Methodol Comput Appl Probab
DOI 10.1007/s11009-008-9101-7

The Gibbs Cloner for Combinatorial Optimization, Counting and Sampling

Reuven Rubinstein

Received: 23 April 2008 / Revised: 8 August 2008 /
Accepted: 14 August 2008

Abstract We present a randomized algorithm, called the cloning algorithm, for approximating the solutions of quite general NP-hard combinatorial optimization problems, counting, rare-event estimation and uniform sampling on complex regions. Similar to the algorithms of Diaconis–Holmes–Ross and Botev–Kroese the cloning algorithm is based on the MCMC (Gibbs) sampler equipped with an importance sampling pdf and, as usual for randomized algorithms, it uses a sequential sampling plan to decompose a "difficult" problem into a sequence of "easy" ones. The cloning algorithm combines the best features of the Diaconis–Holmes–Ross and the Botev–Kroese. In addition to some other enhancements, it has a special mechanism, called the "cloning" device, which makes the cloning algorithm, also called the *Gibbs cloner* fast and accurate. We believe that it is the fastest and the most accurate randomized algorithm for counting known so far. In addition it is well suited for solving problems associated with the Boltzmann distribution, like estimating the partition functions in an Ising model. We also present a combined version of the cloning and cross-entropy (CE) algorithms. We prove the polynomial complexity of a particular version of the Gibbs cloner for counting. We finally present efficient numerical results with the Gibbs cloner and the combined version, while solving quite general integer and combinatorial optimization problems as well as counting ones, like SAT.

Keywords Gibbs sampler · Importance sampling · Cross-entropy · Rare-event · Combinatorial optimization · Counting

AMS 2000 Subject Classifications 65C05 · 60C05 · 68W20 · 90C59

This research was supported by the BSF (Binational Science Foundation, grant No 2008482).

R. Rubinstein (✉)
Faculty of Industrial Engineering and Management,
Technion, Israel Institute of Technology, Haifa, Israel
e-mail: ierrr01@ie.technion.ac.il
URL: iew3.technion.ac.il:8080/ierrr01.phtml

1 Introduction: Randomized Algorithms for Counting and Optimization

The main idea of randomized algorithms for counting (Mitzenmacher and Upfal 2005) is to design a sequential sampling plan, where the "difficult" problem of counting on the set $\mathcal{X}^*$ is decomposed into "easy" ones of counting the number of elements in a sequence of related sets $\mathcal{X}_1, \ldots, \mathcal{X}_m$. Typically, randomized algorithms explore the connection between counting and sampling problems and in particular the reduction from approximately counting the cardinality of a discrete set to approximately sampling elements of the set, where the sampling is performed by employing the classic MCMC method (Bezakova et al. 2007). A typical randomized algorithm contains the following steps:

1. Formulate the counting problem as the problem of estimating the cardinality of some set $\mathcal{X}^*$.
2. Find sets $\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_m$ such that $|\mathcal{X}_m| = |\mathcal{X}^*|$, and $|\mathcal{X}_0|$ is known.
3. Write $|\mathcal{X}^*| = |\mathcal{X}_m|$ as
$$|\mathcal{X}^*| = |\mathcal{X}_0| \prod_{j=1}^{m} \frac{|\mathcal{X}_j|}{|\mathcal{X}_{j-1}|}, \tag{1}$$
4. Develop an efficient estimator $\widehat{\zeta_j}$ for each $\zeta_j = |\mathcal{X}_j|/|\mathcal{X}_{j-1}|$, resulting in an efficient estimator
$$|\widehat{\mathcal{X}^*}| = |\mathcal{X}_0| \prod_{j=1}^{m} \widehat{\zeta}_j, \tag{2}$$
for $|\mathcal{X}^*|$ and similar for rare event estimation.

Algorithms based on the sequential Monte Carlo sampling estimator (2) are called in the computer literature (Mitzenmacher and Upfal 2005), *randomized algorithms*. For a classic references on sequential Monte Carlo methods and their relation to Feynman–Kac formulae see (Del Moral 2004).

It is shown in Bezakova et al. (2007), Mitzenmacher and Upfal (2005), Motwani and Raghavan (1997) that many interesting counting problems like estimating the volume of a convex body, computing the permanent of a non-negative matrix and counting the number of independence sets in a graph can be put into the setting (1), (2).

Quite often randomized algorithms deal with estimation of a partition function $Z(\lambda)$ of the system at some desired temperature λ via generation samples from the Boltzmann distribution. This can be achieved by computing the ratios of the partition functions similar to Eq. 1 for a carefully designed sequence of temperatures $0 = \lambda_0 < \lambda_1 < \ldots < \lambda_T = \lambda$, also called cooling schedule, where $Z(0)$ is trivial to compute and the ratios $Z(\lambda_{i+1})/Z(\lambda_i)$ are easy to estimate by sampling from the distribution corresponding to $Z(\lambda_i)$ (Stefankovic et al. 2007). The challenging problem is to define a "smart" cooling schedule. The best known is an adaptive one due to (Stefankovic et al. 2007), which has length $T = O^*(\sqrt{\ln Z(0)})$. In particular for some well-studied problems such as estimating the partition function of the Ising model, or approximating the number of colorings or matchings of a graph, their cooling schedule (Stefankovic et al. 2007) is of length $O^*(\sqrt{n})$, where n is the size of the problem.

Here we present a randomized algorithm, called the *cloning* algorithm. Similar to the algorithms of Diaconis–Holmes–Ross (Diaconis and Holmes 1994), Ross (2002a, b) and Botev–Kroese (Botev and Kroese 2008) the cloning algorithm is based on the MCMC (Gibbs) sampler equipped with an importance sampling pdf and, as usual for randomized algorithms, it uses a sequential sampling plan to decompose a "difficult" problem into a sequence of "easy" ones. In addition to some other enhancements to Diaconis–Holmes–Ross (DHR) and Botev–Kroese (BK), it has a special mechanism, called the "cloning" device, which makes it fast and accurate. In particular, the cloning algorithm, also called, the *Gibbs cloner*, is well suited for counting the satisfiability assignments in a SAT problem and for solving problems associated with the Boltzmann distribution, like estimating the partition functions in an Ising model and for sampling random variables uniformly distributed on different convex bodies.

In all optimization and counting problems discussed in this paper we have applied Gibbs simpler. For some complex function the Gibbs sampler might not be appropriate. In this case one might use more sophisticated MCMC samplers discussed in Botev and Kroese (2008), Rubinstein and Kroese (2007), Ghate and Smith (2008) and in particular the hit and run method pioneered by Smith (1984) and generalized by Diaconis and Andersen (2007) could be used.

To proceed, consider estimation of the following rare event probability

$$\ell(m) = \mathbb{E}_f\left[I_{\{S(\boldsymbol{X})\geq m\}}\right]. \tag{3}$$

Here $S(\boldsymbol{X})$ is the sample performance, $\boldsymbol{X} \sim f(\boldsymbol{x})$ and m is fixed.

To estimate $\ell(m)$, similar to Eq. 1 we shall use the well known chain rule (nested events) and write $\ell(m)$ as

$$\ell(m) = \mathbb{E}_f[I_{\{S(\boldsymbol{X})\geq m_0\}}]\prod_{t=1}^{T}\mathbb{E}_f[I_{\{S(\boldsymbol{X})\geq m_t\}}|I_{\{S(\boldsymbol{X})\geq m_{t-1}\}}] = c_0\prod_{t=1}^{T}c_t. \tag{4}$$

Note that $\ell(m)$ can be also written as

$$\ell(m) = \mathbb{E}_f[I_{\{S(\boldsymbol{X})\geq m_0\}}]\prod_{t=1}^{T}\mathbb{E}_{g^*_{t-1}}[I_{\{S(\boldsymbol{X})\geq m_t\}}] = \ell_0\prod_{t=1}^{T}\frac{\ell_t}{\ell_{t-1}} = c_0\prod_{t=1}^{T}c_t, \tag{5}$$

where $\ell(m_t) = \mathbb{E}_f\left[I_{\{S(\boldsymbol{X})\geq m_t\}}\right]$,

$$c_t = \mathbb{E}_f[I_{\{S(\boldsymbol{X})\geq m_t\}}|I_{\{S(\boldsymbol{X})\geq m_{t-1}\}}] = \mathbb{E}_{g^*_{t-1}}[I_{\{S(\boldsymbol{X})\geq m_t\}}] \tag{6}$$

and $c_0 = \ell_0 = \mathbb{E}_f[I_{\{S(\boldsymbol{X})\geq m_0\}}]$. Here $\{m_t,\ t = 0, 1, \ldots, T\}$ is a fixed grid satisfying $-\infty < m_0 < m_1 < \cdots < m_T = m$; f denotes the proposal pdf $f(\boldsymbol{x})$; and

$$g^*_{t-1} = g^*(\boldsymbol{x}, m_{t-1}) = \ell(m_{t-1})^{-1} f(\boldsymbol{x}) I_{\{S(\boldsymbol{x})\geq m_{t-1}\}}, \tag{7}$$

where $\ell(m_{t-1})^{-1}$ is the normalization constant. Observe that $g^*_{t-1} = g^*(\boldsymbol{x}, m_{t-1})$ is a *zero variance* importance sampling (IS) pdf at iteration $(t-1)$, but not at iteration t. Note that the part $\mathbb{E}_{g^*_{t-1}}[I_{\{S(\boldsymbol{X})\geq m_t\}}]$ of Eq. 6 is obtained directly by substituting Eq. 7 into c_t. Note finally that $c_0 = \ell_0 = \ell(m_0) = \mathbb{E}_f\left[I_{\{S(\boldsymbol{X})\geq m_0\}}\right]$ can be written in notations involving g^* as $c_0 = \frac{\ell_0}{\ell_{-1}} = \mathbb{E}_{g^*_{-1}}\left[I_{\{S(\boldsymbol{X})\geq m_0\}}\right]$, where $\ell_{-1} = \ell(m_{-1}) \equiv 1$ and $g^*_{-1} \equiv f$.

