How Many Needles Are in a Haystack, or How to Solve #P-Complete Counting Problems Fast

Reuven Y. Rubinstein

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

Abstract

We present two randomized entropy-based algorithms for approximating quite general #P-complete counting problems, like the number of Hamiltonian cycles in a graph, the permanent, the number of self-avoiding walks and the satisfiability problem. In our algorithms we first cast the underlying counting problem into an associate rare-event probability estimation, and then apply dynamic importance sampling (IS) to estimate efficiently the desired counting quantity. We construct the IS distribution by using two different approaches: one based on the cross-entropy (CE) method and the other one on the stochastic version of the well known minimum entropy (MinxEnt) method. We also establish convergence of our algorithms and confidence intervals for some special settings and present supportive numerical results, which strongly suggest that both ones (CE and MinxEnt) have polynomial running time in the size of the problem.

Keywords. Cross-Entropy, Rare-Event Probability Estimation, Hamilton cycles, Self-Avoiding Walks, #P-Complete Problems, Stochastic and Simulation-Based Optimization.

1 Introduction

Many important problems in computer science, engineering, and graph theory are of the following general form: given a certain "universe" of objects $\mathcal{X}$, (for example, paths in a graph, or coloring of a graph) how many objects of a particular type are present? In other words, if $\mathcal{X}^-$ is the subset of these "special" objects, what is its cardinality $|\mathcal{X}^-|$?

In most cases, these counting problems belong to the class of so-called $\#P$-complete problems [10], [16], which is related to the familiar class of NP-hard problems. To date very little is known about how to construct efficient algorithms — or indeed any algorithms — for various $\#P$-complete problems. Here are some examples of #P-complete counting problems:

- The *Hamiltonian cycle* (HC) problem. How many Hamiltonian cycles a graph has? That is, how many tours contains a graph in which every vertex is visited exactly once (except for the beginning/end vertex)? Note that the problem of finding a Hamiltonian cycle is a particular case of finding a longest tour in a TSP, in which each pair of vertices with an edge between them has distance 1, while non-edge vertex pairs are separated by distance 0.

- The *permanent* problem. Calculate the permanent of a matrix. This problem is closely related to counting the number of distinct representations of a set.

- The *self-avoiding walk* (SAW) problem. How many self-avoiding random walks of length n exist without looping, provided we start at 0 and we are allowed to move at each grid point to any of the neighboring direction with equal probability? For example, let c_n denote the number of self-avoiding random walks with exactly n edges, on the lattice $\mathbb{Z}^2$, starting at (0,0). In this case, $c_1 = 4,\ c_2 = 12, \ldots$, and in general it is known [16] that

  $$c_n = \mathrm{e}^{\kappa n + O(n^{1/2})},$$

  where κ is a constant whose exact value is unknown.

- The *connectivity* problem. Given two different nodes in a directed or undirected graph, say nodes A and B, how many paths exist from A to B that do not traverse the same edge more than once?

- The *satisfiability* problem. Let $\mathcal{X}$ be a collection of all sets of n Boolean variables $\{x_1, \ldots, x_n\}$. Thus, $\mathcal{X}$ has cardinality $|\mathcal{X}| = 2^n$. Let $\mathcal{C}$ be a set of Boolean clauses. Examples of such clauses are $x_1 \vee \bar{x}_2$ and $(x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2)$. How many (if any) satisfying truth assignments for $\mathcal{C}$ exist, that is, how many ways are there to set the variables $x_1, \ldots, x_n$ either true or false so that each clause in $\mathcal{C}$ is true?

- The *k-coloring* problem ($k \geq 3$). Given k distinct colors, in how many different ways one can color the nodes (edges) of a graph, so that each two adjacent nodes (edges) in the graph have different colors?

- The *spanning trees* problem. How many *unlabeled* spanning trees has a graph G? Note that this counting problem is easy for *labeled* graphs.

- The *isomorphism problem*. How many isomorphisms exist between two given graphs G and H? In other words, in an isomorphism problem one needs to find all mappings ϕ between the vertices of G and H such that (x, y) is an edge of G if and only if $(\phi(x), \phi(y))$ is an edge of H.

- The *clique* problem. How many cliques of fixed size k exist in a graph $G = (V, E)$?

Some other #P-complete problems include counting the number of perfect matches in a bipartite graph and counting the number of forests in a graph.

It is interesting to note [10], [16] that in many cases the counting problem is hard to solve, while the associated optimization problem is easy; in other words *decision is easy, counting is hard*. As an example, finding the shortest path between two fixed nodes in a network is easy, while finding the total number of paths between the two nodes is difficult.

This paper presents a generic approach to #P-complete counting problems, where the underlying problem is first cast into an estimation problem, and then solved via dynamic *importance sampling* (IS). The importance sampling distribution is derived by using two different methods: the CE [15] and the MinxEnt [13]. Note that in the former we update only the paramers of the IS distribution belonging to a parametric family, while in the later - the entire distribution.

At this end we would like to note that although these #P-complete problems are of both theoretical and practical importance and have been well defined for at least a quarter of a century, we are not aware of generic deterministic or randomized method for *fast* approximating their solution. We are even not aware of any bench mark problem to compare our method with others. Because of that we believe that our method is kind of a *breakthrough* in the field and this work will motivate more research and applications on difficult counting problems similar to what the original CE method [2], [12],[15] has done in the fields of Monte Carlo simulation and simulation-based optimization in recent years.

The rest of the paper is organized as follows. Section 2 deals with foundations of Monte Carlo estimation for counting. In Section 3, which is our main one, we present the main counting algorithm for estimating the number of Hamiltonian cycles in a graph using CE. Here we also introduce two alternatives for generation Hamiltonian cycles (trajectories): the first one is based on, the so-called, *auxiliary* approach, while the second one - on, the so-called, *original* approach. In Section 4 we show how to apply the CE algorithm to the permanent problem. Note that in [6] a polynomial randomized algorithm for calculating the permanent was obtained. Their approach is based on the MCMC method, and the complexity of their algorithm is $O(n^{10})$, where n is the size of the problem. Because of the high power of the polynomial, application of such MCMC method is quite limited. For an earlier work on counting HC's in a dense graph using randomization see [3]. The asymptotic convergence of CE for #P-complete counting problems follows from [15], where a general convergence results of CE for rare-event based problems is established. Our empirical results clearly indicate that the running time of CE for #P-complete counting problems is $O(n^2) - O(n^3)$. In Sections 5, 6 and 7 we deal with the SAW, the connectivity and the satisfiability problems, respectively. Section 8 presents supportive simulation results with the CE method for HC's, permanent and SAW. In particular we show (yet empirically) that if the incidence matrix associated with the graph is not too sparse, then using the original approach

1. The proposed CE algorithm converges after a single iteration This in turn implies that one can immediately establish a central limit theorem and obtain valid confidence intervals for the desired quantities by using, say Theorem 5.4.1 of [14] and the classic *maximum likelihood* (ML) results.

2. One can design a *non iterative* IS pdf which performs almost as well as the one based on the single iterative CE version. We explain the reason for such nice phenomenon.

Section 9 deals with application of MinxEnt to counting problems, while in Section 10 we present simulation results with the MinxEnt method for the HC's, the permanent and the satisfiability problems, which do not indicate significant improvement of MinxEnt estimators as compared to their CE counterparts. Finally in Sections 12 and 13 we give some background on CE and the stochastic version of the the classic MinxEnt method in [13].

2 Counting via Monte Carlo

We start with the fundamentals of the Monte-Carlo method for estimation and counting by considering the following basic example.

Example 2.1 Assume we want to calculate an area of same "irregular" region $\mathcal{X}^-$. The Monte-Carlo method suggests inserting the "irregular" region $\mathcal{X}^-$ into a nice "regular" one $\mathcal{X}$, say a rectangular (see Figure 1) and then apply the following sampling procedure:

1. Generate a random sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$, *uniformly* distributed over the "regular" region $\mathcal{X}$.
2. Estimate the desired area $|\mathcal{X}^-|$ as

$$\widehat{|\mathcal{X}^-|} = |\mathcal{X}| \frac{1}{N} \sum_{k=1}^{N} I_{\{\boldsymbol{X}_k \in \mathcal{X}^-\}} \,, \tag{1}$$

where $I_{\{\boldsymbol{X}_k \in \mathcal{X}^-\}}$ denotes the indicator of the event $\{\boldsymbol{X}_k \in \mathcal{X}^-\}$. Note that according to (1) we accept the generated point $\boldsymbol{X}_k$, if $\boldsymbol{X} \in \mathcal{X}^-$ and reject it otherwise.

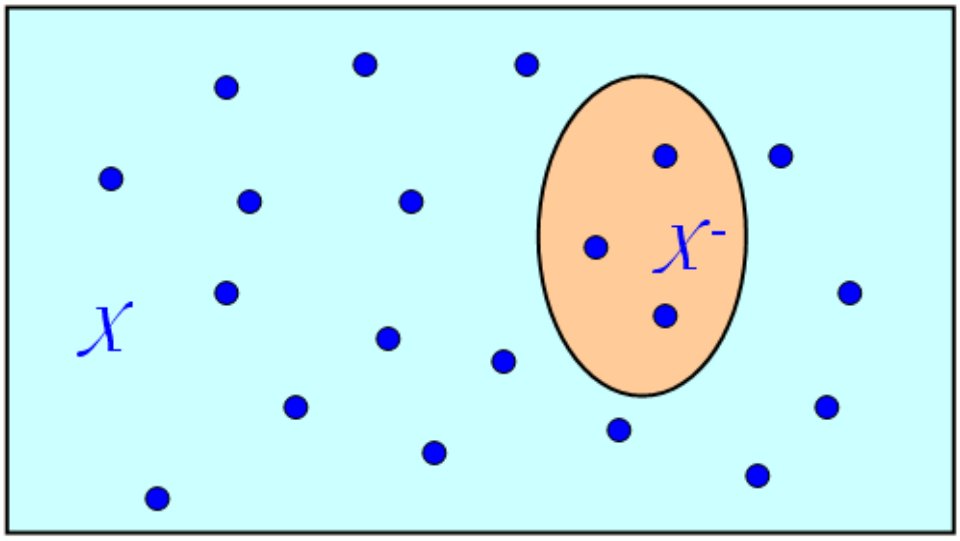

Figure 1: Illustration of the Acceptance-rejection method

Formula (1) is also valid for *counting problems*, that is where $\mathcal{X}^-$ presents a discrete rather a continuous set of points. In this case one generates a uniform sample over the grid points of same larger "nice" region $\mathcal{X}$ and then, as before, uses the acceptance-rejection method to estimate $|\mathcal{X}^-|$.

It is well known that in order for $\widehat{|\mathcal{X}^-|}$ to be an accurate estimate of $\mathcal{X}^-$ the sample size N in (1) should be typically large, especially if the quantity

$$\ell = \frac{|\mathcal{X}^-|}{|\mathcal{X}|} \tag{2}$$

is small. As we shall see below the latter case often occurs for the #P-Complete counting problems.

At this end note that ℓ can be also written as

$$\ell = \mathbb{E}_{\mathsf{U}}[I_{\{\boldsymbol{X}\in\mathcal{X}^-\}}], \tag{3}$$

where the subscript U indicates that $\boldsymbol{X}$ is *uniformly* distributed over $\mathcal{X}$. In short, ℓ is the probability of accepting the trial point $\boldsymbol{X}$ as a point in $\mathcal{X}^-$, while, in fact, it is generated uniformly over $\mathcal{X}$.

The naive or the *crude Monte-Carlo* (CMC) estimate of ℓ (see (1)) is

$$\widehat{\ell} = \frac{1}{N}\sum_{k=1}^{N} I_{\{\boldsymbol{X}_k\in\mathcal{X}^-\}}, \tag{4}$$

where $\{\boldsymbol{X}_k\}$ is a random sample from the uniform density $f(\boldsymbol{x})$ over $\mathcal{X}$, that is from

$$f(\boldsymbol{x}) = \frac{1}{|\mathcal{X}|}.$$

Since ℓ is a small number, (a rare-event probability) the naive CMC estimate (4) is useless. However, we can use the well known *importance sampling* (IS) method to speed up the simulation process. IS is based on the idea of replacing the original probability distribution (in our case replacing the uniform one $f(\boldsymbol{x})$) by an alternative one, say $g(\boldsymbol{x})$, such that by sampling from g the event $I_{\{\boldsymbol{X}\in\mathcal{X}^-\}}$ occurs more frequently. Using IS we can write ℓ as

$$\ell = \sum_{\boldsymbol{x}\in\mathcal{X}} I_{\{\boldsymbol{X}\in\mathcal{X}^-\}} f(\boldsymbol{x})\frac{g(\boldsymbol{x})}{g(\boldsymbol{x})} = \mathbb{E}_g[I_{\{\boldsymbol{X}\in\mathcal{X}^-\}}\frac{f(\boldsymbol{X})}{g(\boldsymbol{X})}], \tag{5}$$

where $\mathbb{E}_g$ means that the expectations is taken with respect to the IS pdf $g(\boldsymbol{x})$.

Combining $|\mathcal{X}^-| = \ell|\mathcal{X}|$ with (3) and with (5) we can present $|\mathcal{X}^-|$ in the following two alternative ways

$$|\mathcal{X}^-| = |\mathcal{X}|\mathbb{E}_{\mathsf{U}}[I_{\{\boldsymbol{X}\in\mathcal{X}^-\}}]$$

and

$$|\mathcal{X}^-| = \mathbb{E}_g[I_{\{\boldsymbol{X}\in\mathcal{X}^-\}}\frac{1}{g(\boldsymbol{X})}], \tag{6}$$

respectively.

We shall use below the representation (6), for which the IS estimate of $\mathcal{X}^-$ can be written as

$$\widehat{|\mathcal{X}^-|} = \frac{1}{N}\sum_{k=1}^{N} I_{\{\boldsymbol{X}_k\in\mathcal{X}^-\}}\frac{1}{g(\boldsymbol{X}_k)} = \sum_{\boldsymbol{X}_k\in\mathcal{X}} I_{\{\boldsymbol{X}_k\in\mathcal{X}^-\}}\frac{1}{g(\boldsymbol{X}_k)} = \sum_{\boldsymbol{X}_k\in\mathcal{X}^-}\frac{1}{g(\boldsymbol{X}_k)}\,, \tag{7}$$

where the random sample $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_N$ is taken from g. That is, to estimate $|\mathcal{X}^-|$ we sample from the IS pdf g and deliver $\widehat{|\mathcal{X}^-|}$ according to (7).

The best choice for g is, clearly, $g^*(\boldsymbol{x}) = 1/|\mathcal{X}^-|$, $\boldsymbol{x}\in\mathcal{X}^-$, which is the *uniform distribution on* $\mathcal{X}^-$. Under g^* the estimator has *zero variance*, since the random

variable $\widehat{|\mathcal{X}^-|} = const$, so that only *one sample is required*. However, sampling from such g^* is impractical, since it requires availability of our target value $|\mathcal{X}^-|$. To overcome this difficulty we shall in the following sections how to construct "good" (law variance) IS sample pdf's (parametric and nonparametric) for different #P-complete counting problems, like counting the number of Hamiltonian cycles, the permanent, the number self-avoiding walks in a graph and the satisfiability problem.

3 Fast Calculation of the Number of Hamiltonian Cycles using CE

In this section we show how to estimate efficiently the number of Hamiltonian cycles in a graph using CE. The results of this section provide the basis for efficient calculation of other #P-complete counting problems, like calculating the permanent and the number of self-avoiding random walks.

A *Hamiltonian cycle* (HC) of a graph is a closed directed path that goes through all n nodes of the graph exactly once. Suppose the nodes of the graph are labelled $1, \ldots, n$. The *incidence matrix* of the graph is an $n \times n$ matrix $A = (a_{ij})$, with $a_{ij} = 1$ if there is a (directed link) from i to j; otherwise $a_{ij} = 0$. For example, consider the following incidence matrix

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}. \tag{8}$$

Note that this graph has 3 Hamiltonian cycles. Figure 2 depicts the graph and one HC, which is emphasized by thicker lines.

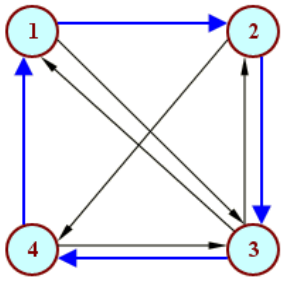

Figure 2: How many Hamiltonian cycles has the graph?

For a given matrix A we can view the HC counting problem as a particular case of the problem of counting the total number of tours in a TSP of length n. Generating tours in such TSP can done, for example, by using the standard trajectory generation algorithm [15] (see Algorithm 12.2 in Appendix 1). Note that Algorithm 12.2, in fact, generates permutations of $(1, \ldots, n)$, fixing the first component to 1, which gives $(n-1)!$ possible permutations. Note also that one of the main reasons for using trajectory generation according to Algorithm 12.2 is to make the trajectories "self-avoiding", that is to insure that no vertex (entity) is visited more than once. We write $f(\boldsymbol{x}, \boldsymbol{P})$ to indicate that the sampling pdf on $\mathcal{X}$ is parameterized by $\boldsymbol{P}$. If not stated otherwise we assume that the elements of $\boldsymbol{P}$ satisfy: $p_{ij} > 0$ if $i \neq j$, and $p_{ii} = 0$. In such case, the uniform pdf $f(\boldsymbol{x}, \boldsymbol{P})$

corresponds to

$$\boldsymbol{P}_{\mathsf{U}} = \begin{pmatrix} 0 & \frac{1}{n-1} & \cdots & \frac{1}{n-1} \\ \cdots & \cdots & & \cdots \\ \frac{1}{n-1} & \cdots & 0 & \frac{1}{n-1} \\ \frac{1}{n-1} & \cdots & \frac{1}{n-1} & 0 \end{pmatrix}. \tag{9}$$

Assuming without loss of generality that $x_1 = 1$, we can write $f(\boldsymbol{x}, \boldsymbol{P}_{\mathsf{U}})$ as

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{P}_{\mathsf{U}}) &= 1 \cdot f_2(x_2 \,|\, x_1) \cdots f_n(x_n \,|\, x_1, \ldots, x_{n-1}) \\ &= \frac{1}{n-1} \cdot \frac{1}{n-2} \cdots 1 = \frac{1}{(n-1)!} = \frac{1}{|\mathcal{X}|}. \end{aligned} \tag{10}$$

Similarly, for any trajectory, also called path $(1, x_2, \ldots, x_n)$,

$$f(\boldsymbol{x}, \boldsymbol{P}) = p_{1x_2} \, \frac{p_{x_2x_3}}{1 - p_{1x_2}} \, \frac{p_{x_3x_4}}{1 - p_{1x_2} - p_{x_2x_3}} \cdots 1 \, . \tag{11}$$

To relate the HC counting problem to the general estimation framework in Section 2 we use the following notations:

1. $\mathcal{X}$ is the entire set of tours in the graph. Note that $|\mathcal{X}| = (n-1)!$
2. $\mathcal{X}^-$ is the subset of $\mathcal{X}$ containing only tours of length n. A tour in $\mathcal{X}$ is said to be *valid* if it belongs to $\mathcal{X}^-$.

Example 3.1 Consider the graph in Figure 2, with incidence matrix A in (8). We have 6 trajectories, 3 of which ($1 \to 2 \to 3 \to 4 \to 1$; $1 \to 2 \to 4 \to 3 \to 1$; and $1 \to 3 \to 2 \to 4 \to 1$) are valid, while the remaining 3 are invalid.

