.1 for trajectory generation, each normal pdf will converge to a degenerated case and the associated trajectory $i_1 \cdots i_n$ will coincide with high probability with the unique optimal trajectory $i_1^* \cdots i_n^*$. Note that using such *SNN like generation* Algorithm 9.1, the optimization algorithms (either the CE Algorithm 2.1, or the MCE Algorithm 5.1) will be able to solve SEN-like problems of the same size as of SNN-like ones, namely of order of thousands. Our preliminary studies show that using Algorithm 9.1 in either the CE Algorithm 2.1, or the MCE Algorithm 5.1, one indeed obtains a substantial speedup, but the accuracy of the solutions is somewhat worse by 2-3% as compared to the standard SEN trajectory generation algorithms [10]. More studies are underway.

9.2 Proof of (26) and (27)

To derive the optimal solution $\boldsymbol{p}^*$ of the program (P_1), called the *primal* program, it is typically easier to solve the associated dual program [6]. Below we present the corresponding algorithm.

Algorithm 9.2 Dual Algorithm

1. Write the Lagrangian function of the *primal*, that is

$$L(\boldsymbol{p}, \boldsymbol{\lambda}, \beta) = \sum_{i=1}^{m} p_i \ln \frac{p_i}{u_i} + \sum_{j=1}^{k} \lambda_j \Big(\sum_{i=1}^{m} S_j(x_i) p_i - \gamma_j \Big) + \beta \Big(\sum_{i=1}^{m} p_i - 1 \Big). \quad (99)$$

2. Solve (for fixed $\boldsymbol{\lambda}, \beta$)

$$\min_{\boldsymbol{p}} L(\boldsymbol{p}, \boldsymbol{\lambda}, \beta) \quad (100)$$

by solving

$$\nabla_{\boldsymbol{p}} L(\boldsymbol{p}, \boldsymbol{\lambda}, \beta) = \mathbf{0},$$

that is, componentwise

$$\nabla_{p_i} L(\boldsymbol{p}, \boldsymbol{\lambda}, \beta) = \ln p_i + 1 - \ln u_i + \sum_{j=1}^{k} \lambda_j S_j(x_i) + \beta = 0.$$

Denote the optimal solution and the optimal function value obtained from the program (100) as $\boldsymbol{p}(\boldsymbol{\lambda}, \beta)$ and $\mathcal{S}(\boldsymbol{\lambda}, \beta)$, respectively. We have

$$p_i(\lambda, \beta) = u_i \exp\{-1 - \beta - \sum_{j=1}^{k} \lambda_j S_j(x_i)\}, \; i = 1, \ldots, m, \quad (101)$$

and

$$\mathcal{S}(\boldsymbol{\lambda}, \beta) = -\sum_{j=1}^{k} \lambda_j \gamma_j - \beta - \sum_{i=1}^{m} u_i \exp\{-1 - \beta - \sum_{j=1}^{k} \lambda_j S_j(x_i)\}.$$

For completeness, compare (P_1) and (100) with the unconstrained program (4) for CE.

3. Solve the *dual* program

$$\max_{\boldsymbol{\lambda},\beta} \mathcal{S}(\boldsymbol{\lambda},\beta). \tag{102}$$

Note that we can write (102) as

$$\max_{\boldsymbol{\lambda}}\{\max_{\beta} \mathcal{S}(\boldsymbol{\lambda},\beta)\} \tag{103}$$

and, thus solve (103) in two optimization stages. More specifically, denote by $\beta^*(\boldsymbol{\lambda})$ the value of β obtained from the first optimization stage, and substitute it back into (103). Denote the resulting function by $D(\boldsymbol{\lambda})$. We obtain

$$\beta^*(\boldsymbol{\lambda}) = \ln\left\{\sum_{i=1}^{m} u_i \exp\{-1-\sum_{j=1}^{k}\lambda_j S_j(x_i)\}\right\}$$

$$D(\boldsymbol{\lambda}) = \ln\left\{\sum_{i=1}^{m} u_i \exp\{-\sum_{j=1}^{k}\lambda_j S_j(x_i)\}\right\} + \sum_{j=1}^{k}\lambda_j\gamma_j.$$

In the second optimization stage solve

$$\max_{\boldsymbol{\lambda}} D(\boldsymbol{\lambda}). \tag{104}$$

Denote by $\boldsymbol{\lambda}^*$ the value of $\boldsymbol{\lambda}$ which solves (104).