Thus, $\ell(m_t)$ can be written as the product of *conditional* expectations (probabilities)

$$\mathbb{E}_f[I_{\{S(\boldsymbol{X})\geq m_t\}}|I_{\{S(\boldsymbol{X})\geq m_{t-1}\}}],\ t=0,1,\ldots,T$$

under f, or as the product of the *unconditional* ones

$$c_t=\mathbb{E}_{g^*_{t-1}}[I_{\{S(\boldsymbol{X})\geq m_t\}}]$$

under g^*_{t-1}. Its estimator can be written in analogy to Eq. 2 as

$$\widehat{\ell}(m)=\prod_{t=0}^{T}\widehat{c}_t=\frac{1}{N^{T+1}}\prod_{t=0}^{T}N_t, \tag{8}$$

where

$$\widehat{c}_t=\frac{1}{N}\sum_{i=1}^{N}I_{\{S(\boldsymbol{X}_i)\geq m_t\}}=\frac{N_t}{N} \tag{9}$$

and $\boldsymbol{X}_i\sim g^*_{t-1}$.

Observe that

1. Although g^*_{t-1} is a zero variance pdf at level m_{t-1}, it is readily seen that *the estimator $\widehat{c}_t$ of $c_t=\mathbb{E}_{g^*_{t-1}}[I_{\{S(\boldsymbol{X})\geq m_t\}}]$ is not a zero variance estimator, since $m_{t-1}\neq m_t$.*
2. The estimator $\widehat{c}_t$ would be a zero variance one if we could replace the pdf g^*_{t-1} in $\mathbb{E}_{g^*_{t-1}}[I_{\{S(\boldsymbol{X})\geq m_t\}}]$ by g^*_t, that is at iteration t to sample from g^*_t.
3. If the proposal density $f(\boldsymbol{x})$ is *uniformly* distributed on the entire set $\mathcal{X}=\{\boldsymbol{x}:S(\boldsymbol{x})\geq m_{-1}\}$, (with m_{-1} typically being $m_{-1}=0$), then g^*_{t-1} is *uniformly* distributed on the reduced set $\mathcal{X}_t=\{\boldsymbol{x}:S(\boldsymbol{x})\geq m_t\}$. That is the main goal of g^*_{t-1} is to sample uniformly on the set $\mathcal{X}_t$, similar as the goal of $f(\boldsymbol{x})$ is to sample uniformly on $\mathcal{X}$.

In summary, when we say that g^*_{t-1} is a zero variance, or optimal IS pdf at the level m_{t-1}, we mean that it is uniformly distribution on the reduced set (space) $\mathcal{X}_t$. It is crucial to note that *the sequence $\{\widehat{c}_t\}$ and the associated estimators $\widehat{\ell}$ in the product form (8) will be required only for rare events and for some particular cases of counting problems, but it will be never required in optimization.*

It follows from Eq. 8, that in order for the estimator $\widehat{\ell}(m)$ to be useful, the levels m_t should be chosen such that each conditional probability $c_t=\mathbb{E}_f[I_{\{S(\boldsymbol{X})\geq m_t\}}|I_{\{S(\boldsymbol{X})\geq m_{t-1}\}}]$ is not too small, say approximately equal to 10^{-2}. Note only we assume that the levels m_t are chosen such that each c_t is not too small, but we shall also require *existence of at least one sequence* $\{m_0,m_1,\ldots,m_T=m\}$, which insures that all $c_t,\ t=0,1,\ldots,T$ values are not too small.

In both counting and optimization problems we shall generate an adaptive sequence of tuples

$$\{(m_0,g^*(\boldsymbol{x},m_{-1})),\ (m_1,g^*(\boldsymbol{x},m_0)),\ (m_2,g^*(\boldsymbol{x},m_1)),\ldots,(m_T,g^*(\boldsymbol{x},m_{T-1}))\}, \tag{10}$$

where as before $g^*(\boldsymbol{x},m_{-1})=f(\boldsymbol{x})$. This is in contrast to CE where we generate a sequence of tuples

$$\{(m_0,\boldsymbol{v}_0),\ (m_1,\boldsymbol{v}_1),\ldots,(m_T,\boldsymbol{v}_T)\}, \tag{11}$$

where $\{\boldsymbol{v}_t,\ t = 1, \ldots, T\}$ is a sequence of parameters in the parametric family of fixed distributions $f(\boldsymbol{x}, \boldsymbol{v}_t)$. The crucial difference is, of course, that in our approach, $\{g^*(\boldsymbol{x}, m_{t-1}) = g^*_{t-1},\ t = 0, 1, \ldots, T\}$ *is a sequence of non-parametric IS distributions, rather than is a sequence of parametric IS ones* $\{f(\boldsymbol{x}, \boldsymbol{v}_t),\ t = 1, \ldots, T\}$. Otherwise the CE and the cloning algorithms are the same, beside the fact that, as we shall see below, CE has the desirable property that the samples are independent, whereas in cloning they are not. Note that a sequence of tuples with non-parametric updating like in Eq. 10, was earlier introduced in the MinxEnt method (see Rubinstein 2008a, b). The main problem with the MinxEnt method is that it is extremely difficult to generate sample from its non-parametric pdfs. So, eventually one needs to sample from the corresponding marginals distributions, which is equivalent to using a sequence of tuples similar to Eq. 11. Similar to DHR and BK methods (and in contrast to MinxEnt), we will be able to sample here from the joint IS pdf $g^*(\boldsymbol{x}, m_{t-1}) = g^*_{t-1}$ using the MCMC machinery and in particular the Gibbs sampler. By doing so we will update the parameters c_t and m_t adaptively. Finally, we will show numerically that the cloning method typically outperforms its CE counterpart, especially for rare-event estimation and counting. The main reason is that sampling from a sequence of pdfs g^*_{t-1} (or even from their approximations) is more beneficial than sampling from sequence of a parametric family, like $f(\boldsymbol{x}, \boldsymbol{v}_t)$. In other words the sequence (10) is more informative than the one in Eq. 11.

As mentioned, the chain rule approach (1), (4) has been extensively used in randomized algorithms (Mitzenmacher and Upfal 2005; Motwani and Raghavan 1997) for estimating counting quantities associated with some graphs. Their sampling mechanism is, however, completely different from ours.

For a quick glance at our sampling mechanism consider

$$\ell = \mathbb{E}_f \left[I_{\{\sum_{i=1}^n X_i \geq m\}} \right],$$

where all X_i's are iid Ber(1/2) random variables. Assume for concreteness that we want to count the number of outcomes on the set $\mathcal{X}^* = \{x : \sum_{i=1}^n X_i \geq m\}$. Let for simplicity $n = m = 3$. Although it is obvious that the cardinality $|\mathcal{X}^*| = 1$, as mentioned our goal is to demonstrate the sampling mechanism.

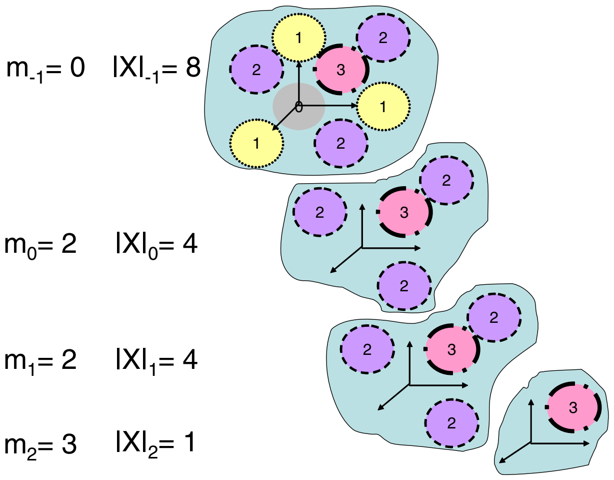
Figure 1 presents a possible dynamic of the evolution of the sequence

$$\{(m_{-1}, |\mathcal{X}_{-1}|),\ (m_0, |\mathcal{X}_0|), \ldots, (m, |\mathcal{X}_m|)\},$$

where m_t and $|\mathcal{X}_t|$ denote the level reached and cardinality (the number of points) of the corresponding sample space at each iteration t, respectively. Clearly $(m_{-1}, |\mathcal{X}_{-1}|) = (0,\ 2^3 = 8)$.