If we set the transition matrix $\boldsymbol{P}_{\mathsf{U}}$ as

$$\boldsymbol{P}_{\mathsf{U}} = \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}, \tag{12}$$

then the trajectories are generated *uniformly*. That is,

$$f(\boldsymbol{x}, \boldsymbol{P}_{\mathsf{U}}) = \frac{1}{3} \times \frac{1}{2} \times 1 = \frac{1}{6} = \frac{1}{|\mathcal{X}|}, \quad \text{for all } \boldsymbol{x} \in \mathcal{X} \, . \tag{13}$$

Based on trajectories generated from $f(\boldsymbol{x}, \boldsymbol{P}_{\mathsf{U}})$ we can estimate $\ell = 3/6 = 1/2$ and $|\mathcal{X}^*| = 3$ using (4) and (7), respectively.

Using the IS pdf $f(\boldsymbol{x}, \boldsymbol{P})$ formulas (5)- (7) can be written as

$$\ell = \mathbb{E}_{\boldsymbol{P}}[I_{\{\boldsymbol{X} \in \mathcal{X}^-\}} W(\boldsymbol{X})], \tag{14}$$

$$|\mathcal{X}^-| = \mathbb{E}_{\boldsymbol{P}}[I_{\{\boldsymbol{X} \in \mathcal{X}^-\}} \frac{1}{f(\boldsymbol{X}, \boldsymbol{P})}], \tag{15}$$

and

$$\widehat{|\mathcal{X}^-|} = \frac{1}{N} \sum_{k=1}^{n} I_{\{\boldsymbol{X}_k \in \mathcal{X}^-\}} \frac{1}{f(\boldsymbol{X}_k, \boldsymbol{P})} = \sum_{\boldsymbol{X}_k \in \mathcal{X}^-} \frac{1}{f(\boldsymbol{X}_k, \boldsymbol{P})} \, , \tag{16}$$

respectively. Here

$$W(\boldsymbol{X}) = \frac{f(\boldsymbol{X}, \boldsymbol{P}_{\mathsf{U}})}{f(\boldsymbol{X}, \boldsymbol{P})}$$

presents the *likelihood ratio* (LR), the subscript $\boldsymbol{P}$ in $\mathbb{E}_{\boldsymbol{P}}$ means that the random trajectories $\boldsymbol{X}_k,\ k = 1,\ldots,N$ are generated according to Algorithm 12.2 using the IS pdf $f(\boldsymbol{x},\boldsymbol{P})$, the pdf's $f(\boldsymbol{x},\boldsymbol{P}_{\mathsf{U}})$ and $f(\boldsymbol{x},\boldsymbol{P})$ are defined in (10) and (11), respectively, and as before, the event $\{\boldsymbol{X}_k \in \mathcal{X}^-\}$ means that the trajectory $\boldsymbol{X}_k$ is valid. Note that

1. $\widehat{|\mathcal{X}^-|}$ in (16) can also be written as

$$\widehat{|\mathcal{X}^-|} = \frac{1}{N}\sum_{k=1}^{N} I_{\{\boldsymbol{X}_k \in \mathcal{X}^-\}} H(\boldsymbol{X}_k) = \sum_{\boldsymbol{X}_k \in \mathcal{X}^-} H(\boldsymbol{X}_k), \tag{17}$$

   where

$$H(\boldsymbol{X}_k) = \frac{1}{f(\boldsymbol{X}_k,\boldsymbol{P})}. \tag{18}$$

2. Both $W(\boldsymbol{X}_k)$ and $H(\boldsymbol{X}_k)$ present likelihood ratio terms and $H(\boldsymbol{X}_k)$ can be viewed as particular case of $W(\boldsymbol{X}_k)$ with $f(\boldsymbol{X}_k,\boldsymbol{P}_{\mathsf{U}}) = 1$.

3. For a complete graph, that is for $\mathcal{X}^- = \mathcal{X}$, we have

$$|\mathcal{X}^-| = \frac{1}{f(\boldsymbol{x},\boldsymbol{P}_{\mathsf{U}})} = (n-1)(n-2)\cdots 1 = (n-1)!.$$

   This implays that *simulating a* **single trajectory** *from the uniform pdf* $f(\boldsymbol{x},\boldsymbol{P}_{\mathsf{U}})$ *one automatically obtains the exact total number of trajectories* $|\mathcal{X}^-| = (n-1)!$. In other words, for a complete graph, the naive *crude Monte Carlo* (CMC) pdf $f(\boldsymbol{x},\boldsymbol{P}_{\mathsf{U}})$ coincides with the optimal IS pdf g^* and, thus, produces an estimator with *zero variance*.

When the graph is not complete, $f(\boldsymbol{x},\boldsymbol{P}_{\mathsf{U}})$ is no longer equal to g^*. In that case it makes sense to chose the matrix $\boldsymbol{P}$ in the IS pdf $g(\cdot) = f(\boldsymbol{x},\boldsymbol{P})$ in some optimal way, say by minimizing the variance of the estimator $\widehat{|\mathcal{X}^-|}$ in (17) or by using the *cross-entropy* (CE) method [15]. We shall use the latter.

Consider the CE optimal IS pdf for estimating $\ell = \mathbb{P}_{\mathsf{U}}(S(\boldsymbol{X}) \geq n)$, that is, the pdf $f(\boldsymbol{x},\boldsymbol{P}^*)$, with the elements p^*_{ij} of $\boldsymbol{P}^*$ equal to

$$p^*_{ij} = \frac{\mathbb{E}_{\mathsf{U}}[I_{\{S(\boldsymbol{X})\geq n\}} I_{\{\boldsymbol{X}_k \in \mathcal{X}_{ij}\}}]}{\mathbb{E}_{\mathsf{U}}[I_{\{S(\boldsymbol{X})\geq n\}}]} = \mathbb{P}_{\mathsf{U}}(\boldsymbol{X} \in \mathcal{X}_{ij} \,|\, S(\boldsymbol{X}) = n)\,. \tag{19}$$

Here $\mathcal{X}_{ij}$ is the set of trajectories that make the transition from i to j and $S(\boldsymbol{x})$ is the length of a Hamiltonian cycle corresponding to $\boldsymbol{x} \in \mathcal{X}$, that is,

$$S(\boldsymbol{x}) = \sum_{i=1}^{n-1} a_{x_i,x_{i+1}} + a_{x_n,1}\,.$$

Clearly, $0 \leq S(\boldsymbol{x}) \leq n$. We shall use both events $\{\boldsymbol{X} \in \mathcal{X}^-\}$ and $\{S(\boldsymbol{X}) = n\}$ interchangeably.

As usual with CE, we can estimate adaptively the probability matrix $\boldsymbol{P}^* = (p^*_{ij})$ by starting with some $\boldsymbol{P}_0 = (p_{0,ij})$, for example $\boldsymbol{P}_0 = \boldsymbol{P}_{\mathsf{U}}$, and then update $\boldsymbol{P}$ using the standard CE updating rule (see (57))

$$\widehat{p}_{t,ij} = \frac{\displaystyle\sum_{k=1}^{N} I_{\{\boldsymbol{X}_k \in \mathcal{X}_{ij}\}}\, I_{\{S(\boldsymbol{X}_k)\geq \widehat{\gamma}_t)\}}\; W(\boldsymbol{X}_k,\boldsymbol{P}_{\mathsf{U}},\widehat{\boldsymbol{P}}_{t-1})}{\displaystyle\sum_{k=1}^{N} I_{\{S(\boldsymbol{X}_k)\geq \widehat{\gamma}_t)\}}\; W(\boldsymbol{X}_k,\boldsymbol{P}_{\mathsf{U}},\widehat{\boldsymbol{P}}_{t-1})}, \tag{20}$$

where, as before, the LR term

$$W(\boldsymbol{X}_k, \boldsymbol{P}_{\mathsf{U}}, \widehat{\boldsymbol{P}}_{t-1}) = \frac{f(\boldsymbol{X}_k, \boldsymbol{P}_{\mathsf{U}})}{f(\boldsymbol{X}_k, \widehat{\boldsymbol{P}}_{t-1})}.$$

Note that the likelihood ratio W in $\widehat{p}_{t,ij}$, appears because the $\{\boldsymbol{X}_k\}$ are generated from $f(\boldsymbol{x}, \widehat{\boldsymbol{P}}_{t-1})$ and not from the uniform pdf.

We can next estimate $|\mathcal{X}^-|$ *indirectly* by multiplying $|\mathcal{X}|$ with the IS estimate $\widehat{\ell}$ based on (5) using an estimate $\widehat{\boldsymbol{P}}_t$ (see (20)) obtained after, say $t = T$ iterations of the CE algorithm.

Since $f(\boldsymbol{x}, \boldsymbol{P}_{\mathsf{U}}) = 1/|\mathcal{X}|$, we can rewrite (20) as

$$\widehat{p}_{t,ij} = \frac{\sum_{k=1}^{N} I_{\{\boldsymbol{X}_k \in \mathcal{X}_{ij}\}}\, I_{\{S(\boldsymbol{X}_k) \geq \widehat{\gamma}_t\}} H(\boldsymbol{X}_k, \widehat{\boldsymbol{P}}_{t-1})}{\sum_{k=1}^{N} I_{\{S(\boldsymbol{X}_k) \geq \widehat{\gamma}_t\}} H(\boldsymbol{X}_k, \widehat{\boldsymbol{P}}_{t-1})}, \tag{21}$$

where as before (see (18))

$$H(\boldsymbol{X}_k, \widehat{\boldsymbol{P}}_{t-1}) = \frac{1}{f(\boldsymbol{X}_k, \widehat{\boldsymbol{P}}_{t-1})}$$

and estimate $|\mathcal{X}^-|$ *directly* using (17), (with $\boldsymbol{P}$ replaced by $\widehat{\boldsymbol{P}}_T$), that is

$$\widehat{|\mathcal{X}^-|} = \sum_{\boldsymbol{X}_k \in \mathcal{X}^-} H(\boldsymbol{X}_k, \widehat{\boldsymbol{P}}_T), \tag{22}$$

where T denotes the total number of iterations.

We shall use below *only* the direct approach. Note that in order to construct the optimal $\boldsymbol{P}^*$ one first needs to compute the number n_{ij} of *valid* trajectories passing through each transition (ij) of $\boldsymbol{P}^*$ and then normalize the $\{n_{ij}\}$ row-wise. Although such policy assumes a total enumerations of all $|\mathcal{X}^-|$ trajectories and it has no practical value, it nevertheless provides a clear insight on how to estimate efficiently from simulation $\boldsymbol{P}^*$ and, thus $|\mathcal{X}^-|$. The following example provides details.

Example 3.2 Consider again the matrix A in (8). The counting matrix $\mathcal{N} = (n_{ij})$ associated with 3 valid trajectories $1 \to 2 \to 3 \to 4 \to 1$; $1 \to 2 \to 4 \to 3 \to 1$ and $1 \to 3 \to 2 \to 4 \to 1$ is

$$\mathcal{N} = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 \\ 1 & 1 & 0 & 1 \\ 2 & 0 & 1 & 0 \end{pmatrix}. \tag{23}$$

Normalizing (23) row-wise we obtain

$$\boldsymbol{P}^* = \begin{pmatrix} 0 & \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{2}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{2}{3} & 0 & \frac{1}{3} & 0 \end{pmatrix}. \tag{24}$$

Simple calculation yield that $\boldsymbol{P}^*$ if better than $\boldsymbol{P}_{\mathsf{U}}$ in the sense that the variance of the random variable $I_{\{\boldsymbol{X} \in \mathcal{X}^-\}} H(\boldsymbol{X})$ under $f(\boldsymbol{x}, \boldsymbol{P}^*)$ is smaller than under $f(\boldsymbol{x}, \boldsymbol{P}_{\mathsf{U}})$.

Remark 3.1 Taking into account that the sample function S (and thus γ_t) accepts *only* integers from the set $\{0, 1, \ldots, n\}$, and that our ultimate goal is to generate and calculate only the valid trajectories, it follows that after several iterations of the modified Algorithm 3.1, most of generated values of S will be equal n. It is clear that all such trajectories with $S = n$ should be *included in the elite sample*. As an example, suppose that $N = 10$, $\rho = 0.2$, and the ordered sample values of $S(\boldsymbol{X}$ are 2, 3, 3, 4, 5, 6, 6, 6, 6, 6. In this case $\lfloor \rho N \rfloor = 2$ and by definition the 80% sample quantile equals to $\gamma = 6$ (the 80% sample quantile γ is such that 80% or more of the performances are $\geq \gamma$ and 20% or more of the performances are $\leq \gamma$). Thus, the elite set $\mathcal{E} = \{\boldsymbol{X}_{\|} : \mathcal{S}(\boldsymbol{X}_{\|}) \geq \gamma\}$ has size $N^{\text{elite}} = 5$, and Clearly, $N^{\text{elite}} > \lfloor \rho N \rfloor$. In general $N^{\text{elite}} \geq \lfloor \rho N \rfloor$.

We denote the final estimates of $\boldsymbol{P}^*$ and $|\mathcal{X}^-|$ in our direct Algorithm 3.1 below as $\widehat{\boldsymbol{P}}_T$ and $\widehat{|\mathcal{X}^-|}$, respectively.

Algorithm 3.1 (The Cross-Entropy Counting Algorithm)

1. Choose some $\widehat{\boldsymbol{P}}_0 = \boldsymbol{P}_0 = (p_{0,ij})$, say $p_{0,ij} = \frac{1}{n-1}$, if $i \neq j$ and $p_{0,ii} = 0$. Set $t = 1$ (level counter).

2. Generate a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from the density $f(\boldsymbol{x}, \widehat{\boldsymbol{P}}_{t-1})$. Calculate the performances $\{S(\boldsymbol{X}_k)\}$, let $\widehat{\gamma}_t$ be the sample $(1-\rho)$ quantile of thes of these performances, and define the elite set as $E = \{\boldsymbol{X}_k : S(\boldsymbol{X}_k) \geq \widehat{\gamma}_t\}$, which corresponds to $\widehat{\gamma}_t = S_{(N - N^{\text{elite}}+1)}$. Denote $\widehat{\gamma}_t^* = \max S(\boldsymbol{X}_k),\ k = 1, \ldots, N = S_N$.

3. Use the **same** sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ and compute the estimate of $\boldsymbol{P}_t = (p_{t,ij})$ according to (21). Denote the solution by $\widetilde{\boldsymbol{P}}_t$.

4. Apply
$$\widehat{\boldsymbol{P}}_t = \alpha \widetilde{\boldsymbol{P}}_t + (1-\alpha)\widehat{\boldsymbol{P}}_{t-1} \tag{25}$$
to smooth out the matrix $\widetilde{\boldsymbol{P}}_t$. Here α, $(0 < \alpha < 1)$ is called the *smoothing parameter*.

5. If the largest tour $\widehat{\gamma}_t^* < n$, set $t = t+1$ and reiterate from step 2. Else proceed with step 6.

6. Estimate $|\mathcal{X}^-|$ according to the (22).

Remark 3.2 Algorithm 3.1 is written with the view that *there exist at least a single valid trajectory*. If, hovewer, the inequality $\widehat{\gamma}_t^* < n$ in Step 5 concentrates around some fixed value $\widehat{\gamma}_t^* = n - k$ $(k \geq 1)$, and its holds, say for 5 iterations in turn, one should stop the algorithm, delivering $|\mathcal{X}^-| = 0$, which means *there is no single valid trajectory in the graph*.

If there exist only a *single* valid trajectory, then Algorithm 3.1 reduces to an algorithm for solving TSP [15] in the sense that the pdf $f(\boldsymbol{x}, \widehat{\boldsymbol{P}}_T)$ becomes a degenerated one with all mass concentrated at that single trajectory and, as result, the step 6 of Algorithm 3.1 becomes redundant.

Remark 3.3 A better way of choosing $\boldsymbol{P}_0 = (p_{0,ij})$ in step 1 of the algorithm is to assign smaller initial probabilities $p_{0,ij}$ to the elements of $\boldsymbol{P}_0$ associated with zero elements of the matrix A rather then to these associated with unity elements. In particular,let k_i be the number of 1's at i-ts row of the matrix A. Then we can proceed as follows

1. Set $p_{0,ij} = \frac{1-\delta}{k_i}$, if the corresponding element a_{ij} of the distance matrix A equals unity, and set the remaining elements $p_{0,ij} = \varepsilon$. Here ε is a small number obtained from the solution of the equation that $(n - k_i) \times \varepsilon = \delta$, where, say $\delta = 0.1$. That is to we set $p_{0,ij} = \varepsilon(k_i) = \frac{\delta}{n-k_i}$ each element of $\boldsymbol{P}_0$ corresponding to 0 element of the i-th row of A.

2. Keep the above $p_{0,ij} = \varepsilon(k_i) = \frac{\delta}{n-k_i}$ for *all iterations* of Algorithm 3.1.

Since for all iterations $t < T$ the elements $p_{t,ij}$ corresponding to the i-th row of A are assumed to be $p_{t,ij} = \varepsilon(k_i) = \frac{\delta}{n-k_i}$ one can speed up the trajectory generation Algorithm 12.2 by arguing as follows. Consider the probability matrix $\boldsymbol{P}_t$ at iteration $t < T$. The straightforward way for generating trajectories according to Algorithm 12.2 is just to use $\boldsymbol{P}_t$ as it is. However, since δ is the sum of all $p_{t,ij}$'s corresponding to the zero elements in the i-th row of A and since all such $p_{t,ij}$ are equal each other, we can generate a transition from each state i using only an $k_i + 1$ point discrete pdf rather than the entire n point one. Indeed, consider the i-th row of $\boldsymbol{P}_t$ and let as before $\sum_{i=1}^{k_i} p_{t,ij} = 1 - \delta$. Relabel next the elements $p_{t,ij}$ of the row i such that the first k_i ones will correspond to the 1's in A, while the remaining $n - k_i$ ones corresponding to 0's in A will be labeled as the last ones. With this to hand, a faster procedure for generating a transition from state i to j is as follows.

Algorithm 3.2 [A Fast Procedure for Generating Trajectories]

1. Generate a random variable U uniformly distributed on the interval $(0, 1)$.

2. If $U \leq 1 - \delta$, renormalize the sum $\sum_{i=1}^{k_i} p_{t,ij} = 1 - \delta$ by setting $\sum_{i=1}^{k_i} p_{t,ij}^{(n)} = 1$, where
$$p_{t,ij}^{(n)} = \frac{p_{t,ij}}{1 - \delta}$$
(recall that the first k_i elements of the i-th row of $\boldsymbol{P}_t$ correspond to 1's in A) and generate from the discrete k_i point pdf with probabilities $p_{t,ij}^{(n)}$ and proceed with Algorithm 12.2 as usual.

3. If $U > 1-\delta$, generate a discrete random variable Z uniformly distributed over the points $k_i + 1, \ldots, n$ (recall that the last $n - k_i$ elements of the i-th row of $\boldsymbol{P}_t$ correspond to 0's in A). As soon as the new state $k_i + s,\ s = 1, \ldots, n - k_i$ is selected, make a transition from it according to Algorithm 12.2.