Since $D(\boldsymbol{\lambda})$ is continuously differentiable and concave with respect to $\boldsymbol{\lambda}$, we can derive $\boldsymbol{\lambda}^*$ by solving

$$\nabla_{\boldsymbol{\lambda}} D(\boldsymbol{\lambda}) = \mathbf{0}, \tag{105}$$

which can be written componentwise in the following explicit form:

$$\begin{aligned}\nabla_{\lambda_j} D(\boldsymbol{\lambda}) &= -\frac{\sum_{i=1}^{m} S_j(x_i)u_i \exp\{-\sum_{r=1}^{k} S_r(x_i)\lambda_r\}}{\sum_{i=1}^{m} u_i \exp\{-\sum_{r=1}^{k} S_r(x_i)\lambda_r\}} + \gamma_j \quad = \\ &= -\frac{\mathbf{E}_{\boldsymbol{u}} S_j(X)\exp\{-\sum_{r=1}^{k} S_r(X)\lambda_r\}}{\mathbf{E}_{\boldsymbol{u}}\exp\{-\sum_{r=1}^{k} S_r(X)\lambda_r\}} + \gamma_j \qquad = 0.\end{aligned} \tag{106}$$

The optimal vector $\boldsymbol{\lambda}^* = (\lambda_1^*,\ldots,\lambda_k^*)$ can be found by solving (27) numerically. Note that if (P_1) has a nonempty interior optimal solution, then the dual program (104) has an optimal solution $\boldsymbol{\lambda}^*$.

4. Substitute $\boldsymbol{\lambda}^*$ and $\beta^*(\boldsymbol{\lambda})$ into (101) to obtain the solution of (P_1):

$$\begin{aligned}p_i^* &= \frac{u_i \exp\{-\sum_{j=1}^{k}\lambda_j^* S_j(x_i)\}}{\sum_{r=1}^{m} u_r \exp\{-\sum_{j=1}^{k}\lambda_j^* S_j(x_r)\}} = \\ &= \frac{\mathbf{E}_{\boldsymbol{u}} I_{\{X=x_i\}}\exp\{-\sum_{j=1}^{k}\lambda_j^* S_j(X)\}}{\mathbf{E}_{\boldsymbol{u}}\exp\{-\sum_{j=1}^{k}\lambda_j^* S_j(X)\}}, \quad i=1,\ldots,m.\end{aligned} \tag{107}$$

5. Deliver $\boldsymbol{p}^*$ as the optimal solution of the primal program (P_1).

Thus the proof of (26) and (27) has been established. □

9.3 Convergence of the Deterministic MCE

Here we present the convergence proof of the sequence of the solutions of programs (39) to the optimal solution $(\gamma^*, \boldsymbol{p}^*)$ for the single-dimensional case when γ_t is defined by (43). The extension of the above to multi-dimensional programs (45) is similar.

To proceed, let us denote by m the total number of possible realizations of x, an let us enumerate these realizations in a way which makes the performance $S(\cdot)$ a nondecreasing function of the index i, $i = 1, ..., n$, of a realization. The value of performance at realization $\# i$ will be denoted S_i, so that $S_1 \leq S_2 \leq ... \leq S_m$. For the sake of simplicity, assume that the maximizer of S is unique, that is, $S_{m-1} < S_m$. Further, let $\Delta_m = \{\boldsymbol{p} = (p_1, ..., p_m) : p \geq 0, \sum\limits_i p_i = 1\}$ be the set of all probability distributions on the set of all possible realizations of x (which now becomes the set $\{1, ..., m\}$), $\boldsymbol{u} = m^{-1}(1, ..., 1)$ be the uniform distribution from Δ_m. For $\boldsymbol{p} \in \Delta_m$, let