According to Fig. 1 we obtain $m_0 = 2$ after the first iteration, which means that while flipping three symmetric coins $\sum_{i=1}^3 X_i = m_0 = 2$, (two coins resulted to one and one coin resulted to 0). As soon as we obtain $m_0 = 2$ we reduce the original sample space $\mathcal{X}_{-1}$ containing eight points to the one $\mathcal{X}_0$ containing four points. This is done by eliminating 4 outcomes corresponding to events $\left\{ \sum_{i=1}^3 X_i = 0 \right\}$ and $\left\{ \sum_{i=1}^3 X_i = 1 \right\}$ from the space $\mathcal{X}_{-1} = \left\{ \boldsymbol{X} : \sum_{i=1}^3 X_i \geq 0 \right\}$. In other words, as soon as we obtain an outcome, such that $\sum_{i=1}^3 X_i = 2$ we truncate the sample space $\mathcal{X}_{-1}$ by excluding from it all points corresponding to the event $\left\{ \sum_{i=1}^3 X_i \leq 1 \right\}$. We proceed to the next iteration by sampling from the IS pdf $g^*_0 = g^*(\boldsymbol{x}, m_0)$ (see Eq. 7), which is proportional to $I_{\left\{ \sum_{i=1}^3 X_i \geq 2 \right\}}$, that is we sample uniformly on the reduced space

Fig. 1 Three-dimensional balls



$\mathcal{X}_0 = \{\boldsymbol{X} : \sum_{i=1}^{3} X_i \geq 2\}$, which contains now four points (three points corresponding to the outcome $\sum_{i=1}^{3} X_i = 2$ and one point corresponding to the outcome $\sum_{i=1}^{3} X_i = 3$). Following Fig. 1 we see that at the second iteration we obtain $m_1 = 2$. This means that in the reduced space $\mathcal{X}_0$ the outcome is again as $\sum_{i=1}^{3} X_i = 2$. Clearly, that $|\mathcal{X}_1| = |\mathcal{X}_0| = 4$ and $g^*(\boldsymbol{x}, m_0) = g^*(\boldsymbol{x}, m_1)$. Proceeding to the third (final) iteration we see from Fig. 1 that $m_2 = 3$ and $|\mathcal{X}_2| = 1$, that is we converged to the desired quantities $m = 3$ and $|\mathcal{X}^*| = 1$. Observe that in this case $g^*(\boldsymbol{x}, m_2)$ corresponds to a degenerate pdf, that is all of its mass is concentrated at point $\boldsymbol{X} = (1, 1, 1)$.

As we shall see below, one of the main challenges of this work will be to obtain a sequence of reduced (nested) spaces $\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_T = \mathcal{X}^*$ starting from the original one $\mathcal{X}_{-1}$ and then to generate points uniformly distributed on each set $\mathcal{X}_t,\ t = 0, 1, \ldots, T$.

The rest of our paper is organized as follows. Section 2 deals with the method of Diaconis-Holmes-Ross, from which we adopted some basic ideas, while Section 3 deals with the method of Botev–Kroese, from which our motivation and inspiration comes. Section 4 introduces the cloning mechanism, which essential for our optimization and counting algorithms. Sections 5 and 6 are the main ones. In particular, the former presents the main cloning algorithm for unconstrained and constrained combinatorial optimization including TSP, while the latter presents the main cloning algorithm for rare events and counting, also called the *Gibbs cloner*. In addition, we present in Section 6 application of the cloning algorithm for counting multiple events, like counting the number of feasible solution on the sets associated with linear integer programs and counting the number of satisfiability assignments in a SAT problem. In particular, consider a set containing both equality and inequality constraints of an integer program, that is

$$\begin{aligned}
&\sum_{k=1}^{n} a_{ik} x_k = b_i,\ i = 1, \ldots, m_1, \\
&\sum_{k=1}^{n} a_{jk} x_k \geq b_j,\ j = m_1 + 1, \ldots, m_1 + m_2, \\
&\boldsymbol{x} \geq 0,\quad x_k \text{ integer } \forall k = 1, \ldots, n.
\end{aligned} \tag{12}$$

It is shown in Section 6 that in order to count the number of points (feasible solutions) of the set (12) one can consider the following associated rare-event probability problem

$$\ell = \mathbb{E}_{\boldsymbol{u}}\left[I_{\{\sum_{i=1}^{m} C_i(\boldsymbol{X}) \geq m\}} \right], \tag{13}$$

where the first m_1 terms $C_i(\boldsymbol{X})$'s in Eq. 13 are

$$C_i(\boldsymbol{X}) = I_{\{\sum_{k=1}^{n} a_{ik} X_k = b_i\}}, \ i = 1, \ldots, m_1, \tag{14}$$

while the remaining m_2 ones are

$$C_i(\boldsymbol{X}) = I_{\{\sum_{k=1}^{n} a_{ik} X_k \geq b_i\}}, \ i = m_1 + 1, \ldots, m_1 + m_2. \tag{15}$$

Thus, in order to count the number of feasible solution on the set (12) we shall consider an associated rare event probability estimation problem (13), involving a *sum of dependent Bernoulli random variables*. We shall see below that representation (13) is crucial for a large set of counting problems. A particular emphasis in Section 6 will be paid to a modified version of the cloning algorithm involving, what we call a *direct estimator* for counting. We shall show that the direct estimator is extremely useful and more accurate than the one in Eq. 8 based on the products of $\widehat{c}_t$'s in Eq. 9. Finally, we introduce in Section 6 a cloning algorithm to handle counting problems associated with the Boltzmann distribution. We call this algorithm, the *Boltzmann cloning algorithm*. We show that it is easy to switch from the Boltzmann cloning algorithm to the main cloning one and vise-versa. This, for example, means that the classic Ising model can be treated by using the main cloning algorithm rather than via the Boltzmann cloning one. Section 7 discusses how to sample uniformly on different convex regions with the Gibbs cloner. Section 8 presents a combined version of the cloning and the cross-entropy (CE) algorithms, which we call the *CLONCE* algorithm and discuss the range of it possible applications, in particular to combinatorial optimization. In addition to the main cloning algorithm it contains one more step involving CE for an adaptive choice of the parameter vector in the IS pdf. In Section 9 we present some possible applications of the cloning method, while in Section 10 we prove the polynomial complexity of a particular version of the Gibbs cloner for counting. Section 11 presents some numerical results with the cloning algorithms for combinatorial and integer optimization and counting. Finally, in Section 12 conclusions and some final remarks are given.

2 The Method of Diaconis–Holmes–Ross for Counting

In the method of Diaconis and Holmes (1994) and Ross (2002a, b), called *Diaconis–Holmes–Ross* (DHR) method assume the set of levels $\{m_t, \ t = 0, \ldots, T\}$ is given in advance. Each quantity $\mathbb{E}_f[I_{\{S(\boldsymbol{X}) \geq m_t\}} | I_{\{S(\boldsymbol{X}) \geq m_{t-1}\}}]$ is estimated *separately and independently* by using the MCMC, in particular, the Gibbs sampler. Ross (2002a, b) presents several interesting applications using Gibbs sampler. His original algorithm for estimating c_t is recapitulated for convenience in the Appendix.

The main idea of DHR algorithm, while estimating each conditional probability $\mathbb{E}_f[I_{\{S(\boldsymbol{X}) \geq m_t\}} | I_{\{S(\boldsymbol{X}) \geq m_{t-1}\}}]$ is to run *Markov Chain Monte Carlo* (MCMC) and count the proportion of values satisfying $\{S(\boldsymbol{X}) \geq m_t\}$.

More formally, their algorithm can be written as

Algorithm 2.1 (DHR Algorithm)

Given a sequence of levels $m_0 < m_1 < \ldots < m_T = m$ and the sample size N, execute the following steps

1. **Acceptance–Rejection** Set a counter $t = 1$. Initialize by generating a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from the proposal density $f(\boldsymbol{x})$. Let $\widetilde{\mathcal{X}}_0 = \{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_0}\}$ be the largest subset of the population $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ (elite samples) for which $S(\boldsymbol{X}_i) \geq m_0$. Note that $\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_0} \sim g^*(\boldsymbol{x}, m_0)$ and that
$$\widehat{c}_0 = \widehat{\ell}(m_0) = \frac{1}{N}\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i)\geq m_0\}} = \frac{N_0}{N} \tag{16}$$
is an *unbiased* estimator of $\ell(m_0)$.
2. **MCMC step**. Find a feasible point $\boldsymbol{X}$ such that $S(\boldsymbol{X}) \geq m_{t-1}$. Starting from $\boldsymbol{X}$, run the MCMC sampler, such that after some *burn-in* period, each vector $\boldsymbol{X} = (X_1, \ldots, X_n)$, of the *new* population denoted as $\mathcal{X} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ is *approximately* distributed as $g^*(\boldsymbol{x}, m_{t-1})$.
3. **Estimating** c_t Let $\widetilde{\mathcal{X}}_t = \{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_t}\}$ be the subset of the population $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ for which $S(\boldsymbol{X}_i) \geq m_t$. Take $\widehat{c}_t$ in Eq. 9 is an estimator of c_t in Eq. 6. Note that $\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_t}$ is distributed *approximately* $g^*(\boldsymbol{x}, m_t)$. Note also that as a feasible point $\boldsymbol{X}$ satisfying $S(\boldsymbol{X}) \geq m_t$ one can take, for example, any point from the subset $\{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_t}\}$
4. **Stopping rule** If $t = T$ go to step 5, otherwise set, set $t = t + 1$ and repeat from step 2.
5. **Final Estimator** Deliver $\widehat{\ell}(m)$ in Eq. 8 as an estimator of $\ell(m)$.