Remark 3.4 It is obvious that if $a_{ij} = 0$, then the corresponding elements p_{ij}^* of the resulting (unknown) optimal matrix $\boldsymbol{P}^*$ should be zero. However, Algorithm 3.1 generates a matrix $\widehat{\boldsymbol{P}}_T$ (an estimate of $\boldsymbol{P}^*$) with elements $\widehat{p}_{T,ij}$ (associated with $a_{ij} = 0$) being only near to zero, since we take all of diagonal elements $p_{0,ij}$ of $\boldsymbol{P}_0$ to be positive. These close to zero elements should be automatically set to zero before performing the final step 6 of Algorithm 3.1 and the resulting matrix $\widehat{\boldsymbol{P}}_T$ should be renormalized. In short, before stopping the algorithm it is desirable to

1. Set to zero all non-zero elements of the resulting matrix $\widehat{\boldsymbol{P}}_T$ corresponding to the zero elements of A.

2. Renormalize the matrix $\widehat{\boldsymbol{P}}_T$.

3. Run Algorithm 3.1 for a couple of more iteration and only then proceed to step 6.

3.1 The Original Approach

In principle, any transition matrix $\boldsymbol{P}_0$ can serve as starting matrix in Algorithm 3.1, provided that all $\boldsymbol{x} \in \mathcal{X}^-$ (valid trajectories) can be generated with strictly positive probability. The choice $p_{0,ij} = 1/(n-1)$ for $i \neq j$ and $p_{ii} = 0$ seems to be natural because it results in a uniform generation of the trajectories. However, by doing so most of the generated trajectories (even with dense matrices A) will not be valid, since ℓ in (2) is typically very small. A better way is to either assign small positive probabilities $p_{0,ij}$ for all of diagonal elements of the initial matrix $\boldsymbol{P}_0$ (see Remark 3.3) for which $a_{ij} = 0$, or to set these probabilities to zero. In the latter case the remaining non-zero $p_{0,ij}$ elements could be chosen equal, that is $p_{ij} = 1/d_i$, where d_i is the degree of i. Clearly, in this case instead of the sampling from a huge space $\mathcal{X}$ (with cardinality $|\mathcal{X}| = (n-1)!$) we shall sample from a much smaller one. We shall call such approach, the *original* approach to distinguish it from the one proposed in Remark 3.3, which we call the *auxiliary* one.

Denote the initial matrix associated with the original approach by $\boldsymbol{P}_0^{(o)}$. Note that $\boldsymbol{P}_0^{(o)}$ typically does not lead to a uniform generation of cycles and, thus $f(\boldsymbol{x}, \boldsymbol{P}_0^{(o)})$ should be viewed as an IS pdf. Moreover, it is important to realize that in this case not all trajectories generated by Algorithm 12.2 will pass through all cities, that is, not all trajectories will loop back to the starting city. In fact, many of these trajectories will stop at some intermediate city. As in Algorithm 3.1 we shall *not* reject automatically such invalid trajectories but rather keep them in the pull of all trajectories (valid and invalid) with the view that some of them (the ones with longer trajectories) will be included into the elite sampling to provide smooth convergence of γ_t to n. Clearly, as Algorithm 3.1 evolves the event $\{\boldsymbol{X}_i \in \mathcal{X}^-\} = \{S(\boldsymbol{X}_i) = n\}$ will occur more often and if the number of valid trajectories (events $\{S(\boldsymbol{X}_i) = n\}$) exceeds ρN, Algorithm 3.1 will automatically reject the invalid ones.

To speed up the convergence process one can introduce the rejection sampling into Algorithm 3.1 in the sense that is if, say an invalid trajectory terminates quite earlier, Algorithm 3.1 may go one transition back and sample from $f(\boldsymbol{x}, \boldsymbol{P})$ again. We shall call such acceptance-rejection approach, the *backtracking* one.

Example 3.3 Consider again the incidence matrix A in (8) (see also Example 3.1). The original approach suggests taking $\boldsymbol{P}_0^{(o)}$ as

$$\boldsymbol{P}_0^{(o)} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{pmatrix} \tag{26}$$

It is not difficult to check that $\boldsymbol{P}_0^{(o)}$ is already "close" to $\boldsymbol{P}^*$ in the sense that the variance of the random variable $I_{\{\boldsymbol{X} \in \mathcal{X}^-\}} H(\boldsymbol{X})$ under $f(\boldsymbol{x}, \boldsymbol{P}_0^{(o)})$ is quite close to the one under $f(\boldsymbol{x}, \boldsymbol{P}^*)$.

Remark 3.5 Note that since not all trajectories passing through the unity elements of the matrix A will be valid, same of the elements $\widehat{p}_{t,ij}$ will become very small after several iterations of Algorithm 3.1. Such small elements, which called the *redundant* elements, should be set automatically to zero on-line (in the course of simulation) and the resulting matrix $\widehat{\boldsymbol{P}}_T$ should be renormalized before estimate $|\mathcal{X}^-|$. An alternative (off-line procedure) for eliminating such redundant elements is based on the screening algorithm proposed in [9]. Basically, the screening algorithms calculates (estimates) all partial derivatives of the objective function $S(\boldsymbol{P})$ with

respect to all underlying parameter vector (matrix) $\boldsymbol{P}$ and eliminates the ones for which the corresponding derivatives are very small.

We shall see from our numerical results below that for not too sparse matrices A the IS pdf $f(\boldsymbol{x}, \boldsymbol{P}_0^{(o)})$ can be used directly in (22) (instead of $f(\boldsymbol{X}_k, \widehat{\boldsymbol{P}}_T)$ to estimate the desiged quantity $|\mathcal{X}^-|$, and, thus to avoid steps 1-5 of Algorithm 3.1.

4 Calculating the Permanent

The permanent of a general $n \times n$ matrix $A = (a_{ij})$ is defined as

$$\text{perm}(A) = |\mathcal{X}^-| = \sum_{\boldsymbol{x} \in \mathcal{X}} \prod_{i=1}^{n} a_{ix_i}, \tag{27}$$

where $\mathcal{X}$ is the set of all permutations $\boldsymbol{x} = (x_1, \ldots, x_n)$ of $(1, \ldots, n)$.

Example 4.1 Let A be 3×3 matrix given as

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}. \tag{28}$$

Note that in contrast to the HC problem the diagonal elements of the matrix A do not need to be zeros. The $3! = 6$ permutations with the corresponding product values $\prod_{i=1}^{3} a_{ix_i}$ are given in the table below. The permanent is $|\mathcal{X}^-| = 3$.

Table 1: Six permutations and their product values of the matrix A in (28)

$\boldsymbol{x}$			$\prod_{i=1}^{3} a_{ix_i}$	
1	2	3	$a_{11}\ a_{22}\ a_{33}$	$= 1$
1	3	2	$a_{11}\ a_{23}\ a_{32}$	$= 0$
2	1	3	$a_{12}\ a_{21}\ a_{33}$	$= 1$
2	3	1	$a_{12}\ a_{23}\ a_{31}$	$= 0$
3	1	2	$a_{13}\ a_{21}\ a_{32}$	$= 1$
3	2	1	$a_{13}\ a_{22}\ a_{31}$	$= 0$

The procedure of fast calculating (estimating) the permanent of a binary matrix is quite similar to the one for calculating the number of HC's. However, unlike the HC problem, where the trajectories are typically generated via node transitions (see Algorithm 12.2), the natural way to tackle the permanent is via *node placements* (see Algorithm 12.3).

The reason why the trajectories for the permanent problem should be generated via a node placement algorithm rather than a node transitions algorithm is exemplified by the fact that the performance functions for the permanent and HC problems can be written as

$$S(\boldsymbol{x}) = \sum_{i=1}^{n} a_{x_i} \tag{29}$$

and

$$S(\boldsymbol{x}) = \sum_{i=1}^{n} a_{x_i x_{i+1}}, \tag{30}$$

respectively. Note that although in both cases we are interested in permutations $\boldsymbol{x} \in \mathcal{X}^-$ for which $S(\boldsymbol{x}) = n$, the former $S(\boldsymbol{x})$ contains terms of the form a_{ix_i} , while the letter, terms of the form $a_{x_i x_{i+1}}$.

Consider the matrix A in (28). The function $S(\boldsymbol{x})$ values with all 6 possible permutation vectors $\boldsymbol{x}$ is given in Table 2.

Table 2: The S values corresponding to the permutations for the matrix A in (28).

$S(\boldsymbol{x})$
$a_{11} + a_{22} + a_{33} = 3$
$a_{11} + a_{23} + a_{32} = 2$
$a_{12} + a_{21} + a_{33} = 3$
$a_{12} + a_{23} + a_{31} = 1$
$a_{13} + a_{21} + a_{32} = 3$
$a_{13} + a_{22} + a_{31} = 2$

Writing $|\mathcal{X}^-|$ as $\mathbb{E}_{\boldsymbol{P}}[I_{\{S(\boldsymbol{X})\geq n\}}\, H(\boldsymbol{X})]$, where the permutations are generated via the node-placement algorithm parameterized by a matrix $\boldsymbol{P}$, we can estimate $|\mathcal{X}^-|$ via (7). The ellements of the CE optimal matrix $\boldsymbol{P}^* = (p^*_{ij})$ satisfy

$$p^*_{ij} = \frac{\mathbb{E}_{\mathsf{U}}[I_{\{S(\boldsymbol{X})\geq n\}} I_{\{X_i=j\}}]}{\mathbb{E}_{\mathsf{U}}[I_{\{S(\boldsymbol{X})\geq n\}}} = \frac{\mathbb{E}_{\boldsymbol{P}}[I_{\{S(\boldsymbol{X})\geq n\}} I_{\{X_i=j\}} H(\boldsymbol{X})]}{\mathbb{E}_{\boldsymbol{P}}[I_{\{S(\boldsymbol{X})\geq n\}} H(\boldsymbol{X})]},$$

where $H(\boldsymbol{X}) = 1/f(\boldsymbol{x}, \boldsymbol{P})$, and can be determined adaptively via the updating formula

$$\widehat{p}_{t,ij} = \frac{\sum_{\boldsymbol{X}_k \in \mathcal{E}_t} I_{\{X_{ki}=j\}} H(\boldsymbol{X}_k)}{\sum_{\boldsymbol{X}_k \in \mathcal{E}_t} H(\boldsymbol{X}_k)}\,. \tag{31}$$

Example 4.2 Consider the permanent with $n = 4$ and the matrix A given as

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}. \tag{32}$$

In this case we have the following five valid permutation vectors: $\boldsymbol{x}_1 = (4, 3, 2, 1)$, $\boldsymbol{x}_2 = (4, 1, 2, 3)$, $\boldsymbol{x}_3 = (3, 1, 4, 2)$, $\boldsymbol{x}_4 = (2, 3, 4, 1)$ and $\boldsymbol{x}_5 = (2, 1, 4, 3)$. It is readily seen that

1. The optimal CE matrix $\boldsymbol{P}^*$ is

$$\boldsymbol{P}^* = \begin{pmatrix} 0 & \frac{2}{5} & \frac{1}{5} & \frac{2}{5} \\ \frac{3}{5} & 0 & \frac{2}{5} & 0 \\ 0 & \frac{2}{5} & 0 & \frac{3}{5} \\ \frac{2}{5} & \frac{1}{5} & \frac{2}{5} & 0 \end{pmatrix}. \tag{33}$$

2. The probabilities to generate $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_5$ (under $\boldsymbol{P}^*$) are $4/25, 6/25, 3/25, 4/25$ and $6/25$, respectively. The only invalid permutation vector that can be generated is $(3, 1, 2, 4)$, and this occurs with probability $2/25$.

3. Under $\boldsymbol{P}^*$ the variance of the random variable $I_{\{S(\boldsymbol{X})\geq 4\}} H(\boldsymbol{X})$ is $(175/6) - 52 = 25/6$, while the variance under the uniform pdf is $60 - 25 = 35$, so there is a substantial improvement.

Remark 4.1 Note that Algorithm 3.1 can be applied to the case where the elements of the permanent matrix A attain arbitrary values, rather than 0's and 1's.

5 Self-Avoiding Walk

In a self-avoiding walk (SAW) on a latice we are interested to know how many trajectories of length n exist without looping while starting at $\boldsymbol{x} = (x_1, x_2) = (0, 0)$ and we are allowed to move at each grid point to any of the four neighboring direction with equal probability.

Note that starting at $(x_1, x_2) = (0, 0)$ there are 4 SAW's each coinciding with the respective positive and negative coordinates of x_1 and x_2; the rest are inside the box of length $2n$ with the center $(0, 0)$ and they are symmetric with respect to the coordinates x_1 and x_2.

For SAW's we can again apply Algorithm 3.1, where

1. The number of states equals to $4 \times n^2 \times n^2$.
2. Each row of the associate cost matrix $A = (a_{rs})$ $r = 1, \ldots, n;\ s = 1, \ldots, n$, which is obtained *online* via trajectory generation, contains at most 4 unity elements, while the rest are 0's.
3. Each row of the probability matrix $\boldsymbol{P} = (p_{rs})$, $r = 1, \ldots, n;\ s = 1, \ldots, n$ (associate with $A = (a_{rs})$) contains at most 4 positive elements ($p_{rs} \geq 0$ if $a_{rs} = 1$), while the rest are 0's.

Example 5.1 Consider a SAW with $n = 3$. In this case we have $|\mathcal{X}^-| = 36$ valid trajectories. To see this consider a single trajectory starting at origin along the positive x coordinate. Denoting the points associated with this trajectory as $\{0,\ 1,\ 2,\ 3\}$, then the corresponding trajectory is $0 \to 1 \to 2 \to 3$. Proceeding we obtain that

1. There are 9, 3 and 1 trajectories passing trough the transition p_{01}, p_{12} and p_{23}, respectively.
2. The associated CE optimal probabilities p^*_{01}, p^*_{12} and p^*_{23} are $p^*_{01} = 9/36 = 1/4$, $p^*_{12} = 3/9 = 1/3$ and $p^*_{23} = 1/3$, respectively.
3. The optimal pdf is
$$f(\boldsymbol{x}, \boldsymbol{P}^*) = \frac{1}{4} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot 1 = \frac{1}{36}.$$
4. The cardinality is $|\mathcal{X}^-| = 36$.

6 The Connectivity Problem

Given two different nodes, say nodes A and B, in a not complete graph (directed or undirected) our goal is to find the number of trajectories from node A to B, provided that an edge can be selected in a trajectory at most once. Such problem is of importance in the fields of reliability and communication networks.

Note that in the connectivity problem, $|\mathcal{X}^-|$ counts the total number of trajectories of length $0, 1 \ldots, n-2$, rather than of length n alone as compared to the Hamiltonian cycle problem, that is γ obtains *multiple values*, namely $\gamma = 0, 1 \ldots, n-2$, rather than a single value, $\gamma = n$.

Note that counting problems with multiple values of γ occur in many applications including counting the number of spanning trees in a graph and in counting the permanent, where the elements of the matrix A are not restricted to 0's and 1's, see Remark 4.1.

6.1 Complete Graph

Consider a complete network. In this case it is not difficult to see that the total number of trajectories between two nodes in the network equals

$$|\mathcal{X}| = \sum_{k=0}^{n-2} k! \binom{n-2}{n-k-2}. \tag{34}$$

Below we show how to construct an optimal CE matrix $\boldsymbol{P}^*$ of $f(\boldsymbol{x}, \boldsymbol{P}^*)$ for a complete 5 node network. Such optimal $\boldsymbol{P}^*$ will provide insight on how to chose a good initial matrix $\boldsymbol{P}_0$, while estimating the desired quantity $|\mathcal{X}|^-$ in a not complete graph.

Example 6.1 Consider a complete 5 node network. From (34) we have that the total number of trajectories between any two nodes in the network, say nodes 1 and 5, is $|\mathcal{X}| = 16$. The detailed calculations are

$$\mathcal{X} = 0! \cdot \binom{3}{3} + 1! \cdot \binom{3}{2} + 2! \cdot \binom{3}{1} + 3! \cdot \binom{3}{0} = 1 + 3 + 6 + 6 = 16. \tag{35}$$

To find the optimal IS pdf $f(\boldsymbol{x}, \boldsymbol{P}^*)$ we shall associate with our graph the following 5×4 probability matrix $\boldsymbol{P}$.

$$\boldsymbol{P} = \begin{pmatrix} 0 & p_{12} & p_{13} & p_{14} & p_{15} \\ 0 & 0 & p_{23} & p_{24} & p_{25} \\ 0 & p_{32} & 0 & p_{34} & p_{35} \\ 0 & p_{42} & p_{43} & 0 & p_{45} \end{pmatrix}. \tag{36}$$

It is not difficult to see that the CE optimal $\boldsymbol{P}^*$ is

$$\boldsymbol{P}^* = \begin{pmatrix} 0 & 5/16 & 5/16 & 5/16 & 1/16 \\ 0 & 0 & 3/11 & 3/11 & 5/11 \\ 0 & 3/11 & 0 & 3/11 & 5/11 \\ 0 & 3/11 & 3/11 & 0 & 5/11 \end{pmatrix}. \tag{37}$$

Indeed, we have from (35) for $k = 0, 1, 2, 3$ the following 16 trajectories:

1. $k = 0$, trajectory $1 \to 5$.
2. $k = 1$, trajectories $1 \to 2 \to 5,\ 1 \to 3 \to 5,\ 1 \to 4 \to 5$.
3. $k = 2$, trajectories $1 \to 2 \to 3 \to 5,\ 1 \to 2 \to 4 \to 5,\ 1 \to 3 \to 4 \to 5,\ 1 \to 3 \to 2 \to 5,\ 1 \to 4 \to 2 \to 5,\ 1 \to 4 \to 3 \to 5$.
4. $k = 3$, trajectories $1 \to 2 \to 3 \to 4 \to 5,\ 1 \to 2 \to 4 \to 3 \to 5,\ 1 \to 3 \to 2 \to 4 \to 5,\ 1 \to 3 \to 4 \to 2 \to 5,\ 1 \to 4 \to 2 \to 3 \to 5,\ 1 \to 4 \to 3 \to 2 \to 5$.

For example, for the trajectories passing through the transition $1 \to 2$ we have that $n_{12} = 5$ (1 for $k = 1$, 2 for $k = 2$ and 2 for $k = 3$) among the 16 trajectories, and thus $p_{12} = 5/16$.

Taking (34) into account one can extend the result (37) to an arbitrary n and to find the exact analytic expression for the optimal matrix $\boldsymbol{P}^* = (p^*_{ij}),\ i = 1, \ldots, n-1;\ j = i+1, \ldots, n$ (element-wise for any n). The resulting expression (after some algebra) is

$$p^*_{i,j} = \frac{a_{i,j}}{b_i}, \tag{38}$$

where

$$a_{i,j} = \begin{cases} 0, & i = j \\ 1, & i = 0, j = n-1 \\ \sum_{k=0}^{n-3} \binom{n-3}{n-3-k} k!, & i = 0, j \neq n-1. i \neq j \\ \sum_{k=0}^{n-4} \binom{n-4}{n-4-k} (k+1)!, & i \neq 0, j \neq n-1, i \neq j \end{cases}$$

and

$$b_i = \begin{cases} \sum_{k=0}^{n-2} \binom{n-2}{n-2-k} k!, & i = 0 \\ \sum_{k=0}^{n-3} \binom{n-3}{n-3-k} (k+1)!, & i \neq 0 \end{cases}$$

6.2 Not Complete Graph

To find the optimal $\widehat{\boldsymbol{P}}_T^*$ and thus a good estimate of $|\mathcal{X}^-|$ in a not complete graph we can employ again Algorithm 3.1. Note, however, that in this case, any trajectory of length $1, \ldots, n$, which is able to pass from node A to B, will be a valid one. This in turn implies, that in contrast to HC, permanent and SAW problems, where all valid trajectories are of length n, the trajectory generation Algorithms 12.2 and 12.3 are not necessary appropriate for the connectivity problem and, thus alternative ones should be constructed. Although this can be considered as a subject of further research, we nevertheless present for completeness a simple example of calculating the CE optimal matrix $\boldsymbol{P}^*$ based on the trajectory generation Algorithm (12.2).