$$\mathbb{E}(\boldsymbol{p}) = \sum_{i=1}^{m} S_i p_i;$$

Consider the transformation

$$\boldsymbol{p} \mapsto \boldsymbol{p}^+ = \mathbb{P}_\gamma(\boldsymbol{p}) : \boldsymbol{p}_i^+ = \frac{p_i \exp\{-S_i\lambda^*\}}{\sum\limits_{j=1}^{m} p_j \exp\{-S_j\lambda^*\}},$$

where λ^* is the root of the equation

$$\frac{\sum\limits_i S_i p_i \exp\{-S_i\lambda\}}{\sum\limits_i p_i \exp\{-S_i\lambda\}} = \gamma; \tag{108}$$

note that $\mathbb{E}(\boldsymbol{p}^+) = \gamma$, provided that $\boldsymbol{p}^+$ is well-defined. Finally, let Δ_m^+ be the set of monotone distributions form Δ_m:

$$\Delta_m^+ = \{\boldsymbol{p} \in \Delta_m : p_1 \leq p_2 \leq ... \leq p_m\},$$

and let $i(\boldsymbol{p})$ be the first index i such that $p_i > 0$. Since $S_m > S_{m-1}$, for $\boldsymbol{p} \in \Delta_m^+ \backslash \{(0, 0, ..., 0, 1)\}$ one has $\mathbb{E}(\boldsymbol{p}) < S_m$.

Proposition 9.1 (i) *For $\boldsymbol{p} \in \Delta_+$ and $S_m > \gamma > \mathbb{E}(\boldsymbol{p})$ the distribution $\boldsymbol{p}^+ = \mathbb{P}_\gamma(\boldsymbol{p})$ is well-defined and belongs to $\Delta_m^+ \backslash \{(0, ..., 0, 1)\}$.*

(ii) *Let $\boldsymbol{p}_0 = \boldsymbol{u}$, and let $\boldsymbol{p}_{t+1} = \mathbb{P}_{\gamma_t}(\boldsymbol{p}_t)$, where the policy of choosing γ_t ensures that for $p \in \Delta_m^+$ one has $\gamma_t - \mathbb{E}(\boldsymbol{p}_t) \geq \rho_t(S_m - \mathbb{E}(\boldsymbol{p}_t))$ with $1 > \rho_t > 0$ satisfying $\prod\limits_{t=1}^{\infty}(1 - \rho_t) = 0$. Then the sequence $\{\boldsymbol{p}_t\}$ converges to the degenerate distribution $\boldsymbol{p}^* = (0, ..., 0, 1) \in \Delta_m$.*

Proof. (i) Let $\boldsymbol{p} \in \Delta_m^+$. Clearly the function

$$f(\lambda) = \frac{\sum\limits_i S_i p_i \exp\{-S_i\lambda\}}{\sum\limits_i p_i \exp\{-S_i\lambda\}}$$

is strictly decreasing. Indeed,

$$-f'(\lambda) = \frac{\left(\sum\limits_{i\geq i(\boldsymbol{p})} S_i^2 p_i \exp\{-S_i\lambda\}\right)\left(\sum\limits_{i\geq i(\boldsymbol{p})} p_i \exp\{-S_i\lambda\}\right) - \left(\sum\limits_{i\geq i(\boldsymbol{p})} S_i p_i \exp\{-S_i\lambda\}\right)^2}{\left(\sum\limits_{i\geq i(\boldsymbol{p})} p_i \exp\{-S_i\lambda\}\right)^2}.$$

Setting next

$$v_i = \frac{p_i \exp\{-S_i\lambda\}}{\sum\limits_j p_j \exp\{-S_j\lambda\}}$$

we see that $v \in \Delta_m$ and that $-f'(\lambda)$ is the difference between the second moment of the random variable taking value S_i with probability v_i, $i = 1, ..., m$, and the squared first moment of the same variable. Such a difference is always nonnegative and it is zero if and only if the above random variable is constant almost surely, which of course is not the case since $\boldsymbol{p} \in \Delta_+$ and therefore $\mathbb{E}(\boldsymbol{p}) < S_m$.