For convenience we also present below the basic versions of the Gibbs sampler.

2.1 The Gibbs Sampler

There are two basic version of the Gibbs sampler (Rubinstein and Kroese 2007): *systematic* and *random*. In the former one the components of the vector $\boldsymbol{X} = (X_1, \ldots, X_n)$ are updated in a fixed, say increasing order: $1, 2, \ldots, n, 1, 2, \ldots$ while in the latter, they are chosen randomly, that is according to a discrete uniform n-point pdf. Below we present the systematic Gibbs sampler algorithm. In a systematic Gibbs sampler, for a given vector $\boldsymbol{X} = (X_1, \ldots, X_n) \sim g(\boldsymbol{x})$, one generates a *new* vector $\widetilde{\boldsymbol{X}} = (\widetilde{X}_1, \ldots, \widetilde{X}_n)$ with the same distribution $\sim g(\boldsymbol{x})$ as follows:

Algorithm 2.2 (Gibbs Sampler)

1. Draw $\widetilde{X}_1$ from the conditional pdf $g(x_1|X_2, \ldots, X_n)$.
2. Draw $\widetilde{X}_i$ from the conditional pdf $g(x_i|\widetilde{X}_1, \ldots, \widetilde{X}_{i-1}, X_{i+1}, \ldots, X_n)$, $i = 2, \ldots, n-1$.
3. Draw $\widetilde{X}_n$ from the conditional pdf $g(x_n|\widetilde{X}_1, \ldots, \widetilde{X}_{n-1})$.

Note that in Gibbs sampler it is assumed that generating samples from the conditional pdfs $g(x_i|X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n)$, $i = 1, \ldots, n$ is simple.

Example 2.1 Sum of Independent Random Variables

Consider application of the Gibbs sampler to estimate ℓ with $S(\boldsymbol{x}) = \sum_{i=1}^{n} X_i$, that is

$$\ell = \mathbb{E}_f \left[I_{\{\sum_{i=1}^{n} X_i \geq m\}} \right]. \tag{17}$$

In this case generating random variables $X_i,\ i = 1, \ldots, N$ for a fixed value m can be easily performed by using the Gibbs sampler based on the following conditional pdf

$$g(x_i, m|\boldsymbol{x}_{-i}) = r_i(m)\, f_i(x_i) I_{\left\{x_i \geq m - \sum_{j\neq i} x_j\right\}}, \tag{18}$$

where $|\boldsymbol{x}_{-i}$ denotes conditioning on all random variables but *excluding* the i-th component and $r_i(m)$ denotes the normalization constant.

Note also that each of the n conditional pdfs $g(x_i, m|\boldsymbol{x}_{-i})$ presents a truncated version of the proposal marginal pdf $f_i(x_i)$ with the truncation point at $m - \sum_{j\neq i} x_j$. In short, the random variable $\widetilde{X}$ from $g(x_i, m|\boldsymbol{x}_{-i})$ presents a shifted original random variable $X \sim f_i(x_i)$. Generation from such truncated single dimensional pdf $g(x_i, m|\boldsymbol{x}_{-i})$ is easy and can be typically performed by using the inverse-transform method, provided the inverse-transform method can be applied to $f_i(x_i)$.

For example, sampling a Ber(p) random variable $\widetilde{X}_i$ from truncated pdf (18) can be performed as follows:

Generate $Y \sim$ Ber(p). If $Y = 1$ and

$$Y \geq m - \sum_{j\neq i} X_j,$$

set $\widetilde{X}_i = 1$, otherwise set $\widetilde{X}_i = 0$.

Remark 2.1 It is important to note that the above Bernoulli model is crucial for the rest of this paper, since we shall show that a large set of binary integer optimization and counting problems can be

1. Reduced to estimation of the rare event probability (13), that is to $\ell = \mathbb{E}_f \left[I_{\{\sum_{i=1}^{m} C_i(\boldsymbol{X}) \geq m\}} \right]$, where the C_i's are *dependent* Bernoulli random variables.
2. The complexity properties for such associated counting problems can be treated via the rare event probability $\ell = \mathbb{E}_f \left[I_{\{\sum_{i=1}^{n} X_i \geq m\}} \right]$, where all X_i's are distributed Bernoulli with *independent* components.

Example 2.2 Sum of Bernoulli Random Variables in Pairs

Consider application of Gibbs sampler for estimating

$$\ell = \mathbb{E}_f \left[I_{\left\{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} X_i Y_j \geq m\right\}} \right], \tag{19}$$

where X_i and Y_j are each iid distributed Ber$(1/2)$.

In this case we write ℓ with respect to each coordinate X_k and Y_k as

$$\ell = \mathbb{E}_f \left[I_{\left\{\sum_{i=1}^{n} X_i \sum_{j=1}^{n} a_{ij} Y_j \geq m\right\}} \right] = \mathbb{E}_f \left[I_{\left\{X_k \geq \frac{m - \sum_{i\neq k}^{n} \sum_{j=1}^{n} a_{ij} X_i Y_j}{\sum_{j=1}^{n} a_{kj} Y_j}\right\}} \right] \tag{20}$$

and

$$\ell = \mathbb{E}_f\left[I_{\left\{\sum_{j=1}^n Y_j \sum_{i=1}^n a_{ij} X_i \geq m\right\}}\right] = \mathbb{E}_f\left[I_{\left\{Y_k \geq \frac{m - \sum_{i=1}^n \sum_{j\neq k}^n a_{ij} X_i Y_j}{\sum_{i=1}^n a_{ki} X_i}\right\}}\right], \tag{21}$$

respectively. In this case, in analogy to Eq. 18, for a for a fixed value m and a fixed tuple vector $(\boldsymbol{X}, \boldsymbol{Y}) = (X_1, \ldots, X_n,\ Y_1, \ldots, Y_n) \sim g(\boldsymbol{x})$, a *new* tuple vector $(\widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{Y}}) = (\widetilde{X}_1, \ldots, \widetilde{X}_n,\ \widetilde{Y}_1, \ldots, \widetilde{Y}_n)$ with the same distribution $g(\boldsymbol{x})$ can be generated using the *systematic* Gibbs sampler based on the following conditional pdfs

$$g(x_k, m|\boldsymbol{x}_{-k}, \boldsymbol{y}) = c_{x_k}(m)\, f_k(x_k)\left[I_{\left\{x_k \geq \frac{m - \sum_{i\neq k}\sum_{j=1}^n a_{ij} x_i y_j}{\sum_{j=1}^n a_{kj} y_j}\right\}}\right], \tag{22}$$

and

$$g(y_k, m|\boldsymbol{y}_{-k}, \boldsymbol{x}) = c_{y_k}(m)\, f_k(y_k)\left[I_{\left\{y_k \geq \frac{m - \sum_{i=1}^n \sum_{j\neq k} a_{ij} x_i y_j}{\sum_{i=1}^n a_{ki} x_i}\right\}}\right], \tag{23}$$

respectively. Here as before $|\boldsymbol{x}_{-k}, \boldsymbol{y}$ denotes conditioning on all random variables $\boldsymbol{x}$ but excluding its k-th component and $c_{x_k}(m)$ denotes the normalization constant and similar $|\,\boldsymbol{y}_{-k}, \boldsymbol{x}$.

Note that when $a_{ij} = a_i b_j$, we obtain

$$\ell = \mathbb{E}_f\left[I_{\left\{X_k \geq \frac{m - \left(\sum_{i\neq k} a_i X_i\right)\left(\sum_{j=1}^n b_j Y_j\right)}{a_k \sum_{j=1}^n b_j Y_j}\right\}}\right] \tag{24}$$

and

$$\ell = \mathbb{E}_f\left[I_{\left\{Y_k \geq \frac{m - (\sum_{j\neq k} b_j Y_j)(\sum_{i=1}^n a_i X_i)}{b_k \sum_{i=1}^n a_i X_i}\right\}}\right], \tag{25}$$

respectively. In this case the conditional pdfs $g(x_k, m|\boldsymbol{x}_{-k}, \boldsymbol{y})$ and $g(y_k, m|\,\boldsymbol{y}_{-k}, \boldsymbol{x})$ can be written similar to Eqs. 22 and 23, respectively.