Example 6.2 Consider again the graph with $n = 4$ and adjacency matrix A in (32). To find the CE optimal probability matrix $\boldsymbol{P}^*$ and the cardinality$|\mathcal{X}|$ (the number of valid trajectories from node 1 to 4) note that we have for $k = 0, 1, 2$ the following 3 valid trajectories:

1. $k = 0$, trajectory $1 \to 4$.
2. $k = 1$, trajectory $1 \to 3 \to 4$,
3. $k = 2$, trajectory $1 \to 2 \to 3 \to 4$.

The optimal 4×3 matrix $\boldsymbol{P}^*$ is therefore

$$\boldsymbol{P}^* = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

7 Satisfiability Problem

An instance of the satisfiability (SAT) problem is a Boolean formula that has three components [1], [7]:

- A set of m variables $x_1, x_2, \ldots, x_m$.
- A set of literals. A literal is a variable ($q = x$) or a negotiation of a variable ($q = \bar{x}$).

- A set of n distinct clauses: $C_1, C_2, \ldots, C_n$ consisting of different literals.

We distinguish between the *conjunctive normal form* (CNF), where the clauses $C_1, C_2, \ldots, C_n$ are related via $\wedge$, that is

$$C_1 \wedge C_2 \wedge \cdots \wedge C_n,$$

where $(\wedge)$ is a logical *and* connectivity and the *disjunctive normal form* (DNF), where the clauses are related via $\vee$, where $(\vee)$ is a logical *or* connectivity.

Given a set of Boolean variables m and a set of clauses n, how many satisfying truth assignments $|\mathcal{X}^-|$ exists, i.e. how many ways $|\mathcal{X}^-|$ exist to set the variables $x_1, \ldots, x_m$ either true or false so that each clause contains at least one true literal?

For example, for the CNF SAT problem

$$\{x_1 \vee \bar{x}_2\} \wedge \{\bar{x}_1 \vee x_2\},$$

which is satisfied by setting $x_1 = x_2$ is true or $\bar{x}_1 = \bar{x}_2$ is false, its DNF equivalent is

$$\{\bar{x}_1 \wedge x_2\} \vee \{x_1 \wedge \bar{x}_2\}.$$

Remark 7.1 Extensions The SAT becomes more difficult if we allow *quantifiers*, such as "for all" $(\forall)$ and "there exist" that binds the Boolean variables [1]. Such problem is called *quantified boolean problem* . We shall consider the standard SAT.

To proceed, we shall

- Associate with each literal q a Bernoulli random variable with the parameter p.
- Choose $f(\boldsymbol{x}, \boldsymbol{p}_0)$ as the product of n iid Bernoulli marginal pdf's, each with the parameter, say $p_{0k} = 0.5,\ k = 1, \ldots, n$.

If not stated otherwise we shall consider the SAT problem in SNF assuming without loss of generality that each clause C_k is of length $r,\ r \leq n$.

Consider a clause C_k and define

$$\delta(\boldsymbol{q}_k) = \min\{1, \sum_{i=1}^{r} q_{ik}\},\ \boldsymbol{q}_k = (q_{1k}, \ldots, q_{rk}).$$

Clearly each $\delta(\boldsymbol{q}_k) = \delta_k \in \{0,\ 1\}$.

Define next

$$S(\boldsymbol{\delta}) = \sum_{k=1}^{n} \delta(\boldsymbol{q}_k) = \sum_{k=1}^{n} \delta_k,$$

where $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_n)$. Using $S(\boldsymbol{\delta})$, the rare-event probability ℓ in $|\mathcal{X}^-| = \ell|\mathcal{X}|$ can be written as

$$\ell = \mathbb{P}\{S(\boldsymbol{\delta}) = n\} = \mathbb{E}_{\boldsymbol{p}_0}\{I_{(\sum_{k=1}^{n} \delta_k = n)}\}. \tag{39}$$

It is not difficult to see that ℓ in (39) defines a probability that the generated SAT sample (trajectory) $\boldsymbol{\delta}$ is valid, that is $S(\boldsymbol{\delta})$ is of length n.

At this end we note that we reduced the SAT problem to probability of rare event estimation (the DNF case can be reduced to probability of rare even similarly) and we can proceed directly with Algorithm 3.1 by updating the probability vector $\boldsymbol{p}$ in order to estimate efficiently the number of valid trajectories $|\mathcal{X}^-|$.

As for examples, consider the following SAT's:

1. $\{x_1 \wedge \bar{x}_2\}$, which is satisfied by setting x_1 is true and $\bar{x}_2$ is false. We have $|\mathcal{X}^-| = 1$ and the optimal $\boldsymbol{p}^* = (1,\ 0)$.
2. $\{x_1 \vee \bar{x}_2\} \wedge \{\bar{x}_1 \vee x_2\}$, which is satisfied by setting $x_1 = x_2$ is true or $\bar{x}_1 = \bar{x}_2$ is true. We have $|\mathcal{X}^-| = 2$ and the optimal $\boldsymbol{p}^* = \boldsymbol{p}_0 = (0.5, 0.5)$.

8 Numerical Results using CE

We shall present here numerical results with Algorithm 3.1 for the *Hamiltonian cycle* (HC), the permanent, the SAW and the satisfiability problems using mainly the original approach. We found that the it is typically faster and more accurate then the auxiliary one, especialy for not too sparse matrices. To verify the validity of our numerical results for such (not too sparse matrices) we run ISO ($t = 0$) for a large sample, say $N = 1,000,000$, to make sure that the ISO estimator is accurate (relative error κ is very small).

If not stated otherwise, we set $\rho = 0.001$, $\alpha = 0.7$ and we use the same sample size while estimating both $\boldsymbol{P}$ and the $|\mathcal{X}^-|$. To study the variability in the solutions we run each problem 10 times.

Denote by

1. t the iteration number
2. n the number of nodes of the graph
3. $\boldsymbol{P}_0^{(o)}$ the initial matrix associated with the original approach.
4. $\rho_{0,v}^{(o)}$ the proportion of valid trajectories for $t = 0$, while using a pilot run with $f(\boldsymbol{x}, \boldsymbol{P}_0^{(o)})$
5. $|\mathcal{X}^-|$ and $\widehat{|\mathcal{X}^-|}$ the optimal and the average estimate respectively
6. $\bar{\varepsilon}$ the average relative experimental error based on 10 replications
7. ε_* and ε^* the smallest and the largest relative errors
8. κ the sample squared coefficient of variation of the estimate based on 10 runs.

Note that

$$\widehat{|\mathcal{X}^-|} = \frac{1}{10} \sum_{i-1}^{10} \widehat{|\mathcal{X}_i^-|},$$

where $\widehat{|\mathcal{X}_i^-|}$, $i = 1, \ldots, 10$, is the estimate of $|\mathcal{X}^-|$ obtained at i-th experiment.

Note that in our tables below $t = 0$ corresponds to sampling associated with the initial matrix $\boldsymbol{P}_0^{(o)}$ based on the original approach, (that is without CE updating). We shall call the estimator $\widehat{|\mathcal{X}_i^-|}$ in (22) for $t = 0$ based on the IS pdf $f(\boldsymbol{x}, \boldsymbol{P}_0^{(o)})$, the *IS estimator with the original approach* (ISO).

If not otherwise stated we shall often use instead of the updating rule (21) the following one

$$\widetilde{p}_{t,ij} = \frac{\sum_{k=1}^{N} I_{\{\boldsymbol{X}_k \in \mathcal{X}_{ij}\}} \, I_{\{S(\boldsymbol{X}_k) \geq \widehat{\gamma}_{t-1})\}} \; H(X_k, \widehat{p}_{t-1,ij})}{\sum_{k=1}^{N} I_{\{S(\boldsymbol{X}_k) \geq \widehat{\gamma}_{t-1})\}} H(X_k, \widehat{p}_{t-1,ij})}, \tag{40}$$

where $H(X_k, \widehat{p}_{t-1,ij}) = \frac{1}{p_{k,ij}}$ and as before, $p_{k,ij}$ denotes the transition probability from node i to node j using the trajectory generation Algorithm 12.2 at the k-th sample. We shall call such estimator, a *partial likelihood ratio estimator*, to distinquish it from the original (21), which contains the entire LR term W. Note that for an additive sample function (40) presents an unbiased estimate of (21). Note, however, that since S is only nearly additive, the approximation (40) still introduces a small bias, but at the same time reduces the variance as compared to the original estimator (21) in particular when n is large.

8.1 The Hamiltonian Cycle Problem

We present numerical results for the HC problem for a set of instances with the a randomly generated incidence matrix A. If not stated otherwise we use the original approach. Define

$$\eta = \frac{|E|}{n(n-1)}$$

as the *density* (sparsity) parameter of the graph $G = (V, E)$, where $|E|$ and $n(n-1)$ are the number of edges in the original graph $G = (V, E)$ and the complete one, respectively.

Table 3 summarizes the results of 10 runs with Algorithm 3.1 using the original approach for a graph with $n = 30$ nodes, density $\eta = 0.5$ and sample size $N = 10n^2 = 9,000$. For this particular instance we found that the proportion of valid trajectories $\rho_{0,v}^{(o)}$ (under $f(\boldsymbol{x}, \boldsymbol{P}_0^{(o)})$) is about 0.16, and for $t \geq 1$ (with CE), this proportion, denoted as $\rho_{t,v}$ is about 0.19. Since $\rho_{0,v}^{(o)} > \rho$ (recall that ρ is typically set to 0.001) the entire set of elite sample (samples with $S(\boldsymbol{X}_k) \geq n$) consist of *valid trajectories alone*. This implies that with the original approach and $\alpha = 1$ it is sufficient to use *only a single CE iteration*. The latter is equivalent to applying the classic *maximum likelihood* (ML) method, for which statistical inference (confidence intervals, etc.) is available. Note also that since all the elements at each row of $\boldsymbol{P}_0^{(o)}$ are equal, trajectory generation from such IS density $f(\boldsymbol{x}, \boldsymbol{P}_0^{(o)})$ is very fast. As result, the CPU time of the first CE iteration with the Algorithm 3.1 is typically much less than that of the following ones ($t \geq 2$). We nevertheless present 4 CE iterations to indicate that the CE method stabilizes after the first iteration.

Table 3: Performance of Algorithm 3.1 for $n = 30$, $\eta = 0.5$ and $N = 10n^2 = 9,000$

t		0	1	2	3	4
$\widehat{\lvert\mathcal{X}^-\rvert}$	Average	6.39E+21	6.39E+21	6.33E+21	6.39E+21	6.33E+21
	Min	5.66E+21	6.18E+21	5.73E+21	5.96E+21	5.85E+21
	Max	7.32E+21	6.65E+21	6.78E+21	6.90E+21	6.83E+21
ε	$\bar{\varepsilon}$	0.055	0.021	0.041	0.033	0.046
	ε_*	0.012	0.002	0.004	0.004	0.006
	ε^*	0.145	0.040	0.094	0.079	0.078
κ		0.0766	0.0457	0.0533	0.0425	0.0545

It follows from Table 3 that in this case the the ISO estimator is approximatelly two times less accurate (in terms of κ) than the single iterative CE one, (compare, $\kappa_0 = 0.0766$ with $\kappa_1 = 0.0457$ obtained after the first iteration).

Tables 4 and 5 present data similar to Table 3 for $n = 100$ with $\eta = 0.3$ and $\eta = 0.4$, respectively. In both cases we set the sample size $N = 5n^2 = 50,000$. For these instances we obtained $\rho_{0,v} \approx 0.012$, $\rho_{t,v} \approx 0.015$ and $\rho_{0,v} \approx 0.057$, $\rho_{t,v} \approx 0.061$, respectively. It is readily seen that that ISO performs at least as good as CE and both $\rho_{0,v}$ and $\rho_{t,v}$ increase in η. The execution time was about 30 seconds per iteration in both cases.

Table 4: Performance of Algorithm 3.1 with the original approach for $n = 100$, $\eta = 0.3$ and $N = 5n^2 = 50,000$

t		0	1	2	3	4
$\widehat{\lvert\mathcal{X}^-\rvert}$	Average	1.44E+103	1.31E+103	1.99E+103	1.18E+103	1.10E+103
	Min	9.20E+102	6.87E+102	6.39E+102	7.52E+102	7.48E+102
	Max	1.89E+103	2.34E+103	6.83E+103	2.72E+103	1.56E+103
ε	$\bar{\varepsilon}$	0.138	0.318	0.542	0.347	0.238
	ε_*	0.008	0.013	0.033	0.090	0.105
	ε^*	0.359	0.783	2.434	1.310	0.419
κ		0.184	0.391	0.662	0.508	0.278

Table 5: Performance of Algorithm 3.1 with the original approach for $n = 100$, $\eta = 0.4$ and $N = 5n^2 = 50,000$

t		0	1	2	3	4
$\widehat{\lvert\mathcal{X}^-\rvert}$	Average	7.42E+115	8.00E+115	7.12E+115	7.48E+115	7.84E+115
	Min	6.14E+115	6.79E+115	6.55E+115	6.79E+115	6.76E+115
	Max	8.21E+115	8.97E+115	7.95E+115	8.12E+115	8.70E+115
ε	$\bar{\varepsilon}$	0.056	0.074	0.053	0.048	0.058
	ε_*	0.005	0.001	0.007	0.003	0.005
	ε^*	0.173	0.151	0.116	0.092	0.137
κ		0.077	0.089	0.068	0.059	0.078

One can see that the SCV κ decreases in η (compare $\kappa = 0.278$ with $\kappa = 0.078$ for $t = 4$, respectively).

Our extensive empirical practice with not too sparse (randomly generated) matrices A of moderate size, say $n \leq 200$, indicate that typically $\rho_{0,v} > \rho = 0.001$, that is the proportion of valid trajectories for $t = 0$ is already greater than ρ. That in turn implies that for all such graphs there is no need to introduce the parameter ρ at all. Moreover, the ISO ($t = 0$) typically performs as well CE ($t \geq 1$), and if CE is used, than a single iteration with Algorithm 3.1 is sufficient.

We next explain why for not too sparse matrices A we have that (**a**) $\rho_{0,v}^{(o)} > \rho$ and (**b**) ISO typically performs as well CE.

(**a**) Denote by $\mathcal{X}^{(o)}$ the set of all sample trajectories (valid and invalid) generated by Algorithm 12.2 using the ariginal approach. Clearly $\mathcal{X}^{(o)}$ contains the optimal set $\mathcal{X}^-$, since using the auxiliary approach no single valid trajectory will be able to pass through any transition associated with zero elements of matrix A. Define next

$$\ell^{(o)} = \frac{|\mathcal{X}^-|}{|\mathcal{X}^{(o)}|}.$$

We argue (and our experiments justify it) that the cardinality of the (reduced) space $\mathcal{X}^{(o)}$, based on the initial matrix $\boldsymbol{P}_0^{(o)}$ is much smaller than that of the entire one $\mathcal{X}$ based on $\boldsymbol{P}_{\mathsf{U}}$. In addition, if and if A is not too sparse, say $0.3 < \eta \leq 1$, then $\ell^{(o)}$ is typically not a rare event (say $\ell^{(o)} > 10^{-4}$ as compared to $\rho = 0.001$). The above should be compared with

$$\ell = \frac{|\mathcal{X}^-|}{|\mathcal{X}|},$$

which is typically a rare event, ($\ell < 10^{-4}$ as compared to $\rho = 0.001$) even for large η (dence A), say for $0.9 < \eta < 1$, provided n is not too small, say $n \geq 10$.

(**b**) To explain why ISO is typically as good as CE for moderate graphs with not too sparse matrices ($0.3 < \eta \leq 1$), recall first that for a complete graph ($\eta = 1$) ISO produces an estimator with *zero variance*. Take next into account that according to our (yet empirical) studies with the original approach we have $\rho_{0,v} > \rho$ and $\rho_{0,v} \approx \rho_{t,v},\ t \geq 1$. The former inequality means that all elite sample trajectories are valid, while the latter means that there is, in fact, *no speed up by using CE as compared to ISO*. Taking finally into account that the likelihood ratio terms in CE introduces some additional noise as compared to ISO, (which contains no likelihoods), the results becomes evident.

We next present numerical results for sparse matrices A, say where $\rho_{0,v}^{(o)} < \rho = 0.001$. It is important to note that in order to get accurate estimates with sparse matrices A the sample size N should be increased as the sparsity increases (see Tables 4 and 5). Also, as we shall see below, for such cases we shall need to resort to multiple iterations, while using both the original and the auxiliary approach.

We consider first a randomly generated instance with $n = 30$ nodes and density $\eta = 0.2$. For this instance we found empirically (by taking a pilot run under $f(\boldsymbol{x}, \boldsymbol{P}_0^{(o)})$) that $\rho_{0,v}^{(o)} = 0.002$. We employ here both (i) the original and (ii) the auxiliary approaches. We found that in both cases that $\rho_{1,v} \approx 0.005$ for $t \geq 1$.

(i) Our results for that case are summarized in Table 6, which is similar to Table 3. Because of the sparsity we used a sample $N = 40n^2 = 36,000$ instead of $N = 10n^2 = 9,000$ as before. It follows that Algorithm 3.1 stabilizes after the first CE iteration meaning that for $t > 1$ the proportion of valid trajectories $\rho_{t,v} \approx \rho_{1,v} \approx 0.005$ and that CE (see the results for $t \geq 1$) outperforms ISO about two times (see the results for $t = 0$).

Table 6: Performance of Algorithm 3.1 with the original approach for $n = 30$, $\eta = 0.2$, $\rho = 0.001$ and $N = 40n^2 = 36,000$

t		0	1	2	3	4
$\widehat{\lvert\mathcal{X}^-\rvert}$	Average	3.29E+09	4.15E+09	3.91E+09	3.87E+09	3.85E+09
	Min	2.33E+09	2.79E+09	2.73E+09	2.79E+09	2.85E+09
	Max	5.06E+09	5.25E+09	4.94E+09	5.02E+09	4.98E+09
ε	$\bar{\varepsilon}$	0.230	0.144	0.150	0.141	0.116
	ε_*	0.105	0.035	0.020	0.015	0.021
	ε^*	0.536	0.327	0.303	0.298	0.295
κ		0.272	0.187	0.189	0.181	0.156

(ii) For the auxiliary approach we consider a *non-uniform* initial probability matrix $\boldsymbol{P}_0$. More specifically, let $p_{0,ij}$ and $p_{0,ik}$ denote the initial probability associated with the corresponding unity and zero elements of the matrix A, respectively. Denote

$$q_i = \frac{p_{0,ik}}{p_{0,ij}}. \tag{41}$$

Note that for $q_i = 1$ we obtain the auxiliary approach with the uniform matrix $\boldsymbol{P}_0$ and for $q_i = 0$ we obtain the original approach with $\boldsymbol{P}_0^{(o)}$.

The reason for using the auxiliary approach with a small q_i for several initial iterations is to ensure smooth updating of the elements of the matrix $\boldsymbol{P}$ and smooth convergence to $\gamma = n$. As soon as this is achieved one can switch back to the original less noisy approach.