The strictly decreasing continuous function $f(\lambda)$ converges to $S_{i(\boldsymbol{p})}$ as $\lambda \to \infty$ and converges to S_m as $\lambda \to -\infty$; thus, when $S_{i(\boldsymbol{p})} < \gamma < S_m$, equation (108) has a single root, and this root is negative, provided that $\gamma > f(0) = \mathbb{E}(\boldsymbol{p})$. Thus, when $S_m > \gamma > \mathbb{E}(\boldsymbol{p})$, λ^* is well-defined and negative. Consequently, $\boldsymbol{p}^+$ is well-defined and is monotone along with $\boldsymbol{p}$ (recall that S_i is nondecreasing in i). Finally, $\mathbb{E}(\boldsymbol{p}^+) = \gamma < S_m$, so that $\boldsymbol{p}^+ \in \Delta_m^+ \backslash \{(0, ..., 0, 1)\}$. We have proved (i).

(ii) Note first that by (i) the sequence of distributions $\boldsymbol{p}_t$ is well-defined and belongs to Δ_m^+. Besides this, by construction we have $\mathbb{E}(\boldsymbol{p}_{t+1}) = \gamma_t \geq \mathbb{E}(\boldsymbol{p}_t)$. It follows that the sequence $\gamma_t = \mathbb{E}(\boldsymbol{p}_{t+1})$ is nondecreasing. We claim that this sequence converges to S_m (whence, of course, $\boldsymbol{p}_t \to (0, ..., 0, 1)$ as $t \to \infty$ due to $\mathbb{E}(\boldsymbol{p}_t) = \gamma_{t-1}$ and $S_m > \max\limits_{i\leq m-1} S_i$). Indeed, since $\gamma_t - \mathbb{E}(\boldsymbol{p}_t) \geq \rho_t(S_m - \mathbb{E}(\boldsymbol{p}_t))$, we have

$$S_m - \gamma_t \leq S_m - [\mathbb{E}(\boldsymbol{p}_t) + \rho_t(S_m - \mathbb{E}(\boldsymbol{p}_t))] = (1-\rho_t)(S_m - \mathbb{E}(\boldsymbol{p}_t)) = (1-\rho_t)(S_m - \gamma_{t-1}),$$

whence

$$S_m - \gamma_t \leq (S_m - \gamma_0) \prod_{\tau=1}^{t} (1 - \rho_\tau) \to 0,\ t \to \infty.$$

□

Acknowledgments

I would like to thank Prof. Arkadi Nemirovski from the Technion for his proof of Proposition 9.1 and Yohai Gat from the Technion for performing the computational part for this paper and some insightful remarks. Finally, while the paper was completed Dr. David H. Wolpert from NASA Ames Research Center, brought my attention to their work on *Probability Collectives* (PC), which has some common ground with the MCE method. The interested reader is refereed to [11] and references therein.

References

[1] Aarts E. H. L. and J. H. M. Korst, *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons, 1989.

[2] Cover T.M. and Thomas J.A., *Elements of Information Theory*, John Wiley & Sons, inc, 1991.

[3] de Boer P.T., Kroese D.P., Mannor S. and Rubinstein R.Y., *A Tutorial on the Cross-Entropy Method*, Annals of Operations Research, 2005, (to appear).

[4] Goldberg D., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1989.

[5] Liu J.S. *Monte Carlo Strategies in Scientific Computing*. Springer, 2001.

[6] Kapur J.N. and H.K. Kesavan, *Entropy Optimization with Applications*, Academic Press, Inc., 1992.

[7] Rubinstein, R.Y., The cross-entropy method for combinatorial and continuous optimization, *Methodology and Computing in Applied Probability* **2**, 127–190, 1999.

[8] Rubinstein R.Y. , *Cross-entropy and rare-event for maximal cut and bipartition problems*, ACM Transactions on Modelling and Computer Simulation **12** (1), 27–53, 2002.

[9] Rubinstein, R.Y., Melamed B., *Modern Simulation and Modeling*, John Wiley & Sons, Inc., 1998.

[10] Rubinstein R.Y. and Kroese D.P., *The Cross-Entropy Method: a Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*, Springer, 2004.

[11] Wolpert H. D. *Information Theory - The Bridge Connecting Bounded Rational Game Theory and Statistical Physics*. Manuscript, NASA Ames Research Center.