3 The Method of Botev and Kroese for Counting

The main drawback of DHR (Diaconis and Holmes 1994) Algorithm 2.1 is that for each fixed level m_t it starts basically from scratch. That is, at each iteration, starting at some feasible point $\boldsymbol{x}$, it runs a *single* Markov chain before the entire sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ becomes distributed approximately stationary, that is approximately $g^*_{t-1}(\boldsymbol{x}, m_{t-1})$. The time to reach the stationarity (the burn-in period) might be quite long, however. Note, finally that since DHR Algorithm 2.1 always starts with a single (elite) sample and the samples between different levels are independent, it requires not only quite a long burn-in period for the samples $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ to become at approximately stationary, but as a result of that the $\widehat{c}_t$'s are typically biased estimators of their true parameters c_t's.

To overcome this difficulty, Botev and Kroese (2008) introduced several important enhancements into the DHR Algorithm 2.1.

The main enhancement of Botev and Kroese (2008) is that their algorithm has an additional step, called the *bootstrap resampling step*. It reuses iteratively all the elite samples $\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_t} \sim g^*(\boldsymbol{x}, m_t)$ from the previous Markov chain runs and, thus it runs many Markov chains in parallel. By doing so, the elite samples at different levels become dependent, but the stationarity in terms of sampling from the optimal importance sampling pdf $g^*_{t-1}(\boldsymbol{x}, m_t)$ is preserved. To define the level sets $\{\widehat{m}_t\}_{t=0}^T$, Botev and Kroese make an additional (pilot) run. Note that in Botev and Kroese the level sets $\{\widehat{m}_t\}_{t=0}^T$ are defined adaptively, while in the DHR Algorithm 2.1 they are assumed to be fixed in advance. Note that the level sets $\{\widehat{m}_t\}_{t=0}^T$ in the latter case are chosen similarly to the CE method, in the sense that they involve a rarity parameter ρ. The adaptive choice of $\widehat{m}_t$ seems to be more natural and more flexible than the fixed one.

It is crucial to note that in contrast to the CE and the MinxEnt algorithms (Rubinstein and Kroese 2007) both algorithms, Botev–Kroese (Botev and Kroese 2008) and DHR (Diaconis and Holmes 1994) possesses the following properties:.

1. Sample from the non-parametric IS distribution $g^*(\boldsymbol{x}, \widehat{m}_t)$ rather than from the parametric one $f(\boldsymbol{x}, \widehat{\boldsymbol{p}}_t)$. The latter is associated with the original (proposal) pdf $f(\boldsymbol{x}, \boldsymbol{u})$.
2. Do not involve any optimization procedure, like the MinxEnt program, which minimizes the Kulback–Liebler divergence, subject to some constraints. They are based solely on the samples from $g^*(\boldsymbol{x}, \widehat{m}_t)$, $t = 0, 1, \ldots, T$, or their approximations.

Before presenting Botev–Kroese algorithm we summarize its main features.

- It requires a pilot run to define a sequence $\{\widehat{m}_t\}$ such that $\widehat{m}_0 < \widehat{m}_1 < \ldots < \widehat{m}_T = m$.
- It samples recursively from the sequence of IS pdfs: $\{g^*_{t-1}\} = \{g^*(\boldsymbol{x}, \widehat{m}_{t-1})\}$. The recursive process of sampling from $g^*(\boldsymbol{x}, m_t)$ is continued until eventually the level m is reached and, thus one can generate from the desired IS pdf $g^*(\boldsymbol{x}) = g^*(\boldsymbol{x}, m)$.
- The exact sampling from $g^*_0 = g^*(\boldsymbol{x}, \widehat{m}_0)$ is obtained from the original distribution $f(\boldsymbol{x})$ by using the acceptance–rejection method (with the acceptance probability ρ), that is coincides with step 1 of the DHR Algorithm 2.1. The goal of the sample from g^*_0 (or from an associated kernel density approximation based on that sample) is to help generating exact samples at the next iteration from g^*_1.
- The estimators of c_t are dependent and thus the entire estimator of $\ell(m)$ given in Eq. 8, which is based on the product of dependent random variables $\widehat{c}_t$ is biased.

The resulting Botev–Kroese (Botev and Kroese 2008) algorithm for rare events estimation can be written as

Algorithm 3.1 (Botev–Kroese Algorithm for Rare Events)

Given a sequence of levels $\widehat{m}_0 < \widehat{m}_1 < \ldots < \widehat{m}_T = m$ and the sample size N, execute the following steps

1. **Acceptance–Rejection** Set a counter $t = 1$. Initialize by generating a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from the proposal density $f(\boldsymbol{x})$. Let $\widetilde{\mathcal{X}}_0 = \{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_0}\}$ be the largest subset of the population $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ (elite samples) for which $S(\boldsymbol{X}_i) \geq \widehat{m}_0$. Note that $\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_0} \sim g^*(\boldsymbol{x}, \widehat{m}_0)$ and that $\widehat{\ell}(\widehat{m}_0)$ in Eq. 16 is an *unbiased* estimator of $\ell(m_0)$.
2. **Bootstrap step**. Sample *uniformly with replacement* N times from the population $\widetilde{\mathcal{X}}_{t-1} = \{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_{t-1}}\}$ to obtain a new (bootstrap) population $\{\boldsymbol{X}^*_1, \ldots, \boldsymbol{X}^*_N\}$. Note that $\boldsymbol{X}^*_1, \ldots, \boldsymbol{X}^*_N \sim g^*(\boldsymbol{x}, \widehat{m}_{t-1})$.
3. **MCMC step**. For each vector $\boldsymbol{X}^* = (X^*_1, \ldots, X^*_n)$ of the population $\{\boldsymbol{X}^*_1, \ldots, \boldsymbol{X}^*_N\}$ generate, say by using the Gibbs sampler, a new vector $\widetilde{\boldsymbol{X}}^* = (\widetilde{X}^*_1, \ldots, \widetilde{X}^*_n)$. Note that the *new* population $\{\widetilde{\boldsymbol{X}}^*_1, \ldots, \widetilde{\boldsymbol{X}}^*_N\}$ of $\widetilde{\boldsymbol{X}}^*$'s is distributed again as $g^*(\boldsymbol{x}, \widehat{m}_{t-1})$. Denote the new population thus obtained by $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$.
4. **Estimating** c_t Let $\widetilde{\mathcal{X}}_t = \{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_t}\}$ be the subset of the population $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ for which $S(\boldsymbol{X}_i) \geq \widehat{m}_t$. Take $\widehat{c}_t$ in Eq. 9 as an estimator of c_t given in Eq. 6. Note again as that $\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_t}$ is distributed $g^*(\boldsymbol{x}, \widehat{m}_t)$.
5. **Stopping rule** If $t = T$ go to step 6, otherwise set, set $t = t + 1$ and repeat from step 2.
6. **Final Estimator** Deliver $\widehat{\ell}(m)$ given in Eq. 8 as an estimator of $\ell(m)$.

The pilot run algorithm of Botev–Kroese for selection of the levels m_t, $t = 1, \ldots, T$ can be find in (Botev and Kroese 2008). Note that an earlier version of Algorithm 3.1 contained no pilot run.

Remark 3.1 **Reducing Dependency** Note that although the **MCMC step** in Algorithm 3.1 assumes a burn-in period $= 1$ at each iteration, in practice (Botev and Kroese 2008) suggest using, a burn-in period ≥ 1. By doing so one can reduce the dependence between the vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ at different iterations t. In their modified MCMC step one

1. Generates at the **MCMC step** instead of $\widetilde{\boldsymbol{X}}^*_1, \ldots, \widetilde{\boldsymbol{X}}^*_N$ a larger new population, namely $\{\widetilde{\boldsymbol{X}}^*_1, \ldots, \widetilde{\boldsymbol{X}}^*_{rN}\}$, $r > 1$.
2. Takes only the last N samples from $\{\widetilde{\boldsymbol{X}}^*_1, \ldots, \widetilde{\boldsymbol{X}}^*_{rN}\}$ (discarding the first $(r-1)N$ ones) and denote these, as before, by $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$.

It is not difficult to see that if the size of the elite sample $\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_t}$ equals 1 at every iteration t, and if these single elites are independent for all t, $(t = 1, \ldots, T)$, then we automatically obtain the DHR Algorithm 2.1.

The rest of this paper deals with the cloning algorithms and their applications to optimization, counting, and uniform sampling on different sets. As mentioned the proposed cloning algorithms adopt the best features of the DHR Algorithm 2.1 and

BK Algorithm 3.1. *They can be viewed as enhancements of both*. Below we mention several major enhancements introduced in the cloning algorithms:

- A new mechanism, called the *cloning*.
- A new *screening* step.
- An adaptive choice of ρ.
- A combined algorithm based on CE and cloning.
- A new estimator, called the *direct estimator*, for counting.

Recall again as that neither pilot run no bootstrap step is used in the proposed algorithms.

Before presenting the cloning algorithms we introduce the cloning mechanism.