Table 7 presents data similar to that in Table 6 with $q_i = 0.001,\ i = 1, \ldots, 30$ instead of $q_i = 0$, for $t = 4$ and with $q_i = 0$ for $t = 5$. As one can see from Table 7, by switching back to the original approach after the fourth iteration we obtained a more accurate final estimate. (Compare, for example, $\kappa_5 = 0.147$ obtained after the fifth iteration by switching to the original approach with $\kappa_4 = 0.258$ obtained after the fourth iteration using the auxiliary one). Note that switching implies re-normalization of the elements $p_{t=4,ij}$ corresponding to unity elements of the matrix A.

Table 7: Performance of Algorithm 3.1 with the auxiliary approach with $q_i = 0.001$ for $n = 30$, $\eta = 0.2$, $\rho = 0.001$ and $N = 40n^2 = 36,000$.

t		0	1	2	3	4	5
$\widehat{\lvert\mathcal{X}^-\rvert}$	Average	3.92E+09	3.88E+09	4.41E+09	3.89E+09	4.70E+09	3.90E+09
	Min	1.64E+09	2.58E+09	3.13E+09	2.49E+09	2.93E+09	3.10E+09
	Max	8.32E+09	6.25E+09	7.17E+09	4.80E+09	6.43E+09	4.64E+09
ε	$\bar{\varepsilon}$	0.387	0.205	0.206	0.139	0.206	0.123
	ε_*	0.092	0.023	0.008	0.018	0.018	0.011
	ε^*	1.120	0.612	0.625	0.360	0.377	0.205
κ		0.503	0.282	0.281	0.180	0.258	0.147

It follows from Tables 6 and 7 that both the original and the auxiliary approach (with $q_i = 0.001$) perform similarly.

For quite sparse matrices, and in particular, where $\rho_{0,v}$ is much less than ρ, say where $\rho_{0,v} < 0.001$, we used the auxiliary approach with small q. As an example consider an instance with $n = 30$ and $\eta = 0.15$. By making a pilot run with $N = 10,000$ under the original approach with $\boldsymbol{P}_0^{(o)}$ we could not allocate any valid HC.

Table 8 presents data similar to Table 7 for $n = 30$, $\eta = 0.15$, $\rho = 0.001$ and $N = 100,000$ at each iteration. We obtained $\rho_{0,v} \approx 0.0001$, $\rho_{t,v} \approx 0.0042$ (for $t \geq 4$) and the execution time was about 18 seconds per iteration. After the fifth iteration we switched back to the original approach. As one can see from Table 8 that one needs (similar to Table 7) several iteration of CE in order to stabilize and eventually to outperform the ISO. But even with $N = 100,000$ the relative error κ of CE (for $t \geq 1$) still fluctuates from iteration to iteration within 20-30%.

Table 8: Performance of Algorithm 3.1 with the auxiliary approach $n = 30$, $\eta = 0.15$ and $N = 100,000$

t		0	1	2	3	4	5	6
$\widehat{\lvert\mathcal{X}^-\rvert}$	Average	8.16E+04	7.28E+04	6.11E+04	7.79E+04	6.62E+04	6.50E+04	6.56E+04
	Min	1.25E+04	4.85E+04	3.94E+04	5.17E+04	5.40E+04	4.53E+04	4.47E+04
	Max	1.90E+05	1.53E+05	7.83E+04	3.56E+05	1.04E+05	8.81E+04	8.89E+04
ε	$\bar{\varepsilon}$	0.535	0.237	0.132	0.571	0.122	0.144	0.210
	ε_*	0.007	0.010	0.006	0.053	0.009	0.028	0.029
	ε^*	1.324	1.106	0.354	1.630	0.567	0.356	0.355
κ		0.677	0.405	0.183	0.551	0.211	0.188	0.254

Next we present simulation results for small size sparse matrices A, for which the number of HC's is available via full enumeration.

In our first example we randomly generated an instance with $n = 10$ and $\eta = 0.28$. Using complete enumeration we found that the cordnality $|\mathcal{X}^-| = 3$ and the tours are:

$$1 \to 8 \to 9 \to 2 \to 4 \to 3 \to 7 \to 5 \to 10 \to 6 \to 1$$
$$1 \to 8 \to 10 \to 7 \to 5 \to 4 \to 3 \to 2 \to 9 \to 6 \to 1$$
$$1 \to 8 \to 10 \to 7 \to 5 \to 2 \to 4 \to 3 \to 9 \to 6 \to 1.$$

Figure 3 shows the matrix A and the evolution of $\widehat{\boldsymbol{P}}_t,\ t = 1, 2, 3$ using the CE Algorithm 3.1 with the original approach. Figure 4 presents the corresponding pictural evoluation of $\widehat{\boldsymbol{P}}_t,\ t = 1, 2, 3$. We set $N = 1000$, $\alpha = 1$ (no smoothing), and use the original updating rule (21) rather than the "truncated" one (40), which was used so far. The reason for using (21) is to insure robustness and fast convergence to the correct answer. Applying Algorithm 3.1 we found that $\rho_{0,v} = 0.18$. In Figures 3 and 4, different colors are used for elements corresponding to transitions belonging to different HC's. Table 9 specifies which color represents which of the three HC's.

Table 9: Legend for Figures 3 and 4

Color	Used in HC numbers
Red	1
Green	2
Blue	3
Purple	1, 3
Cyan	2, 3
White	1, 2, 3
Black	—

It follows that

1. After one iteration (see $\boldsymbol{P}_1$) all useless edges, i.e edges which no valid trajectory passes through, were successfully identified and thus assigned with probability 0. They are colored in black.

2. All edges common to all three valid trajectories were also found after one iteration. These edges are white colored. Since all HC's pass through these edges, a trajectory reaching a node connected to them during its generation has to choose them. Therefore these edges should have probability 1. These edges indeed have probability 1 in $\boldsymbol{P}_1$ and, obviously, in $\boldsymbol{P}_2$ and $\boldsymbol{P}_3$ as well.

3. After two more CE iterations the process converged to the CE optimal matrix (recall (23) and (24)). In $\boldsymbol{P}_3$ all edges belonging to a single HC (colored red, green and blue) have a probability of approximately $\frac{1}{3}$, and all those used by two HCs (colored purple and cyan) have a probability of approximately $\frac{2}{3}$.

A =

0	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0	1	0
1	0	1	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	1
1	0	0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1
0	1	0	0	0	1	0	0	0	1
0	0	0	0	0	1	1	0	0	0

P_0 =

0	0	0	0	0	0	0	1	0	0
0	0	0	0.5	0	0	0	0	0.5	0
0	0.3333	0	0	0	0	0.3333	0	0.3333	0
0.3333	0	0.3333	0	0	0	0	0	0.3333	0
0	0.3333	0	0.3333	0	0	0	0	0	0.3333
0.3333	0	0	0.3333	0	0	0	0.3333	0	0
0	0.3333	0	0.3333	0.3333	0	0	0	0	0
0	0	0	0	0	0	0	0	0.5	0.5
0	0.3333	0	0	0	0.3333	0	0	0	0.3333
0	0	0	0	0	0.5	0.5	0	0	0

P_1 =

0	0	0	0	0	0	0	1	0	0
0	0	0	0.7128	0	0	0	0	0.2872	0
0	0.2872	0	0	0	0	0.4256	0	0.2872	0
0	0	1	0	0	0	0	0	0	0
0	0.2872	0	0.2872	0	0	0	0	0	0.4256
1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0.4256	0.5744
0	0.4256	0	0	0	0.5744	0	0	0	0
0	0	0	0	0	0.4256	0.5744	0	0	0

P_2 =

0	0	0	0	0	0	0	1	0	0
0	0	0	0.6001	0	0	0	0	0.3999	0
0	0.3999	0	0	0	0	0.3168	0	0.2833	0
0	0	1	0	0	0	0	0	0	0
0	0.2833	0	0.3999	0	0	0	0	0	0.3168
1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0.3168	0.6832
0	0.3168	0	0	0	0.6832	0	0	0	0
0	0	0	0	0	0.3168	0.6832	0	0	0

P_3 =

0	0	0	0	0	0	0	1	0	0
0	0	0	0.6552	0	0	0	0	0.3448	0
0	0.3448	0	0	0	0	0.3417	0	0.3135	0
0	0	1	0	0	0	0	0	0	0
0	0.3135	0	0.3448	0	0	0	0	0	0.3417
1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0.3417	0.6583
0	0.3417	0	0	0	0.6583	0	0	0	0
0	0	0	0	0	0.3417	0.6583	0	0	0

Figure 3: The adjacency matrix, A, and dynamics of $\mathbf{P}_t$ for a 10 node instance with $|\mathcal{X}^-| = 3$.

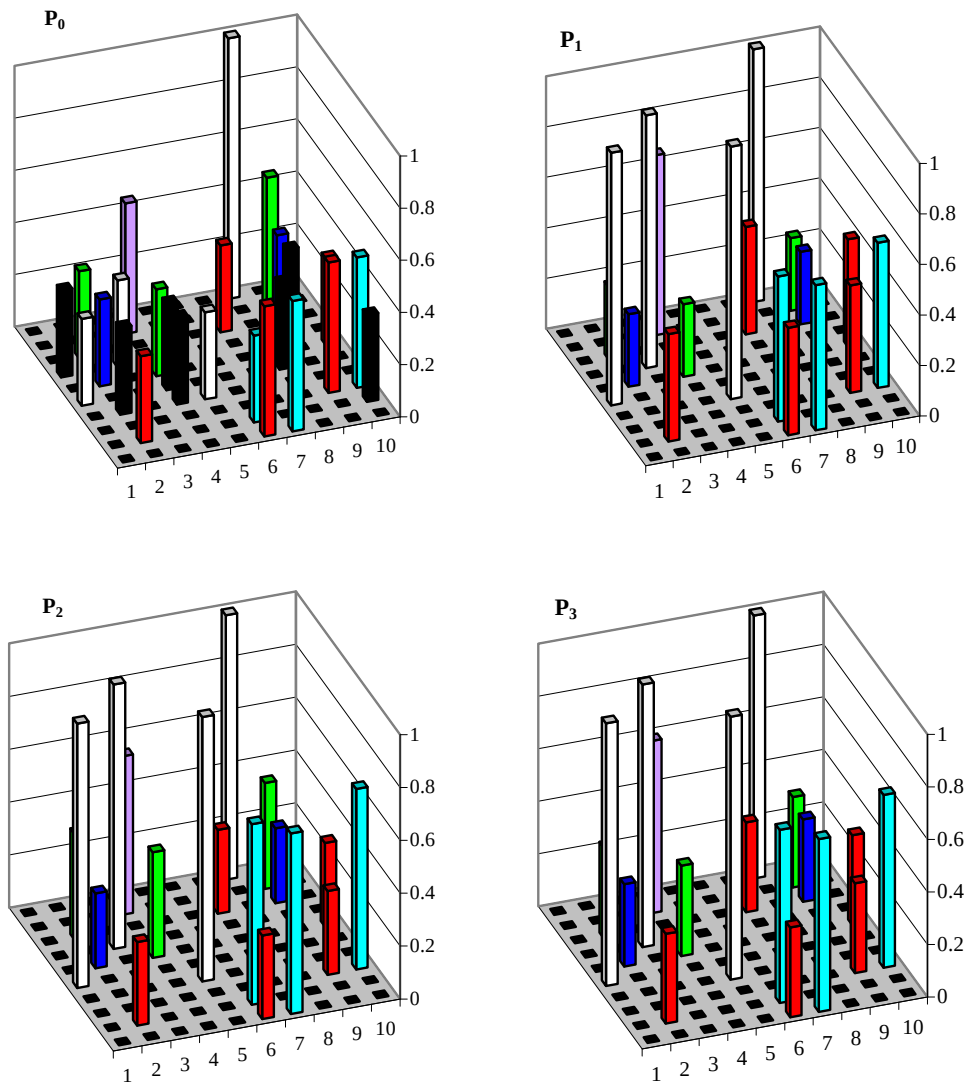

Figure 4: Graphic illustration of the dynamics of $\mathbf{P}_t$ for a 10 node instance with $|\mathcal{X}^-| = 3$.

Our second example is for a matrix A of size $n = 8$ with $\eta \approx 0.4$. Figure 5 is similar to Figure 4 but with $|\mathcal{X}^-| = 2$ instead of $|\mathcal{X}^-| = 3$. It graphically presents the dynamics of $\widehat{\boldsymbol{P}}_t,\ t = 0, 1, 2, 3$. The two HC's are

$$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 1$$
$$1 \to 8 \to 7 \to 6 \to 3 \to 4 \to 5 \to 2 \to 1.$$

Here we set $N = 10n^2 = 640$, $\alpha = 1$ and again use the original approach and updating rule (21). Applying Algorithm 3.1 we found that $\rho_{0,v} \approx 0.13$. As before, we used different colors for bars corresponding to edges belonging to different HC's. Red was used for edges belonging to the first HC, green — for those belonging to the second and yellow — for those common to both HCs. Black represents the redundant edges.

It follows from Figure 5, that after one iteration, the probability matrix, $\widehat{\boldsymbol{P}}_1$, is approximately CE optimal, since all edges used in both HC's (colored red and green) were assigned a probability of approximately 0.5 and all edges common to both cycles (colored yellow) were assigned probability 1. In addition, one can also

see that $\widehat{\boldsymbol{P}}_3 \approx \widehat{\boldsymbol{P}}_2 \approx \widehat{\boldsymbol{P}}_1$, which means the algorithm converged, since $\widehat{\boldsymbol{P}}_2$ and $\widehat{\boldsymbol{P}}_3$ are very close to $\widehat{\boldsymbol{P}}_1$.

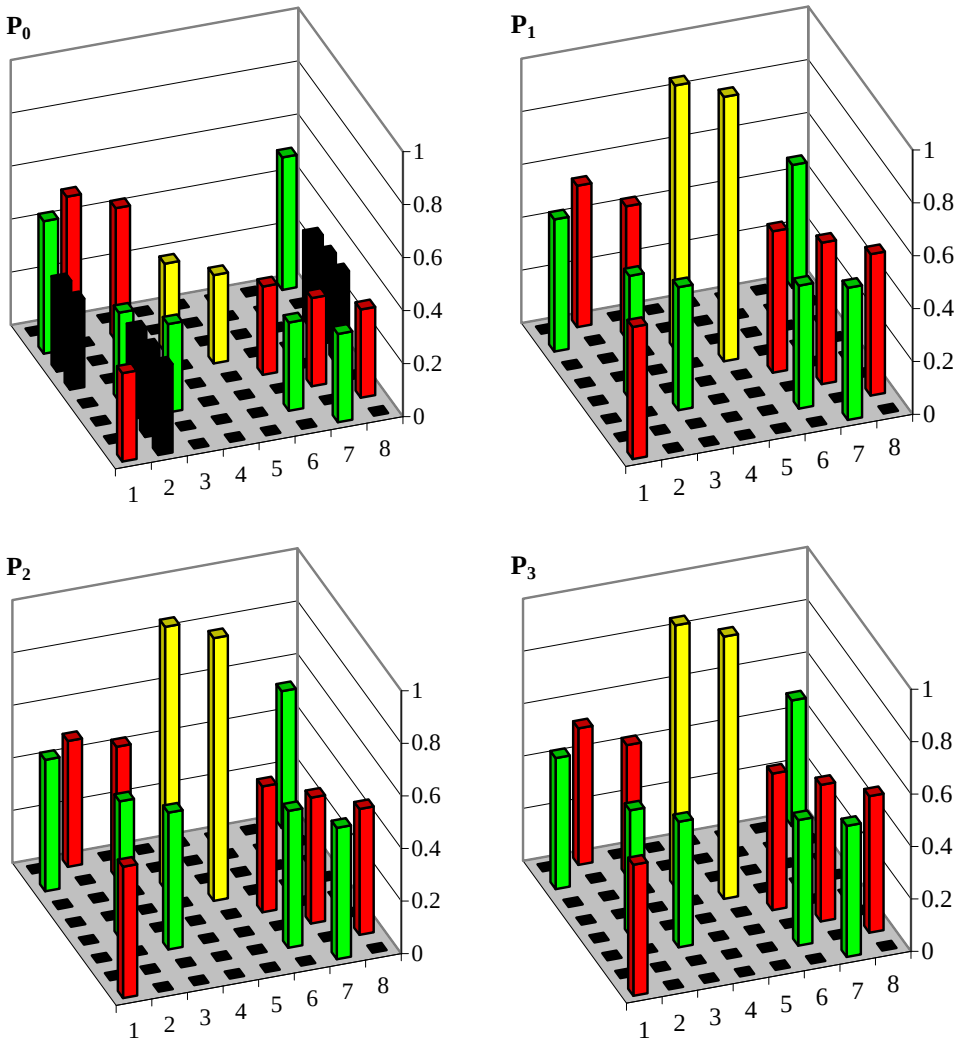

Figure 5: Graphic illustration of the dynamics of $\mathbf{P}_t$ for an 8 node instance with $|\mathcal{X}^-| = 2$.

Our third example is for a matrix A of size $n = 8$ and density $\eta = 0.4$, which contains only a single valid trajectory. The HC is

$$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 1.$$

Figure 6 shows A, $\widehat{\boldsymbol{P}}_0$ and $\widehat{\boldsymbol{P}}_1$. Figure 7 graphically presents $\widehat{\boldsymbol{P}}_0$ and $\widehat{\boldsymbol{P}}_1$. Again we set $N = 10n^2 = 640$, $\alpha = 1$ and used the original approach and updating rule (21). Applying Algorithm 3.1 we found that $\rho_{0,v} \approx 0.13 > \rho = 0.01$. In figures 6 and 7 edges colored white comprise the HC, and edges colored black are redundant edges.

Figures 6 and 7 show that after one iteration, the probability matrix, $\widehat{\boldsymbol{P}}_1$, is exactly CE optimal, since all edges used in the HC were assigned a probability of 1, and all redundant edges were assigned probability 0.

We summarize our results by presenting the following

Algorithm 8.1

$A =$

0	**1**	0	0	0	0	0	**1**
1	0	**1**	0	0	0	0	0
1	0	0	**1**	0	0	0	**1**
1	0	0	0	**1**	0	0	**1**
0	0	0	**1**	0	**1**	0	**1**
0	**1**	**1**	0	0	0	**1**	0
0	**1**	0	0	0	**1**	0	**1**
1	**1**	0	0	0	0	**1**	0

$P_0 =$

0	**0.5**	0	0	0	0	0	**0.5**
0.5	0	**0.5**	0	0	0	0	0
0.3333	0	0	**0.3333**	0	0	0	**0.3333**
0.3333	0	0	0	**0.3333**	0	0	**0.3333**
0	0	0	**0.3333**	0	**0.3333**	0	**0.3333**
0	**0.3333**	**0.3333**	0	0	0	**0.3333**	0
0	**0.3333**	0	0	0	**0.3333**	0	**0.3333**
0.3333	**0.3333**	0	0	0	0	**0.3333**	0

$P_1 =$

0	**1**	0	0	0	0	0	**0**
0	0	**1**	0	0	0	0	0
0	0	0	**1**	0	0	0	**0**
0	0	0	0	**1**	0	0	**0**
0	0	0	**0**	0	**1**	0	**0**
0	**0**	**0**	0	0	0	**1**	0
0	**0**	0	0	0	**0**	0	**1**
1	**0**	0	0	0	0	**0**	0

Figure 6: The adjacency matrix, A, and dynamics of $\boldsymbol{P}_t$ for an 8 node instance with $|\mathcal{X}^-| = 1$.

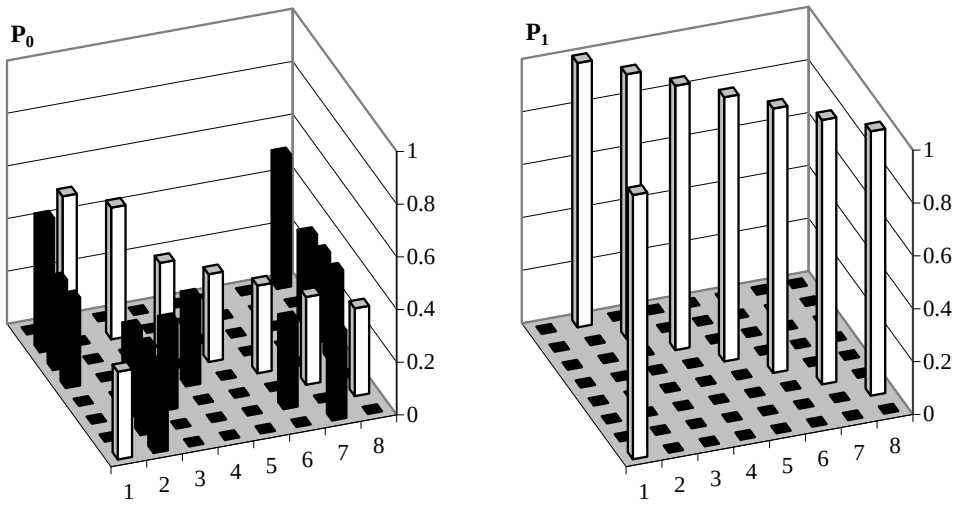

Figure 7: Graphic illustration of the dynamics of $\boldsymbol{P}_t$ for an 8 node instance with $|\mathcal{X}^-| = 1$.

1. Run a pilot sample with the original approach and see whether or not under $\boldsymbol{P}_0^{(o)}$ the proportion of valid trajectories $\rho_{0,v} > \rho = 0.001$.

2. If $\rho_{0,v} > \rho$ (matrix A is dense) proceed with the original approach, use a single iteration with Algorithm 3.1, estimate $\widehat{|\mathcal{X}^-|}$ and stop.

3. If $\rho_{0,v} < \rho$ (matrix A is sparse) proceed with the auxiliary approach; chose a small q, say $q = 0.001$ and perform 3-4 iterations with Algorithm 3.1. Switch back to the original approach, that is nullify all positive elements of $\boldsymbol{P}_t$ wich are associated with zero elements of A, perform one more iteration with Algorithm 3.1, estimate $\widehat{|\mathcal{X}^-|}$ and stop. Note that if no valid trajectories are found Algorithm 3.1 delivers $\widehat{|\mathcal{X}^-|} = 0$.

8.2 Permanent

We present here simulations studies with Algorithm 3.1 for the permanent problem. In particular, we consider 3 data sets with the parameters n, N and η identical to these for HC. As before, if not stated otherwise we shall use $\rho = 0.001$ and $\alpha = 0.7$. As we shall see below, the results obtained for the permanent are quite similar to these for HC's.

Table 10 presents data similar to Table 5 for $n = 100$ with $\eta = 0.4$ and $N = 5n^2 = 50,000$ with the original approach.

For these instances we obtained $\rho_{0,v} \approx 0.012$, and $\rho_{t,v} \approx 0.015$. It is readily seen that again ISO performs at least as good as CE. The execution time was about 20 seconds per iteration.

Table 10: Performance of Algorithm 3.1 with the original approach for $n = 100$ with $\eta = 0.4$, $N = 5n2 = 50,000$ and $\rho = 0.001$.

t		0	1	2	3	4
$\widehat{\lvert\mathcal{X}^-\rvert}$	Average	6.77E+117	6.42E+117	6.52E+117	6.86E+117	6.45E+117
	Min	6.26E+117	5.71E+117	5.63E+117	5.96E+117	5.95E+117
	Max	7.96E+117	7.93E+117	7.36E+117	8.05E+117	7.66E+117
ε	$\bar{\varepsilon}$	0.0576	0.0731	0.0549	0.0806	0.0885
	ε_*	0.0217	0.0029	0.0072	0.001	0.0027
	ε^*	0.1459	0.2356	0.1367	0.1739	0.1315
κ		0.0879	0.1051	0.0748	0.0998	0.0757

Table 11 presents data similar to Table 8 using the auxiliary approach with $q = 0.001$. Similar to Table 8 we used $n = 30$, $\eta = 0.15$ and $\rho = 0.001$. We took a sample $N = 300,000$ while updating both, $\boldsymbol{P}$ and $|\mathcal{X}^-|$. For these instance we obtained $\rho_{0,v} \approx 0.00016$, and $\rho_{t,v} \approx 0.0178$. The execution time was about 65 seconds per iteration.

Table 11: Performance of Algorithm 3.1 with the auxiliary approach $n = 30$, $\eta = 0.15$, $N = 300,000$, $\rho = 0.001$ and $q = 0.001$.

t		0	1	2	3	4
$\widehat{\lvert\mathcal{X}^-\rvert}$	Average	2.3909e+006	2.4536e+006	2.2712e+006	2.1168e+006	2.2991e+006
	Min	1.9019e+006	2.1828e+006	1.8273e+006	1.5113e+006	1.5807e+006
	Max	3.2895e+006	3.0817e+006	3.1535e+006	2.7940e+006	2.8121e+006
ε	$\bar{\varepsilon}$	0.1433	0.0844	0.1532	0.1582	0.1869
	ε_*	0.0200	0.0013	0.0254	0.0305	0.0319
	ε^*	0.3758	0.2560	0.3885	0.3199	0.3276
κ		0.1909	0.1187	0.1902	0.1944	0.1983

8.3 SAW

Before presenting numerical results on SAWs recall that

1. We use here the original approach and as for HC and the permanent we use the updating rule (40) rather than (21).

2. The initial probability matrix, $\boldsymbol{P}_0$, allows transitions only to neighboring grid points, and does so with equal probability. For example, in the row corresponding to $(0,0)$ all elements are zero, except for those corresponding to transitions to $(0,-1)$, $(0,1)$, $(-1,0)$ and $(1,0)$, which are 0.25 each.

3. The results presented here are based on the symmetry of the problem. For example, the following moves are symmetric:

$$\begin{array}{cccc} (2,1) \to (3,1) & (1,2) \to (1,3) & (-1,2) \to (-1,3) & (-2,1) \to (-3,1) \\ (-2,-1) \to (-3,-1) & (-1,-2) \to (-1,-3) & (1,-2) \to (1,-3) & (2,-1) \to (3,-1). \end{array}$$

   In general, every move has seven symmetrical moves. Therefore, when updating the pdf $f(\boldsymbol{x}, \widehat{\boldsymbol{P}}_{t-1})$ in iteration t, we treated all symmetric moves as if they were the same move, thus assigning them equal probabilities.

We shall show next, that in order to estimate quite accurately the number of SAWs for moderate n, say $n \leq 200$, there is no need for Algorithm 3.1 at all, since one can estimate $|\mathcal{X}|$ directly using the initial uniform (to the 4 neighboring states) pdf $f(\boldsymbol{x}, \boldsymbol{P}_0)$, thus, using again in analogy to HC's problem the ISO approach. This is the same as stating that one can go directly to step 6 of Algorithm 3.1 with the uniform IS pdf $f(\boldsymbol{x}, \boldsymbol{P}_0)$ (thereby avoiding its main steps 1-5). All this is based on our empirical observation that for moderate n, a substantial proportion of the trajectories $\rho_{0,v}$ generated from the uniform pdf $f(\boldsymbol{x}, \boldsymbol{P}_0)$ is valid, i.e., they are SAWs of length n.

Table 12 shows the empirical proportion $\rho_{0,v}$ of valid SAWs as function of n based on a sample $N = 100,000$.

Table 12: The empirical proportion of valid SAWs as a function of n based on a sample $N = 100,000$.

n	$\rho_{0,v}$
10	0.99
20	0.91
50	0.58
100	0.225
200	0.025
500	0.00003

Similar to HC, we shall call such estimator $\widehat{|\mathcal{X}^-|}$ generated by $f(\boldsymbol{x}, \boldsymbol{P}_0)$, the ISO estimator of $|\mathcal{X}^-|$. Note that because of the symmetry calculating the number of SAW's resembles, in some sense, calculating the number of HC's an a complete graph. But recall that CMC is exact (produces an estimator with zero variance), while calculating the number of HC's an a complete graph. So, one should not be surprised by such nice performance of ISO for SAW's. We argue that in analogy to HC, by increaing the problem size,or by making SAW's less sparse and not symmetric, by forbidding some if its movements, (say in analogy to HC, one of the four movements randomly), ISO will not necessarely outperform its CE counterpart.

Note that $\rho_{0,v}$ for a SAW differs from $\rho_{0,v}^{(o)}$ for a HC, namely, in the former we generate trajectories for $t = 0$ using the *uniform* pdf $f(\boldsymbol{x}, \boldsymbol{P}_0)$, while in the latter - the *non uniform* pdf $f(\boldsymbol{x}, \boldsymbol{P}_0^{(o)})$.

Note also that since for finite n the random variable $H(\boldsymbol{X}) = \frac{1}{f(\boldsymbol{x}, \boldsymbol{P}_0)}$, has a finite variance, one can use for ISO directly the central limit theorem and obtain a valid confidence interval for the unknown parameter $|\mathcal{X}^-|$. The reader should still keep in mind that the underlying counting problem is #P-complete.

Although, as we shall see below, our numerical results clearly indicate that $f(\boldsymbol{x}, \boldsymbol{P}_0)$ is a good choice for $n \leq 200$, we nevertheless compare the ISO with the one based on the four-iterative ($t = 4$) version of Algorithm 3.1. Note that for $t = 1$ one can obtain (similar to HC) valid confidence regions for the tuple $\{|\mathcal{X}^-|, \boldsymbol{P}^*\}$ (see, for example, Theorem 2.4.1 of [14]).

To study the accuracy of Algorithm 3.1 we run it ten times for different values of n. We start with $n = 27$ for which the cardinality $|\mathcal{X}^-|$ is available. The number is $|\mathcal{X}^-| = 881,317,491,628$.

Table 13 presents $\widehat{|\mathcal{X}^-|}$, $\bar{\varepsilon}$, ε_* ε^* and the CPU time for the case $n = 27$ for various N and $t = 0$. Table 14 presents data similar to Table 13, but for $t = 4$. Here, as before $\widehat{|\mathcal{X}^-|}$ and $\bar{\varepsilon}$ denote the average estimate and the average relative experimental error, both based on 10 independent replications, ε_* and ε^* denote the smallest and the largest relative error and CPU denotes the average CPU time in seconds.

Table 13: Performance of ISO for the SAW problem with $n = 27$ with different sample sizes and $t = 0$.

N	$\widehat{\lvert\mathcal{X}^-\rvert}$	$\bar{\varepsilon}$	ε^*	ε_*	CPU
30,000	882696448700	0.0051	0.0151	0.0001	0.5
60,000	880555570421	0.0043	0.0099	0.0002	1.1
90,000	882408422067	0.0038	0.0087	0.0006	1.6

Table 14: Performance of Algorithm 3.1 for the SAW problem with $n = 27$, with different sample sizes and $t = 4$.

N	$\widehat{\lvert\mathcal{X}^-\rvert}$	$\bar{\varepsilon}$	ε^*	ε_*	CPU
30,000	875786840759	0.0130	0.0287	0.0012	3
60,000	880525655059	0.0043	0.0100	0.0010	7
90,000	877368544116	0.0067	0.0212	0.0010	10

It is readily seen that the ISO outperforms the multi-iterative one CE in both accuracy and CPU time. We found that this is typically true for moderate n and any $t \geq 1$. Our explanation for this phenomenon is as follows: the initial uniform pdf $f(\boldsymbol{x}, \boldsymbol{P}_0)$ is a "good" one. Apparently, for moderate values of n it is "close" to the optimal IS parametric pdf $f(\boldsymbol{x}, \boldsymbol{P}^*)$. Trying to improve the performance of Algorithm 3.1 by updating the parameters $\widetilde{p}_{t,ij}$ using CE one runs (similar to HC) into the instability phenomenon caused by high dimensional likelihood ratio terms (W or H) and as result one gains nothing. To reduce the instability one may apply some alternative method, like the gradient-based ones, or the methods based on the reconstruction of the optimal $\boldsymbol{P}^*$, similar to (38), which a free of the likelihood ratio terms.

Table 15 presents data similar to Table 13 for $n = 100$ with the ISO method. Recall that a typical SAW of length $n = 100$ generated by Algorithm 3.1 is shown in Figure 8. Also since for $n = 100$ the optimal value $|\mathcal{X}^-|$ is not available we replace it, as before, by the arithmetic average

$$\widehat{|\mathcal{X}^-|} = \frac{1}{10} \sum_{i-1}^{10} \widehat{|\mathcal{X}_i^-|}$$

where $\widehat{|\mathcal{X}_i^-|}$. $i = 1, \ldots, 10|$ is the outcome of the i-th run.

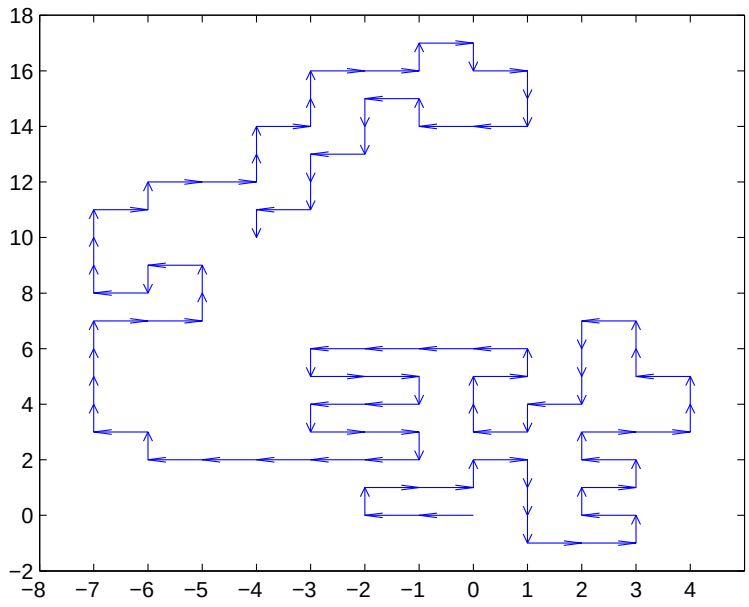

Figure 8: A typical SAW of length $n = 100$ generated by Algorithm 3.1

Table 15: Performance of the ISO method for the SAW problem with $n = 100$ with different sample sizes

N	$\widehat{\lvert\mathcal{X}^-\rvert}$	$\bar{\varepsilon}$	ε^*	ε_*	CPU
400,000	7.733883E+42	0.0390	0.0792	0.0094	15
600,000	7.721973E+42	0.0159	0.0461	0.0054	25
800,000	7.829240E+42	0.0100	0.0162	0.0007	35

As we can see from Table 15 that ISO performs quite stably and reliably.

We also ran Algorithm 3.1 with CE for different different iteration numbers and found again that the ISO is faster and at least as accurate as multi-iterative one. In particular, Table 16 shows a typical dynamics (the cardinality $\widehat{|\mathcal{X}^-|}_t$ and the proportion $\rho_{t,v}$ of valid SAWs trajectories as functions of t) of Algorithm 3.1 for $n = 100$ with $N = 400,000$, $\alpha = 0.7$ and $t = 9$. The CPU time for $t = 9$ was 14 minutes. One can readily seen that in analogy to HC for not too sparse matrices A we have that $\rho_{0,v} > \rho$ (compare $\rho_{0,v} = 0.221$ with $\rho = 0.001$) The rest of the results of Table 16 are self-explanatory.

Table 16: The dynamics of Algorithm 3.1 for $n = 100$ with $N = 400,000$, $\alpha = 0.7$ and $t = 9$.

t	$\widehat{\lvert\mathcal{X}_t^-\rvert}$	$\rho_{t,v}$
0	7.332509E+42	0.221
1	8.253465E+42	0.335
2	8.273701E+42	0.398
3	8.371508E+42	0.429
4	7.371052E+42	0.444
5	7.358388E+42	0.453
6	7.326165E+42	0.456
7	8.053898E+42	0.456
8	7.725057E+42	0.456
9	7.998223E+42	0.457

In summary, we obtain that for moderate problems ($n \leq 200$), the naive ISO outperforms CE. This is similar to HC problems, provided the matrix A in the latter is not too sparse, say $\eta \geq 0.3$. The investigation of the efficiency of CE and other methods for larger sizes of SAW's is underway.

8.4 CE for SAT

We shall present simulation results for the SAT problem in the SNF format. Our basic SAT model will be as follows. The availability (occurrence) of every $x_i,\ i = 1,\ldots,m$ at each clause $C_k,\ k = 1,\ldots,n$ will be chosen randomly, say from a Bernoulli pdf with the same parameter $p^{(1)}$. Furthermore, given that a particular $x_i,\ i = 1,\ldots,m$ has been selected, we flip another coin from a Bernoulli pdf with a fixed parameter $p^{(2)}$ to decide whether we have obtained for (each x_i) a literal ($q = x$) or its negotiation ($q = \bar{x}$). Note that for fixed m, n and $p^{(2)}$ the SAT number $|\mathcal{X}^-|$ heavily depends on the parameter value $p^{(1)}$. In our model we set

$n = 100$, $m = 25$ and $p^{(2)} = 0.5$. Clearly, $|\mathcal{X}^-|$, decreases in $p^{(1)}$. We shall call $p^{(1)}$, in analogy to η for HC, the density (sparsity) parameter of SAT.

Table 17 presents the performance of the CMC method based on 10 independent replications. One can clearly see that the efficiency of CMC decreases in $p^{(1)}$ (the proportion of valid trajectories $\rho_{0,v}$ decreases in $p^{(1)}$, while both, the average relative error $\bar{\varepsilon}$ and κ increase in $p^{(1)}$). For $p^{(1)} = 0.2$ the CMC is highly inaccurate ($\kappa = 0.328$).

Table 17: Performance of CMC for SAT for $n = 100$, $m = 25$ and $p^{(2)} = 0.5$ for different values of $p^{(1)}$.

N	$p^{(1)}$	$\widehat{\lvert\mathcal{X}^-\rvert}$	$\bar{\varepsilon}$	ε_*	ε^*	$\rho_{0,v}$	κ	CPU
10,000	0.4	$1,779E+007$	0.0048	0.0102	0.0002	0.52	0.004	6
10,000	0.3	$1,653E+006$	0.02	0.042	0.018	0.2	0.019	6
50,000	0.2	$9.359E+003$	0.1571	0.2857	0.03457	$2.5E-004$	0.328	32

Table 18 present the performance of CE for the model in Table 17 with the highest sparsity, that is with $p^{(1)} = 0.2$. One can clearly see that after one iteration CE stabilizes and it outperforms substantially CMC (compare $\kappa = 0.3190$ with $\kappa = 0.0770$). We found that for $p^{(1)} = 0.3$ and $p^{(1)} = 0.4$ (higher densities) the advantage of CE as comared to CMC is minor.