4 The Cloning Mechanism

The goal of the cloning mechanism is to find a good balance in the Gibbs sampler, in terms of bias-variance, between the number of elite samples and the length of the burn-in period, denoted by b. Note that DHR Algorithm 2.1 and Botev–Kroese Algorithm 3.1 correspond to $b = N$ and $b = 1$, respectively.

For fixed N and b, we can define at iteration $(t-1)$, for example, the following adaptive *cloning* parameter η_{t-1}

$$\eta_{t-1} = \left\lceil \frac{N}{bN_{t-1}} \right\rceil - 1 = \left\lceil \frac{N_{cl}}{N_{t-1}} \right\rceil - 1. \tag{26}$$

Here $N_{cl} = N/b$ is called the *cloned sample size* and, as before, N_{t-1} denotes the number of elites at iteration $t-1$. Note that $\lceil \cdot \rceil$ denotes rounding to the largest integer. The goal of η_{t-1} is to reproduce η_{t-1} times the N_{t-1} elites.

As an example, let $N = 1{,}000$, $b = 10$. Consider two cases: $N_{t-1} = 21$ and $N_{t-1} = 121$. We obtain $\eta_{t-1} = 4$ and $\eta_{t-1} = 0$ (no cloning).

Our numerical studies show that it is quite reasonable to choose $1 \le b \le 3$ to have manageable bias-variance balance. In this case, the Gibbs sampler is applied b times to each vector $\boldsymbol{X}$ of the cloned samples of size $N_{cl} = b^{-1}N$.

As for an alternative to Eq. 26 one can use the following strategy in defining b and η: find b_{t-1} and η_{t-1} from $b_{t-1}\eta_{t-1} \approx \frac{N}{N_{t-1}}$ and take $b_{t-1} \approx \eta_{t-1}$. In short, one can take

$$b_{t-1} \approx \eta_{t-1} \approx \left(\frac{N}{N_{t-1}} \right)^{1/2}. \tag{27}$$

Consider again as the two cases: $N_{t-1} = 21$ and $N_{t-1} = 121$ and let as before $N = 200$. We have $b_{t-1} \approx \eta_{t-1} = 3$ and $b_{t-1} \approx \eta_{t-1} = 1$, respectively.

With this at hand we now introduce the cloning step, which will be essential in all the cloning algorithm below.

Cloning step Given the size N_{t-1} of elite samples at iteration $(t-1)$, find the cloning and the burn-in parameters η_{t-1} and b_{t-1} either according Eq. 26 or according to Eq. 27. Reproduce each vector $\widetilde{\boldsymbol{X}}_k = (\widetilde{X}_{1k}, \ldots, \widetilde{X}_{nk})$ of the elite sample $\{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_{t-1}}\}$ η_{t-1} times, that is, take η_{t-1} identical copies of each vector $\widetilde{\boldsymbol{X}}_k$ obtained at the $(t-1)$-th iteration. Denote the entire new population ($\eta_{t-1}N_{t-1}$ cloned vectors plus the original screened elite sample $\{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_{t-1}}\}$) by $\mathcal{X}_{cl} =$

$\{(\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_1), \ldots, (\widetilde{\boldsymbol{X}}_{N_{t-1}}, \ldots, \widetilde{\boldsymbol{X}}_{N_{t-1}})\}$. To each of the cloned vectors of the population $\mathcal{X}_{cl}$ apply the MCMC (and in particular the Gibbs sampler) for b_{t-1} burn-in periods. Denote the *new entire* population by $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$. Observe that each member of $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ is distributed approximately $g^*(\boldsymbol{x}, \widehat{m}_{t-1})$.

5 The Cloning Algorithm for Combinatorial and Integer Programming

It shown in Rubinstein and Kroese (2004) and Rubinstein and Kroese (2007) that many interesting combinatorial optimization problems including the maximum cut, TSP and scheduling can be reduced to equivalent unconstrained ones and treated efficiently by the CE and MinxEnt methods by employing an efficient trajectory generation mechanism (Rubinstein and Kroese 2004). When this is not feasible the penalty function approach can be used (Rubinstein and Kroese 2004).

The goal of this section is to introduce the cloning algorithm for optimization as an efficient counterpart to the standard CE and MinxEnt algorithms.

5.1 Unconstrained Optimization

As for unconstrained problem we consider here the maximal cut and the TSP.

The Max-Cut Problem The maximal cut or *max-cut* problem in a graph can be formulated as follows. Given a graph $G = G(V, E)$ with a set of nodes $V = \{1, \ldots, n\}$ and a set of edges E between the nodes, partition the nodes of the graph into two arbitrary subsets V_1 and V_2 such that the sum of the weights (costs) c_{ij} of the edges going from one subset to the other is maximized. Note that some of the c_{ij} may be 0 — indicating that there is, in fact, no edge from i to j.

A cut can be conveniently represented via its corresponding *cut vector* $\boldsymbol{x} = (x_1, \ldots, x_n)$, where $x_i = 1$ if node i belongs to same partition as 1, and 0 else. For each cut vector $\boldsymbol{x}$, let $\{V_1(\boldsymbol{x}), V_2(\boldsymbol{x})\}$ be the partition of V induced by $\boldsymbol{x}$, such that $V_1(\boldsymbol{x})$ contains the set of indices $\{i : x_i = 1\}$. If not stated otherwise we set $x_1 = 1 \in V_1$.

Let $\mathcal{X}$ be the set of all cut vectors $\boldsymbol{x} = (1, x_2, \ldots, x_n)$ and let $S(\boldsymbol{x})$ be the corresponding cost of the cut. Then,

$$S(\boldsymbol{x}) = \sum_{i \in V_1(\boldsymbol{x}),\ j \in V_2(\boldsymbol{x})} c_{ij}. \tag{28}$$

We shall assume below that the graph is *undirected*. Note that for a *directed* graph the cost of a cut $\{V_1, V_2\}$ includes both the cost of the edges from V_1 to V_2 and from V_2 to V_1. In this case the cost corresponding to a cut vector $\boldsymbol{x}$ is therefore

$$S(\boldsymbol{x}) = \sum_{i \in V_1(\boldsymbol{x}),\ j \in V_2(\boldsymbol{x})} c_{ij} + c_{ji}\ . \tag{29}$$

The use of the Gibbs sampler to generate cut vectors in a graph based on $\text{Ber}(1/2)$ is straight forward.

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

Let $\mathcal{X}$ be the set of all possible tours and let $S(\boldsymbol{x})$ the total length of tour $\boldsymbol{x} \in \mathcal{X}$. We can represent each tour via a *permutation* $\boldsymbol{x} = (x_1, \ldots, x_n)$ with $x_1 = 1$. Mathematically the TSP reads as

$$\min_{\boldsymbol{x}\in\mathcal{X}} S(\boldsymbol{x}) \quad = \quad \min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \sum_{i=1}^{n-1} c_{x_i, x_{i+1}} + c_{x_n, 1} \right\}. \tag{30}$$

To apply the Gibbs sampler for TSP we adopt the following heuristics introduced in Botev and Kroese (2008). Given a tour $\boldsymbol{x}$ of length m_t generated by the pdf $g(\boldsymbol{x}, m_t)$, the conditional Gibbs sampling updates the existing tour $\boldsymbol{x}$ to $\widetilde{\boldsymbol{x}}$, where $\widetilde{\boldsymbol{x}}$ is generated from $g(\widetilde{\boldsymbol{x}}, m_t)$ and where $\widetilde{\boldsymbol{x}}$ is the same as $\boldsymbol{x}$ with one exception that the cities x_i and x_j in $\widetilde{\boldsymbol{x}}$ are reversed. We accept the tour $\widetilde{\boldsymbol{x}}$ with probability $I_{\{S(\widetilde{X}) \le m_t\}}$, otherwise we leave the tour $\boldsymbol{x}$ the same. If $\widetilde{\boldsymbol{x}}$ is accepted we update the cost function $S(\boldsymbol{x})$ (for $j > i$) as follows

$$S(\widetilde{\boldsymbol{x}}) = S(\boldsymbol{x}) - c_{x_{i-1}, x_i} - c_{x_j, x_{j+1}} + c_{x_{i-1}, x_j} + c_{x_i, x_{j+1}}, \tag{31}$$

provided the graph is symmetric and similarly if it is not symmetric. The *2-opt* acceptance–rejection single move step can be summarized as follows.

1. Given a tour (permutation) $\boldsymbol{X} = (X_1, \ldots, X_n)$, draw a pair of indices (I, J) such that $I \neq J$ and both I and J are uniformly distributed on the integers $1, \ldots, n$.
2. Given $(I, J) = (i, j)$, generate the pair $(\widetilde{X}_i, \widetilde{X}_j)$ from the conditional bivariate pdf

$$g(\widetilde{x}_i, \widetilde{x}_j, m_t | \boldsymbol{x}_{-i,-j}) = c_{ij}(m_t)\, f_i(x_i)\, f_j(x_j)\, I_{\{S(\widetilde{X}) \le m_t\}}, \tag{32}$$

   where $(\widetilde{x}_i, \widetilde{x}_j) \in \{(x_i, x_j),\ (x_j, x_i)\}$, $\boldsymbol{x}_{-i,-j}$ is the same as $\boldsymbol{x}$ except that the elements i and j are reversed and $c_{ij}(m_t)$ is the normalization constant.
3. Set $X_i = \widetilde{X}_i$ and $X_j = \widetilde{X}_j$. Denote $\widetilde{\boldsymbol{X}} = (X_1, \ldots, \widetilde{X}_i, \ldots, \widetilde{X}_j, \ldots, X_n)$. Note again as that $\widetilde{\boldsymbol{X}}$ is the same as $\boldsymbol{X}$ except that the i-th and j-th positions are exchanged.