Table 18: Performance of Algorithm CE for $n = 100$, $m = 25$, $p^{(1)} = 0.2$, $p^{(2)} = 0.5$, $N = 50,000$ $\alpha = 0.7$ and $\rho = 0.001$.

t		0	1	2
$\lvert\widehat{\mathcal{X}}^*\rvert$	Average	1.1006E+004	1.0107E+004	9.8091E+003
	Max	1.7448E+004	1.1093E+004	1.1211E+004
	Min	6.7109E+003	8.6099E+003	8.8194E+003
ε	$\bar{\varepsilon}$	0.2537	0.0544	0.0633
	ε^*	0.5854	0.1481	0.1429
	ε_*	0.0244	4.8559E-004	0.0020
$\rho_{t,v}$		2.53E-004	7.17E-003	0.052567
κ		0.3190	0.0713	0.0770

9 Solving #P-Complete Problems Using MinxEnt

The main purpose of using MinxEnt instead of CE is to attempt to obtain a *"better"* IS pdf, and, thus a more accurate estimate of $|\mathcal{X}^-|$.

Note that for a complete graph in HC problem we obtain (similarly to CE) that $f^*(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{P}_{\mathsf{U}})$ and thus, MinxEnt is *exact*. Recall again that $f(\boldsymbol{x}, \boldsymbol{P}_{\mathsf{U}})$ corresponds to the CMC pdf. To make a fair comparison of MinxEnt with CE

1. We shall use the original approach for not too sparse matrices A (say with $\rho_{0,v} > 0.001$), and we shall use the auxiliary one (with the parameter q being small, say $q = 0.001$) for sparse ones (say with $\rho_{0,v} \leq 0.001$).
2. While implementing MinxEnt we assume that the prior pdf $h(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{P}_{\mathsf{U}})$, where, as before, $f(\boldsymbol{x}, \boldsymbol{P}_{\mathsf{U}})$ is the uniform pdf.

For the original approach we shall use the pdf $f(\boldsymbol{x}, \boldsymbol{P}_0^{(o)})$ instead of $f(\boldsymbol{x}, \boldsymbol{P}_0)$, which can be viewed as an IS pdf relative to $h(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{P}_0)$. This in turn implies, for example, that under $h^{(0)}(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{P}_0^{(o)})$ the optimal solution should be written as

$$f^*(\boldsymbol{x}, \lambda^*) = \frac{h(\boldsymbol{x}) \exp\{-S(\boldsymbol{x})\lambda^*\}}{\mathbb{E}_{h^{(0)}}\{[h^{(0)}(\boldsymbol{X})]^{-1} \exp\{-S(\boldsymbol{X})\lambda^*\}\}} \tag{42}$$

and

$$\frac{\mathbb{E}_{h^{(0)}}\{[h^{(0)}(\boldsymbol{X})]^{-1} S(\boldsymbol{X}) \exp\{-\lambda S(\boldsymbol{X})\}\}}{\mathbb{E}_{h^{(0)}}\{[h^{(0)}(\boldsymbol{X})]^{-1} \exp\{-\lambda S(\boldsymbol{X})\}\}} = \gamma, \tag{43}$$

respectively and similarly the stochastic counterparts of (42)-(43). In particular, (under $h^{(0)}(\boldsymbol{x})$) the IS version of the stochastic MaxEnt solution (79)-(80) can be written as

$$\widetilde{p}_{i_1 \cdots i_n} = \frac{\sum_{k=1}^N I_{\{\boldsymbol{X}_k = i_1 \cdots i_n\}} \exp\{-\widehat{\lambda} S(\boldsymbol{X}_k)\}}{\sum_{k=1}^N [h^{(0)}(\boldsymbol{X}_k)]^{-1} \exp\{-\widehat{\lambda} S(\boldsymbol{X}_k)\}}, \tag{44}$$

$$\frac{\sum_{k=1}^N [h^{(0)}(\boldsymbol{X}_k)]^{-1} S(\boldsymbol{X}_k) \exp\{-\lambda S(\boldsymbol{X}_k)\}}{\sum_{k=1}^N [h^{(0)}(\boldsymbol{X}_k)]^{-1} \exp\{-\lambda S(\boldsymbol{X}_k)\}} = \widehat{\gamma}, \tag{45}$$

where the sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ is taken from $h^{(0)}(\boldsymbol{x})$ rather than from $h(\boldsymbol{x})$ and similarly with the auxiliary approach with $f(\boldsymbol{x}, \boldsymbol{P}) \neq f(\boldsymbol{x}\boldsymbol{P}_{\mathsf{U}})$.

To get a more accurate (although a little biased) estimates of $\widehat{\lambda}$ and $\widetilde{p}_{i_1 \cdots i_n}$ one can use (in analogy to (40)) in (44)- (45) partial likelihoods instead of the entire one $\frac{1}{h^{(0)}(\boldsymbol{X})}$.

10 Numerical Results with MinxEnt

We shall present here numerical results for Hamilton cycle, permanent, and satisfiability problems using the stochastic version of the MinxEnt method, developed in [13] and briefly recapitulated in Appendix 2 (see Section 13). It is crucial to note that similar to [13] we shall generate samples from the *marginal* pdf's of $\widetilde{p}_{i_1 \cdots i_n}$ in (44) rather than from the *joint* one $\widetilde{p}_{i_1 \cdots i_n}$. Generating sampes with MinxEnt from the joint pdf would require finding the conditional distributins, which is a time consuming task. While estimating the desired quantity $|\mathcal{X}^-|$ we used both alternatives; the joint pdf $\widetilde{p}_{i_1 \cdots i_n}$ and its marginal ones.

Our numerical results *do not indicate any significant improvement of MinxEnt estimators as compared to their CE counterparts*. We believe that the main reason for that lies in the trajectories generation: *we generate trajectories from the marginal pdf's instead of the true joint one*.

Since there is no significant improvement while using MinxEnt, we shall present only a coupe of tables with the same data sets as for CE and as before we shall use $\rho = 0.001$ and $\alpha = 0.7$. In our tables below we use the same marginal pdfs of the joint one $\widetilde{p}_{i_1 \cdots i_n}$ while generating trajectories and estimating $|\mathcal{X}^-|$.

10.1 MinxEnt for HC's

Tables 19 presents data similar to Table 8 for the same parameters, namely for $n = 30$, $\eta = 0.15$, and $N = 100,000$ for the original approach. We chosed here $\rho = 0.01$ and obtained $\rho_{0,v} \approx 0.00015$, $\rho_{t,v} \approx 0.0048$ (for $t \geq 4$) The execution time was about 21 seconds per iteration. As one can see MinxEnt outpeforms ISO, but there is no (significant) improvement relative to CE.

Table 19: Performance of the MCE Algorithm with the original approach for $n = 30$, $\eta = 0.15$, $N = 100,000$, $\alpha = 0.7$ and $\rho = 0.01$.

t		0	1	2	3	4
$\widehat{\lvert\mathcal{X}_t\rvert}$	Average	4.00E+04	6.55E+04	6.19E+04	6.39E+04	5.45E+04
	Min	9.50E+03	4.40E+04	4.04E+04	4.37E+04	3.24E+04
	Max	1.35E+05	8.62E+04	7.58E+04	7.65E+04	7.81E+04
ε	$\bar{\varepsilon}$	0.557	0.157	0.160	0.120	0.187
	ε_*	0.045	0.015	0.024	0.002	0.013
	ε^*	2.370	0.329	0.348	0.316	0.432
κ		0.897	0.199	0.197	0.164	0.255

10.2 MinxEnt for Permanent

Table 20 presents data similar to Table 11 using the auxiliary approach with $q = 0.001$. Similar to Table 8 we use $n = 30$, $\eta = 0.15$, $\rho = 0.001$ and a sample $N = 300,000$ while updating both, $\boldsymbol{P}$ and $|\mathcal{X}^-|$. For this instance we obtained $\rho_{0,v} \approx 0.00015$, and $\rho_{t,v} \approx 0.019$. The execution time was about 65 seconds per iteration.

Table 20: Performance of Algorithm 3.1 with the auxiliary approach $n = 30$, $\eta = 0.15$, $N = 300,000$, $\rho = 0.001$ and $q = 0.001$.

t		0	1	2	3	4
$\widehat{\lvert\mathcal{X}^-\rvert}$	Average	2.4906e+006	2.2536e+006	2.5133e+006	2.2456e+006	2.4123e+006
	Min	1.9019e+006	2.1828e+006	1.8273e+006	1.5113e+006	1.5807e+006
	Max	3.3895e+006	3.0817e+006	3.1535e+006	2.7940e+006	2.8121e+006
ε	$\bar{\varepsilon}$	0.1622	0.0844	0.1532	0.1582	0.1869
	ε_*	0.0200	0.0013	0.0254	0.0305	0.0319
	ε^*	0.4758	0.2560	0.3885	0.3199	0.3276
κ		0.2609	0.1321	0.1724	0.1837	0.1801

10.3 MinxEnt for SAT

Table 21 present the performance of MinxEnt for the same model as CE in Table 18. Again, one can see that after one iteration MinxEnt stabilizes and it outperforms substantially CMC (compare $\kappa = 0.2127$ with $\kappa = 0.0550$) and that there is no advantage by using MinxEnt versus CE (compare Tables 18 and 21) .

Table 21: Performance of MinxEnt for $n = 100$, $m = 25$, $p^{(1)} = 0.2$, $p^{(2)} = 0.5$, $N = 50,000$, $\alpha = 0.7$ and $\rho = 0.001$.

t		0	1	2
$\lvert\widehat{\mathcal{X}}^*\rvert$	Average	9.6637E+003	9.8180E+003	1.0147E+004
	Max	1.4093E+004	1.0774E+004	1.1167E+004
	Min	7.3820E+003	9.2355E+003	9.1166E+003
ε	$\bar{\varepsilon}$	0.1583	0.0423	0.0407
	ε^*	0.4583	0.0974	0.1016
	ε_*	0.0278	0.0087	0.0011
$\rho_{t,v}$		2.85E-004	3.63E-03	0.02075
κ		0.2127	0.0539	0.0550

It is interesting to not that we found numerically that employing in addition the screening algorithm of [9] the accuracy of both CE and MinxEnt (for this particular model) increases (κ decreases) by a factor of 3. More research on SAT and the screening algorithm is underway.

11 Concluding Remarks and Further Research

In this paper we presented two algorithms (CE and MinxEnt) for estimating the quantities $|\mathcal{X}^-|$ in #P-complete counting problems, like the number of HC's, the permanent, the number of self-avoiding walks and the satisfiability problem. In both algorithms we first cast the graph into the framework of rare-event, and then apply dynamic importance sampling (although different for CE and MinxEnt) to estimate $|\mathcal{X}^-|$. We showed that

1. The straightforward ISO method typically outperforms both CE and MinxEnt (with marginal pdfs), provided the incidence matrix A is not too sparse and the proportion of valid trajectories at $t = 0$ is substantial ($\geq \rho$). For sparse matrices, both the multi-iterative CE and multi-iterative MinxEnt outperform ISO, but both requires quite a large sample to get reliable estimates. To overcome this difficulty some alternative approaches should be introduced, like variance minimization [15], gradient-based procedures and the exponential change of measure. Another alternative is to employ the screening algorithm of [9], to identify (nulify) the so-called non-important and redundant elements in $\boldsymbol{P}_0^{(0)}$ associated with the unity elements of matrix A. Our preliminary numerical results with HC and SAT clearly indicate a substantial improvement (variance reduction), while using of the screening algorithm in both CE and MinxEnt.

2. Our simulation results with MinxEnt show that using in MinxEnt the marginal pdfs instead of the true joint one does not lead to improvement of the counting estimators as compared to CE.

As for further research we suggest to

1. Apply the above methodology to a broad variety of #P-complete counting problems.

2. Investigate the impact (improvement) by using the screening algorithm for counting problems.

3. Investigate the efficiency of CE and MinxEnt on larger sizes of #P-complete counting problems.

4. Investigate the efficiency of some other ϕ-divergence entropic type programs (note that MinxEnt is a particular case of the ϕ-divergence, see [8]) for #P-complete counting problems. In particular we are interested in finding a ϕ-divergence, which leads to convenient sample generation from its resulting joint pdf f^*.

5. Investigate the efficiency of CE and MinxEnt for non-symmetric SAW, where a certain proportion of movements, (chosen, say randomly) are forbidden.

6. Establish rigorous theorems on convergence and speed of convergence of the presented approaches.

7. Establish exact relationships on how in a SAW problem the proportion of valid trajectories $\rho_{0,v}(n)$ decreases in n.

8. Establish exact relationships on how in a HC's problem (using the original approach) the proportion of valid trajectories $\rho^{(o)}_{0,v}(n,\eta)$ decreases in both n and η, provided the number of unity elements in A is generated randomly.

9. Compare the efficiency (accuracy) of MinxEnt based on the marginal pdf's with the one based on true joint one f^*.

10. Compare the accuracy of the CE method with the one based on variance minimization and gradient-based procedures.

11. Investigate the efficiency of CE and MinxEnt, while using alternative to node placement and node transition method for trajectory generation.

12. Apply the above methodology to investigate the following problems:

    (a) Let H^* be the optimal (best known) solution of a combinatorial problem, like the shortest tour in TSP. Let us perturb H^* be ε. We want to know how many trajectories contains the vicinity (interval) $\{H^*, H^*+\varepsilon\}$.

    (b) Given a number k of best solutions, say the number of the shortest tours on TSP, find ε such that the interval $\{H^*, H^*+\varepsilon\}$ contains exactly the k best solutions.

12 Appendix 1: Background on CE

In this section we briefly review the main ideas behind the CE algorithm [15] having in mind estimation of the permanent (27). As mentioned we associate with the original deterministic problem a relevant *estimation problem*. The associated estimation problem (ASP) uses an auxiliary family of probability density functions (pdf's) $f(\boldsymbol{x},\boldsymbol{u})$, with $\boldsymbol{u}\in\mathcal{V}$ being the parameter vector of the pdf f and can be written as

$$\ell(\gamma) = \mathbb{P}_{\boldsymbol{u}}\{S(\boldsymbol{X})\geq\gamma\} = \mathbb{E}_{\boldsymbol{u}} I_{\{S(\boldsymbol{X})\geq\gamma\}}. \tag{46}$$

Here, γ is a known parameter and $\boldsymbol{X}$ is a random vector with pdf $f(\boldsymbol{x},\boldsymbol{u})$, for some $\boldsymbol{u}\in\mathcal{V}$.

The naive way to estimate ℓ is to use *crude Monte-Carlo* (CMC) simulation: Draw a random sample $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_N$ from $f(\cdot\,;\boldsymbol{u})$; then

$$\frac{1}{N}\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i)\geq\gamma\}}$$

is an unbiased estimator of ℓ. However, this poses serious problems when $\{S(\boldsymbol{X}) \geq \gamma\}$ is a rare event since a large simulation effort is required in order to estimate ℓ accurately. Alternatively, we can estimate ℓ using *importance sampling* (IS) as

$$\widehat{\ell} = \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} W(\boldsymbol{X}_i, \boldsymbol{u}, \boldsymbol{p}), \tag{47}$$

where $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ is a random sample from $f(\boldsymbol{x}, \boldsymbol{p})$ using a *different* reference parameter – and

$$W(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) = \frac{f(\boldsymbol{x}, \boldsymbol{u})}{f(\boldsymbol{x}, \boldsymbol{p})}, \tag{48}$$

is the *likelihood ratio*.

The optimal in the cross-entropy sense reference vector $\boldsymbol{p}$ in (47) [15] is

$$\boldsymbol{p}^* = \underset{\boldsymbol{p}}{\operatorname{argmax}} \, \mathbb{E}_{\boldsymbol{u}} \, I_{\{S(\boldsymbol{X}) \geq \gamma\}} W(\boldsymbol{X}, \boldsymbol{u}, \boldsymbol{w}) \ln f(\boldsymbol{x}, \boldsymbol{p}), \tag{49}$$

where $\boldsymbol{w}$ is an arbitrary parameter vector in $f(\boldsymbol{x}, \boldsymbol{w})$.

The estimate of the optimal cross-entropy reference parameter vector $\boldsymbol{p}^*$ can be obtained from the solution of solution of the iterative procedure

$$\widetilde{\boldsymbol{p}}^* = \underset{\boldsymbol{p}}{\operatorname{argmax}} \, \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} \, W(\boldsymbol{X}_i, \boldsymbol{u}, \boldsymbol{w}) \, \ln f(\boldsymbol{X}_i, \boldsymbol{p}) \, . \tag{50}$$

The solution of (50) can often be determined *analytically*.

For rare-event estimation problems, (50) is difficult to carry out: because of the rareness of the event most of the indicators $I_{\{S(\boldsymbol{X}_i) \geq \gamma\}}$ will be zero. For these problems a two-phase cross-entropy procedure is employed, in which apart from $\boldsymbol{p}$ the *level* parameter γ is also updated, creating a sequence of 2-tuples $\{(\widetilde{\boldsymbol{p}}_t, \widehat{\gamma}_t)\}$ with the goal of estimating the optimal cross-entropy reference parameter $\boldsymbol{p}^*$. Starting with $\widetilde{\boldsymbol{p}}_0 = \boldsymbol{u}$ (the original or nominal parameter), the updating rules are as follows: Given a random sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from $f(\boldsymbol{x}, \widetilde{\boldsymbol{p}}_{t-1})$, we concentrate on the best performing ρ–portion of the samples. Let $\widehat{\gamma}_t$ be the sample $(1-\rho)$–quantile of the performances $S(\boldsymbol{X}_i)$, $i = 1, \ldots, N$, provided the sample quantile is less than γ; otherwise we set γ_t to γ. In other words,

$$\widehat{\gamma}_t = \min\{\gamma, \, S_{(\lceil (1-\rho)N \rceil)}\}, \tag{51}$$

where $S_{(j)}$ is the j-th *order-statistic* of the performances. Using the *same sample*, we let

$$\widetilde{\boldsymbol{p}}_t = \underset{\boldsymbol{p}}{\operatorname{argmax}} \, \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \widehat{\gamma}_{t-1}\}} \, W(\boldsymbol{X}_i, \boldsymbol{u}, \widetilde{\boldsymbol{p}}_{t-1}) \, \ln f(\boldsymbol{X}_i, \boldsymbol{p}) \, . \tag{52}$$

We proceed by setting $\boldsymbol{p}_0 = \boldsymbol{u}$, choosing a not very small ρ, called the *rarity parameter*, 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,

$$\widehat{\gamma}_t = S_{(\lfloor (1-\rho)N \rfloor + 1)}. \tag{53}$$

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

$$\max_{\boldsymbol{p}} D(\boldsymbol{p}) = \max_{\boldsymbol{p}} \mathbb{E}_{\boldsymbol{p}_{t-1}} I_{\{S(\boldsymbol{X}) \geq \gamma_{t-1}\}} \; W(\boldsymbol{X}, \boldsymbol{u}, \boldsymbol{p}_{t-1}) \; \ln f(\boldsymbol{X}; \boldsymbol{p}) \; . \tag{54}$$

The stochastic counterpart of (54) is as follows: for fixed $\widehat{\gamma}_t$ and $\widetilde{\boldsymbol{p}}_{t-1}$, derive $\widetilde{\boldsymbol{p}}_t$ from the following program

$$\max_{\boldsymbol{p}_t} \widehat{D}(\boldsymbol{p}) = \max_{\boldsymbol{p}_t} \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \widehat{\gamma}_{t-1}\}} \; W(\boldsymbol{X}_i, \boldsymbol{u}, \widetilde{\boldsymbol{p}}_{t-1}) \; \ln f(\boldsymbol{X}_i; \boldsymbol{p}) \; . \tag{55}$$

Note that in (53) and (55) the same sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from $f(\boldsymbol{x}, \widetilde{\boldsymbol{p}}_{t-1})$ is used.