Note that sampling a pair $(\widetilde{X}_i, \widetilde{X}_j)$ from $g(\widetilde{x}_i, \widetilde{x}_j, m_t | \boldsymbol{x}_{-i,-j})$ in Eq. 32 with $j > i$ is performed as follows. Generate $Y \sim$ Ber(1/2). If $Y = 1$ and if $S(x_1, \ldots, \widetilde{x}_j, \ldots, \widetilde{x}_i, \ldots, x_n) \le m_t$, set $\widetilde{X}_i = \widetilde{x}_j$ and $\widetilde{X}_j = \widetilde{x}_i$, otherwise set $\widetilde{X}_i = \widetilde{x}_i$ and $\widetilde{X}_j = \widetilde{x}_j$.

We next present an alternative approach, where the Gibbs sampler can be easily implemented as well. It is based on the trajectory generation method for TSP described in Rubinstein and Kroese (2007) and recapitulated here. We start with the following simple example.

Consider the permutation $\{1, 2, 3, 4\}$ and assume for simplicity that the current tour is $\boldsymbol{x} = \{1, 2, 3, 4, 1\}$. To generate a new tour we start from city 1 and flip a three sided fair device. Assume that the outcome is the city 3, then our new permutation becomes $\{1, 3, 2, 4\}$. Next we continue from city 3 and flip a two sided fair device (coin) with possible movements to cities 2 and 4. Assume that the outcome is the city 4, so our next permutation becomes $\{1, 3, 4, 2\}$ and our final *new* tour is $\tilde{\boldsymbol{x}} = \{1, 3, 4, 2, 1\}$.

Implementing the Gibbs sampler for generating such TSP tours is easy. We present details.

1. Given an initial tour $\boldsymbol{x}_0 = (x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_n, x_1)$, draw an index J_0 uniformly distributed on the integers $(x_2, \ldots, x_i, x_{i+1}, \ldots, x_n)$ containing $n - 1$ points and, thus excluding the point (node) x_1.

2. Given j_0, apply acceptance–rejection until

$$S(x_0 - x_2,\ -j_0) \le m \tag{33}$$

holds, where $x_0 - x_2, -j_0$ is the same as the initial tour $\boldsymbol{x}_0$ except that the elements x_2 and j_0 are reversed. Denote $x_0 - x_2, -j_0$ by $\boldsymbol{x}_1$. Note that $S(\boldsymbol{x}_1)$ is defined similar to Eq. 31.
3. Given the tour $\boldsymbol{x}_1 = (x_1, \tilde{x}_{j_0}, \ldots, \tilde{x}_2, x_i, x_{i+1}, \ldots, x_n, x_1)$, draw an index J_1 uniformly distributed on the remaining $n-3$ points, which exclude the points x_1 and j_0.
4. Given j_1, apply acceptance–rejection until

$$S(\widetilde{x}_1 - j_0, -j_1) \le m \tag{34}$$

holds, where $\widetilde{x}_1 - j_0, -j_1$ is the same as $\boldsymbol{x}_1$ except that the elements x_3 and j_1 are reversed. Denote $\widetilde{x}_1 - j_0, -j_1$ by $\boldsymbol{x}_2$. Note again that $S(\boldsymbol{x}_2)$ is defined similar to Eq. 31.
5. Continue with the steps 3 and 4 for a total of $n-2$ rounds, that is until the corresponding random variable J_{n-2} becomes a degenerated one. By doing so we shall loop automatically from the corresponding city $\tilde{x}_{j_{n-2}}$ to the initial city x_1.

5.2 Constrained Optimization via Penalty Function Approach

We shall consider here a particular case of the constrained problem (55), namely the one with inequality constraints only, that is

$$\begin{aligned} &\max_{\boldsymbol{x}} \sum_{k=1}^{n} c_k x_k \\ \text{s.t. } &\sum_{k=1}^{n} a_{ik} x_k \ge b_i,\ i = 1, \ldots, m, \\ &\boldsymbol{x} \ge 0,\ \ x_k \text{ integer } \forall k = 1, \ldots, n. \end{aligned} \tag{35}$$

Assume in addition that the vector $\boldsymbol{x}$ is binary and all components b_i and a_{ik} are positive numbers. Using the penalty method approach we can reduce the original constraint problem (35) to the following unconstrained one

$$\min_{\boldsymbol{x}} \{S(\boldsymbol{x}) = \sum_{k=1}^{n} c_k x_k + M(\boldsymbol{x})\}, \tag{36}$$

where the penalty function $M(\boldsymbol{x})$ is defined as

$$M(\boldsymbol{x}) = M(\boldsymbol{x}, \beta) = \beta \sum_{i=1}^{m} \min \left\{ \sum_{k=1}^{n} a_{ik} x_k - b_i, 0 \right\}. \tag{37}$$

Here

$$\beta = \frac{a + \sum_{k=1}^{n} c_k}{\min_{ik}(a_{ik} - b_i)}, \tag{38}$$

and a is a positive number. If not stated otherwise we assume that $a = 1$. Note that the penalty parameter β is chosen such that if $\boldsymbol{x}$ satisfies all constraints in Eq. 35, then

$M(\boldsymbol{x}) = 0$ and $S(\boldsymbol{x}) \geq 0$. Alternatively, if $\boldsymbol{x}$ does not satisfy all constraints in Eq. 35, then $M(\boldsymbol{x}) \leq -(a + \sum_{k=1}^{n} c_k x_k)$ and $S(\boldsymbol{x}) \leq -a$.

Clearly, that the optimization program (35) can be again associated with the rare-event probability estimation problem, where $m \in \left(0, \sum_{k=1}^{n} c_k\right)$ and $\boldsymbol{X}$ is a vector of iid Ber(1/2) components. In particular, in order to employ the Gibbs sampler we can write in analogy to Eq. 18 the conditional one dimensional pdfs as

$$g^*(x_i, m_{t-1}|\boldsymbol{x}_{-i}) = r_i(m_{t-1})\, f_i(x_i)\, I_{\{M(\boldsymbol{x})+p_i x_i \geq m_{t-1} - \sum_{j\neq i} p_j x_j\}}, \tag{39}$$

where, as before, $\boldsymbol{x}_{-i}$ denotes the vector $\boldsymbol{x}$ with the i-th component removed and $r_i(m_{t-1})$ denotes the normalization constant.

Sampling a Bernoulli random variable $\widetilde{X}_i$ from Eq. 39 can be performed as follows. Generate $Y \sim \text{Ber}(1/2)$. If $Y = 1$ and $S(x_1, \ldots, x_{i-1}, Y, x_{i+1}, \ldots, x_n) \geq m_{t-1}$, then set $\widetilde{X}_i = 1$, otherwise set $\widetilde{X}_i = 0$.

5.3 Cloning Algorithm for Optimization

Below we present the cloning algorithm for optimization with the objective function given in Eq. 36. We shall use here formula (26) for choosing η_{t-1} and b.

Algorithm 5.1 (The Cloning Algorithm for Optimization)

Given the proposal rarity parameter ρ, say $\rho = 0.1$, the sample size N, say $N = m \times n$, the burn in period b, say $3 \leq b \leq 10$ execute the following steps:

1. **Acceptance–Rejection** Set a counter $t = 1$. Generate a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from the proposal density $f(\boldsymbol{x}, \boldsymbol{p})$. Let $\widetilde{\mathcal{X}}_0 = \{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_0}\}$ corresponds to the largest $(1-\rho)\%$ subset of the population $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ (elite samples) for which $S(\boldsymbol{X}_i) \geq \widehat{m}_0$. Note that $\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_0} \sim g^*(\boldsymbol{x}, \widehat{m}_0)$.
2. **Cloning** Given the number of burn in periods b and the size N_{t-1} of elites at iteration $(t-1)$, find the cloning parameter η_{t-1} according to $\eta_{t-1} = \left\lceil \frac{N}{b\,N_{t-1}} \right\rceil - 1$. Reproduce each vector $\widetilde{\boldsymbol{X}}_k = (\widetilde{X}_{1k}, \ldots, \widetilde{X}_{nk})$ of the elite sample $\{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_{t-1}}\}$ η_{t-1} times, that is take η_{t-1} identical copies of each vector $\widetilde{\boldsymbol{X}}_k$ obtained at the $(t-1)$-th iteration. Denote the entire new population ($\eta_{t-1} N_{t-1}$) cloned vectors plus the original elite sample $\{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_{t-1}}\}$) by $\mathcal{X}_{cl} = \{(\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_1), \ldots, (\widetilde{\boldsymbol{X}}_{N_{t-1}}, \ldots, \widetilde{\boldsymbol{X}}_{N_{t-1}})\}$. To each of the cloned vectors of the population $\mathcal{X}_{cl}$ apply the MCMC (and in particular the Gibbs sampler) for b_{t-1} burn-in periods. Denote the *new entire* population by $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$. Observe that each member of $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ is distributed approximately $g^*(\boldsymbol{x}, \widehat{m}_{t-1})$.
3. **Estimating** m_t Let $\widetilde{\mathcal{X}}_t = \{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_t}\}$ corresponds to the largest $(1-\rho)\%$ subset of the population $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ for which $S(\boldsymbol{X}_i) \geq \widehat{m}_t$. Deliver $\widehat{m}_t$. Note again as that $\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_t}$ is distributed approximately $g^*(\boldsymbol{x}, \widehat{m}_t)$.
4. **Stopping rule** If for some $t \geq d$, say $d = 5$,

$$\widehat{m}_t = \cdots = \widehat{m}_{t-d} \tag{40}$$

then **stop** and deliver $\widehat{m}_{t,N}$ as the estimator of the optimal solution; otherwise, set $t = t + 1$ and return to Step 2.