Solving, for example, (54) and (55) for the TSP problem we obtain the following simple analytic expressions

$$p_{t,ij} = \frac{\mathbb{E}_{\boldsymbol{p}_{t-1}} I_{\{\boldsymbol{X} \in \mathcal{X}_{ij}\}} I_{\{S(\boldsymbol{X}) \geq \gamma_{t-1}\}} \; W(\boldsymbol{X}, \boldsymbol{u}, \boldsymbol{p}_{t-1})}{\mathbb{E}_{\boldsymbol{p}_{t-1}} I_{\{S(\boldsymbol{X}) \geq \gamma_{t-1}\}} \; W(\boldsymbol{X}, \boldsymbol{u}, \boldsymbol{p}_{t-1})} \tag{56}$$

and

$$\widetilde{p}_{t,ij} = \frac{\sum_{k=1}^{N} I_{\{\boldsymbol{X}_k \in \mathcal{X}_{ij}\}} \, I_{\{S(\boldsymbol{X}_i) \geq \widehat{\gamma}_{t-1})\}} \; W(\boldsymbol{X}_i, \boldsymbol{u}, \widetilde{\boldsymbol{p}}_{t-1})}{\sum_{k=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \widehat{\gamma}_{t-1})\}} \; W(\boldsymbol{X}_i, \boldsymbol{u}, \widetilde{\boldsymbol{p}}_{t-1})} \; , \tag{57}$$

for updating the parameters of $\boldsymbol{p}_t$ and $\widetilde{\boldsymbol{p}}_t$, respectively. Here $\mathcal{X}_{ij}$ is the set of tours in which the transition from i to j is made.

To proceed, note that instead of using the parameter vector $\widetilde{\boldsymbol{p}}_t$ obtained directly from (57) we use its following *smoothed* version

$$\widehat{\boldsymbol{p}}_t = \alpha \widetilde{\boldsymbol{p}}_t + (1 - \alpha) \widetilde{\boldsymbol{p}}_{t-1}, \tag{58}$$

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.

Algorithm 12.1 [CE Algorithm for Rare Event Simulation]

1. Define $\widehat{\boldsymbol{p}}_0 = \boldsymbol{u}$. Set $t = 1$ (iteration = level counter).
2. Generate a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from the density $f(\boldsymbol{x}, \widetilde{\boldsymbol{p}}_{t-1})$ and compute $\widehat{\gamma}_t$, the $(1-\rho)$-quantile of its performance, according to (53). If $S_{(\lceil (1-\rho)N \rceil)} < \gamma$, set $\widehat{\gamma}_t = S_{(\lceil (1-\rho)N \rceil)}$; otherwise, set $\widehat{\gamma}_t = \gamma$.
3. Use the **same** sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ and deliver the solution (57) of the stochastic program (55). Denote the solution by $\widetilde{\boldsymbol{p}}_t$.
4. Apply (58) to smooth out the vector $\widetilde{\boldsymbol{p}}_t$ and obtain $\widehat{\boldsymbol{p}}_t$.
5. If $\widehat{\gamma}_t < \gamma$, set $t = t + 1$ and reiterate from step 2. Else proceed with step 6.

6. Estimate the rare-event probability ℓ using

$$\widehat{\ell} = \frac{1}{N_1} \sum_{i=1}^{N_1} I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} \, W(\boldsymbol{X}_i, \boldsymbol{u}, \widehat{\boldsymbol{p}}_T), \tag{59}$$

where T denotes the final number of iterations (= number of levels used).

Note that in typical applications the sample size N in step 2 should chosen smaller than the final sample size N_1 in step 5.

The prove of convergence of Algorithm 3.1 is given in [15].

Below we also present for convenience two algorithms for trajectory generation in a TSP [15]. The first one is based on node transitions, while the second one on node placement [15].

In the node transitions algorithm one defines the following matrix

$$\boldsymbol{P} = \begin{pmatrix} 0 & p_{1,2} & \dots & p_{1,n} \\ p_{2,1} & 0 & \dots & p_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n,1} & p_{n,2} & \dots & 0 \end{pmatrix}, \tag{60}$$

and then proceed as follows:

Algorithm 12.2 [Trajectory Generation Using Node Transitions]

1. Define $\boldsymbol{P}^{(1)} = \boldsymbol{P}$ and $X_1 = 1$. Let $k = 1$.
2. Obtain $\boldsymbol{P}^{(k+1)}$ from $\boldsymbol{P}^{(k)}$ by first setting the X_k-th column of $\boldsymbol{P}^{(k)}$ to $\boldsymbol{0}$ and then normalizing the rows to sum up to 1. Generate X_{k+1} from the distribution formed by the X_k-th row of $\boldsymbol{P}^{(k)}$.
3. If $k = n - 1$ then **stop**; otherwise set $k = k + 1$ and reiterate from step 2.

In the node placement algorithm one defines the following matrix

$$\boldsymbol{P} = \begin{pmatrix} p_{(1,1)} & p_{(1,2)} & \dots & p_{(1,n)} \\ p_{(2,1)} & p_{(2,2)} & \dots & p_{(2,n)} \\ \vdots & \vdots & \vdots & \vdots \\ p_{(n,1)} & p_{(n,2)} & \dots & p_{(n,n)} \end{pmatrix}. \tag{61}$$

Here $p_{(i,j)}$ corresponds to the probability of node i being visited at the $j-th$ place in a tour of n cities. In other words, $p_{(i,j)}$ can be viewed as probability that city (node) i is "arranged" to be visited at the $j-th$ place in a tour of n cities.

More formally, a *node placement vector* is a vector $\boldsymbol{y} = (y_1, \dots, y_n)$ such that y_i denotes the "place" of node i in the tour $\boldsymbol{x} = (x_1, \dots, x_n)$. The precise meaning is given by the correspondence.

$$y_i = j \quad \Longleftrightarrow \quad x_j = i, \tag{62}$$

for all $i, j \in \{1, \dots, n\}$.

Algorithm 12.3 (Trajectory Generation Using Node Placements)

1. Define $\boldsymbol{P}^{(1)} = \boldsymbol{P}$. Let $k = 1$.
2. Generate Y_k from the distribution formed by the k-th row of $\boldsymbol{P}^{(k)}$. Obtain the matrix $\boldsymbol{P}^{(k+1)}$ from $\boldsymbol{P}^{(k)}$ by first setting the Y_k-th column of $\boldsymbol{P}^{(k)}$ to 0 and then normalizing the rows to sum up to 1.
3. If $k = n$ then **stop**; otherwise set $k = k + 1$ and reiterate from step 2.
4. Determine the tour by using (62).

13 Appendix 2: A Stochastic MinxEnt Method for Combinatorial Optimization

13.1 Background on the MinxEnt Method

The MinxEnt program [8] reads as

$$(\mathrm{P}_0)\qquad \begin{array}{ll} & \min_{f(\boldsymbol{x})} \left\{ \mathcal{D}(f|h) = \int \ln \frac{f(\boldsymbol{x})}{h(\boldsymbol{x})} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_f \ln \frac{f(\boldsymbol{X})}{h(\boldsymbol{X})} \right\} \\ \text{s.t.} & \int S_j(\boldsymbol{x}) f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_f S_j(\boldsymbol{X}) = \gamma_j,\ j = 1, \ldots, k, \\ & \int f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1. \end{array} \tag{63}$$

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. If h is *unknown*, it is taken as a uniform (continuous or discrete) pdf. In this case

$$\min_{f(\boldsymbol{x})} \left\{ \mathcal{D}(f|h) = \int \ln \frac{f(\boldsymbol{x})}{h(\boldsymbol{x})} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_f \ln \frac{f(\boldsymbol{X})}{h(\boldsymbol{X})} \right\} \tag{64}$$

reduces to

$$\max_{f(\boldsymbol{x})} \left\{ \mathcal{H}(f) = -\int f(\boldsymbol{x}) \ln f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = -\mathbb{E}_f \ln f(\boldsymbol{X}) \right\}. \tag{65}$$

The original program (P_0) is called the *Kullback MinxEnt* program, while the program (P_0) with (64) replaced by (65) is called the *Janes MaxEnt* program. Note that the former minimizes the Kullback-Leibler cross-entropy, while the latter maximizes the Shannon entropy [8].

The solution of the MinxEnt program is [8]

$$f^*(\boldsymbol{x}) = \frac{h(\boldsymbol{x}) \exp\left\{ -\sum_{r=1}^k S_r(\boldsymbol{x}) \lambda_r^* \right\}}{\mathbb{E}_h \exp\left\{ -\sum_{r=1}^k S_r(\boldsymbol{X}) \lambda_r^* \right\}}, \tag{66}$$

where $\lambda_r^*,\ r = 1, \ldots, k$ are obtained from the solution of the following system of equations

$$\frac{\mathbb{E}_h S_r(\boldsymbol{X}) \exp\left\{ -\sum_{r=1}^k S_r(\boldsymbol{X}) \lambda_r \right\}}{\mathbb{E}_h \exp\left\{ -\sum_{r=1}^k S_r(\boldsymbol{X}) \lambda_r \right\}} = \gamma_r, \tag{67}$$

respectively, where $\boldsymbol{X} \sim h(\boldsymbol{x})$. Note that $f^*(\boldsymbol{x})$ can be written as

$$f^*(\boldsymbol{x}) = C(\lambda^*) h(\boldsymbol{x}) \exp\left\{ -\sum_{r=1}^k S_r(\boldsymbol{x}) \lambda_r^* \right\}, \tag{68}$$

where

$$C^{-1}(\lambda^*) = \mathbb{E}_h \exp\left\{ -\sum_{r=1}^k S_r(\boldsymbol{X}) \lambda_r^* \right\} \tag{69}$$

is the normalization constant.

It is important to note that in the the Bayesian framework, $h(\boldsymbol{x})$ can be viewed as the prior pdf, while the MinxEnt optimal pdf $f^*(\boldsymbol{x})$ as the posterior pdf.

Consider the MinxEnt program (P_0) with a single constraint, that is

$$\mathbb{E}_f S(\boldsymbol{X}) = \gamma,\ (\int f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1). \tag{70}$$

In this case (66) and (67) reduce to

$$f^*(\boldsymbol{x},\lambda^*)=\frac{h(\boldsymbol{x})\exp\{-S(\boldsymbol{x})\lambda^*\}}{\mathbb{E}_h\exp\{-S(\boldsymbol{X})\lambda^*\}} \tag{71}$$

and

$$\frac{\mathbb{E}_h S(\boldsymbol{X})\exp\{-\lambda S(\boldsymbol{X})\}}{\mathbb{E}_h\exp\{-\lambda S(\boldsymbol{X})\}}=\gamma, \tag{72}$$

respectively.

In the particular case where $S(\boldsymbol{x}),\ \boldsymbol{x}=(x_1,\ldots,x_n)$ is a separable function, that is

$$S(\boldsymbol{x})=\sum_{i=1}^{n}S_i(x_i) \tag{73}$$

and the components $X_i,\ i=1,\ldots,X_n$ of the random vector $\boldsymbol{X}=(X_1,\ldots,X_n)$ are independent the joint pdf $f^*(\boldsymbol{x})$ in (71) reduces to the *product of marginal pdf's*. In particular, the k-th component of $f^*(\boldsymbol{x})$ can be written as

$$f_k^*(x_k,\lambda^*)=\frac{h_k(x_k)\exp\{-\lambda^* S_k(x_k)\}}{\mathbb{E}_{h_k}\exp\{-\lambda^* S_k(x_k)\}},\ k=1,\ldots,n. \tag{74}$$

Remark 13.1 It is well known [4] that the optimal solution of the single-dimensional single-constrained the MinxEnt program

$$\begin{aligned} &\min_{f(x)}\left\{\mathcal{D}(f|h)=\mathbb{E}_f\ln\tfrac{f(X)}{h(X)}\right\}\\ \text{s.t.}\quad &\mathbb{E}_f S(X)=\gamma,\\ &\int f(x)\mathrm{d}x=1 \end{aligned} \tag{75}$$

coincide with cellebrated optimal *exponential change of measure* (ECM). Note, however, that in the multi-dimensional case

- The optimal ECM tilting distributions presents a product of marginal tilting distributions parameterized by an *n-dimensional vector*. In contrast, the optimal solution $f^*(\boldsymbol{x},\lambda^*)$ (see (71)) of the MinxEnt program (P_0) is parameterized by a *single-dimensional* parameter λ^*.

- Even for a *separable function* $S(\boldsymbol{x})$, like (70), where both the ECM and the MinxEnt pdf, present a product of marginal distributions, they still differ each from other in the sense that the former is parameterized by a *n-dimensional vector* $\boldsymbol{t}^*$, while the later, by a *single-dimensional* parameter λ^*. For *non-separable functions*, the optimal parametric ECM $f(\boldsymbol{x},\boldsymbol{t}^*)$ pdf (in the product form), typically differs substantially from the joint MinxEnt pdf $f^*(\boldsymbol{x},\lambda^*)$ in (71).

This is the main reason that ECM has limited applications; it works nicely for separable or "separable-like" functions $S(\boldsymbol{x})$ with light-tailed pdf's, but fails for quite general ones. The advantage of all parametric approaches, like EMC and CE, as compared to the non-parametric MinxEnt is that generation samples from marginal pdf's is simple, while generation from a joint pdf of the type (71) is more difficult.

13.2 The Stochastic MCE method

If not otherwise stated, we consider below only the single constrained 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. 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

$$
\begin{aligned}
& & & \min_{\boldsymbol{p}} \mathcal{D}(\boldsymbol{p}|\boldsymbol{u}) = \min_{\boldsymbol{p}} \textstyle\sum_{\boldsymbol{x}} \boldsymbol{p}(\boldsymbol{x}) \ln \frac{\boldsymbol{p}(\boldsymbol{x})}{\boldsymbol{u}(\boldsymbol{x})} \\
& (\mathrm{P}_1) & & \text{s. t.} \quad \textstyle\sum_{\boldsymbol{x}} S(\boldsymbol{x})\boldsymbol{p}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{p}} S(\boldsymbol{X}) = \gamma \\
& & & \textstyle\sum_{\boldsymbol{x}} \boldsymbol{p}(\boldsymbol{x}) = \sum_{i=1}^{m} p_i = 1.
\end{aligned}
\tag{76}
$$

The MaxEnt counterpart of (76) can be written as

$$
\begin{aligned}
& \max_{\boldsymbol{p}} \{\mathcal{H}(\boldsymbol{p}) = \textstyle\sum_{\boldsymbol{x}} \boldsymbol{p}(\boldsymbol{x}) \ln \boldsymbol{p}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{p}} \ln \boldsymbol{p}(\boldsymbol{X})\} \\
\text{s. t.} \quad & \textstyle\sum_{\boldsymbol{x}} S(\boldsymbol{x})\boldsymbol{p}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{p}} S(\boldsymbol{X}) \ = \gamma, \\
& \textstyle\sum_{\boldsymbol{x}} \boldsymbol{p}(\boldsymbol{x}) = 1.
\end{aligned}
\tag{77}
$$

In the stochastic MaxEnt and MinxEnt programs we shall assume that

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

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

Consider first the stochastic (sample average) version of the MaxEnt program (77). It can be written as

$$
\begin{aligned}
& \max_{\widetilde{\boldsymbol{p}}} \widehat{\mathcal{H}}(\widetilde{\boldsymbol{p}}) = \min_{\widetilde{\boldsymbol{p}}} \textstyle\sum_{k=1}^{N} \widetilde{\boldsymbol{p}}(\boldsymbol{X}_k) \ln \widetilde{\boldsymbol{p}}(\boldsymbol{X}_k) \\
\text{s. t.} \quad & \textstyle\sum_{k=1}^{N} S(\boldsymbol{X}_k)\widetilde{\boldsymbol{p}}(\boldsymbol{X}_k) = \widehat{\gamma}, \\
& \textstyle\sum_{k=1}^{N} \widetilde{\boldsymbol{p}}(\boldsymbol{X}_k) = 1,
\end{aligned}
\tag{78}
$$

where $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_N$ is a sample taken from the pdf $\widetilde{\boldsymbol{p}} = \{\widetilde{p}_{i_1\cdots i_n};\ i_r = 1,\ldots,m;\ r = 1,\ldots,n\}$, and $\widetilde{\boldsymbol{p}}$ denotes the estimate of $\boldsymbol{p}$ in (77).

It is readily seen that the solution of (78) is

$$
\widetilde{p}_{i_1\cdots i_n} = \frac{\sum_{k=1}^{N} I_{\{\boldsymbol{X}_k = i_1\cdots i_n\}} \exp\{-\widehat{\lambda} S(\boldsymbol{X}_k)\}}{\sum_{k=1}^{N} \exp\{-\widehat{\lambda} S(\boldsymbol{X}_k)\}},
\tag{79}
$$

where $\widehat{\lambda}$, the estimate of λ can be obtained from the solution of the following non-linear equation

$$
\frac{\sum_{k=1}^{N} S(\boldsymbol{X}_k) \exp\{-\lambda S(\boldsymbol{X}_k)\}}{\sum_{k=1}^{N} \exp\{-\lambda S(\boldsymbol{X}_k)\}} = \widehat{\gamma}.
\tag{80}
$$

For further details see [13].

Acknowledgments

I would like to thank Shie Manor and Dirk Kroese for many valuable suggestions and some insightful remarks and to Yohai Gat and Alexander Libster from the Technion for performing the computational part for this paper.

References

[1] "Boolean Satisfiability problem", from Wikipedia.

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

[3] Dyer, M., Frieze, A., and M. Jerrum. *Approximately Counting Hamiltonian Paths and Cycles in Dense Graphs*. SIAM Journal of Computing, **27**, No. 5, 1262-1272, 1998.

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

[5] Lieber, D., Rubinstein, R.Y. and Elmakis, D., "Quick Estimation of Rare Events in Stochastic Networks", *IEEE Transactions on Reliability Systems*, vol 46, No 2, 254-265, 1997.

[6] Jerrum M., Sinclair A., and E. Vigoda. "A Polynomial-Time Approximation Algorithm for the Permanent of a Matrix with the Nonnegarive Entries". Journal of the ACM, Vol. 51, No 4, pp. 671-697, 2004.

[7] Jun Gu, et al "Algorithms for the Satisfiability (SAT) problem: A survey", DIMACS Series in Discrete Mathematics, 1991.

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

[9] Lieber, D., Rubinstein, R.Y. and Elmakis, D., "Quick Estimation of Rare Events in Stochastic Networks", *IEEE Transactions on Reliability Systems*, vol 46, No 2, 254-265, 1997.

[10] Motwani R., and R. Raghavan. *Randomized Algorithms* Cambridge University Press, 1997.

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

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

[13] R. Y. Rubinstein " A Stochastic Minimum Cross-Entropy Method for Combinatorial Optimization and Rare-event Estimation", *Methodology and Computing in Applied Probability*, No 1, pp 1-46, 2005.

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

[15] 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.

[16] Welsh D. J. A. *Complexity: Knots, Colouring and Counting*, Cambridge University Press, 1993.