It follows that Algorithm 5.1 contains only *three parameters: the rarity parameter ρ, the sample size N, and either the cloning parameter η_{t-1}, or the burn-in one b_{t-1}*. Observe that as soon as we specify one of them, say η_{t-1} from Eq. 27, the other one b_{t-1} is obtained automatically.

Also, as mentioned before

- The cloning optimization Algorithm 5.1 differs from its CE counterpart mainly by the way of sampling: in the former the sampling is performed from the non-parametric IS pdf $g^*_{t-1}(\boldsymbol{x})$, while in the later from a parametric IS pdf $f(\boldsymbol{x}, \boldsymbol{v}_t)$.
- CE has the desirable property that the samples are independent, whereas in cloning they are not.

6 Cloning Algorithms for Counting

Here we present two cloning algorithms for rare-events and counting assuming that $\mathcal{X}$ is a discrete space. The main difference between the two is that the first is based on the classic IS pdf (7) and is called the *cloning algorithm*, or *Gibbs cloner*, while the second one uses the classic Boltzmann distribution and is called the *Boltzmann cloning algorithm*. As we shall see below our cloning algorithms are somewhat closer to DHR Algorithm 2.1 rather than to Botev–Kroese Algorithm 3.1. Recall again as that neither bootstrap step nor pilot run is required in the cloning algorithms. Note that the pilot run in Algorithm 3.1 typically takes nearly the same amount of time as the main algorithm, provided one wants to estimate the levels m_t reliably. Note, however, that since the levels $\widehat{m}_t$'s are random variables, the cloning algorithm will generate only from approximate IS pdf g^*_{t-1}. For large samples we can neglect the randomness of $\widehat{m}_t$.

Before proceeding further we need the following:

Remark 6.1 Observe that for the continuous case the estimator (8) can be written (see Botev and Kroese 2008) as

$$\widehat{\ell}(m) = \prod_{t=0}^{T} \widehat{c}_t = \rho^T \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_{iT}) \geq m\}}, \tag{41}$$

where the sample $\boldsymbol{X}_{iT},\ i = 1, \ldots, N$ is from $g^*_{T-1} = g^*(\boldsymbol{x}, m_{T-1})$. Formula (41) does not hold, however, for the discrete case. The reason is that for fixed ρ, the root m_t of

$$\mathbb{P}(S(\boldsymbol{X}) \geq m_t) | S(\boldsymbol{X}) \geq m_{t-1}) = \rho$$

and, thus its sample version

$$\frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_{it}) \geq m_t\}},$$

where $\boldsymbol{X}_{it} \sim g^*_{t-1}$, may not be unique.

We can instead generate a sequence of *adaptive* $\{\rho_t\}$'s by arguing as follows. For fixed ρ and a given ordered sequence of the sample functions $S(\boldsymbol{X}_i),\ i = 1, \ldots, N$ we choose m_t corresponding to the closest (from the left) $(1-\rho)$-th empirical value of the elite sample of $S(\boldsymbol{X}_i),\ i=1,\dots,N$. In short, since N might be larger than the number of values of $S(\boldsymbol{X}_i)$, we include in the elite sample all samples $\boldsymbol{X}_i$ for which $\{S(\boldsymbol{X}_i)\geq m_t\}$, and we denote the elite sample size at iteration t by N_t.

For example, assume that $\rho=0.5$, $N=6$ and that the ordered values of the $S(\boldsymbol{X}_i),\ i=1,\dots,N$ for same iteration t are $\{1,1,2,2,2,3\}$. Then we have $m_t=2$ and the new modified ρ becomes $\rho_t=N_t/N=5/6$.

As we shall see soon that we shall further need to modify Remark 6.1 quite substantially since the issue of proper choice of ρ is crucial in counting, especially when dealing with counting of multiple events based on estimation of the rare event probability (see Eq. 13)

$$\ell=\mathbb{E}_f\left[I_{\{\sum_{i=1}^m C_i(\boldsymbol{X})\geq m\}}\right],$$

which involves dependent Bernoulli random variables $C_i(\boldsymbol{X}),\ i=1,\dots,m$. This is so, since the sample size N is typically larger than the number of different values the function $S(\boldsymbol{X}_i),\ i=1,\dots,N$ can get. In short, each fixed value of $S(\boldsymbol{X}_i)$ typically appears more the one time in a sample of size N. This is in contrast to the combinatorial optimization problems, where N is typically much smaller than the number of different sample function values. For example, in a fully connected TSP with m nodes we typically have $(m-1)!$ different values of $S(\boldsymbol{X})$, while N is of order of several thousands only. Thus, in optimization, the adaptive ρ is the same as the proposal one.

6.1 The Simplified Algorithm

Before presenting the main cloning algorithm by introducing what we call its (1) simplified and (2) basic version. They provide a good insight to the main counting algorithm.

(1) **The simplified version**. Let as before N, ρ_t and N_t be the fixed sample size, the actual rarity parameter (see Remark 6.1) and the number of elites at iteration t, respectively. At the simplified version we set $\eta=1$ (no cloning), that is we apply to each of the N_t elites a *burn-in period of length* $\lceil\rho_t^{-1}\rceil$. By doing so we generate $\lceil\rho_t^{-1}\rceil N_t\approx N$ samples at each level m_t. The rationale of this is based on the fact that if ρ is not small, say $\rho=0.1$, then we have enough stationary elite samples and the goal of the Gibbs sampler is merely to continue with these stationary N_t elites and thus to generate N *new* stationary samples for the next level.
Note that Algorithm 6.1 presents an extension of Algorithm 2.1 in the sense that at each iteration instead of a single elite it uses all N_t elites. Also, it can be viewed as particular case of Algorithm 3.1 with both, the bootstrap step and pilot run being omitted and the length of the burn-in period being equal to ρ^{-1}.

Our numerical experience with all three algorithms: Algorithm 2.1, Algorithm 3.1 and Algorithm 6.1 suggest that none perform well. In particular, they often stop without reaching the final level m. To overcome this difficulty we turn next to our basic version.

(2) **Basic version** As compared to Algorithm 6.1 this version contains an additional step for an adaptive choice of ρ. As we shall see below this additional step will prevent from stopping all three algorithms before reaching the target level m.

Algorithm 6.1 (The Simplified Algorithm for Counting)

Given the rarity parameter $\rho \in (0, 1)$ and the sample size N execute the following steps:

1. **Acceptance–Rejection** Set a counter $t = 1$. Generate a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from the proposal density $f(\boldsymbol{x})$. Let $\widetilde{\mathcal{X}}_0 = \{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_0}\}$ be the largest subset of the population $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ (elite samples) for which $S(\boldsymbol{X}_i) \geq \widehat{m}_0$. Take $\widehat{c}_0 = \widehat{\ell}(\widehat{m}_0)$ in Eq. 16 is an *unbiased* estimator of c_0. Note that $\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_0} \sim g^*(\boldsymbol{x}, \widehat{m}_0)$.
2. **MCMC** For each vector $\widetilde{\boldsymbol{X}}_k = (\widetilde{X}_{1k}, \ldots, \widetilde{X}_{nk}),\ k = 1, \ldots, N_{t-1}$ of the elite sample $\{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_{t-1}}\} \sim g^*(\boldsymbol{x}, \widehat{m}_{t-1})$ obtained at the $(t-1)$-th iteration apply $\lceil \rho_t^{-1} \rceil$ burn-in periods, while using the MCMC (and in particular the Gibbs) sampler and, thus generate $\lceil \rho_t^{-1} \rceil$ new vectors $(\boldsymbol{X}_{k1}, \ldots, \boldsymbol{X}_{k\lceil \rho_t^{-1} \rceil})$. Note that the *new entire* population $\{(\boldsymbol{X}_{s1}, \ldots, \boldsymbol{X}_{s\lceil \rho_t^{-1} \rceil}),\ s = 1, \ldots, N_{t-1}\}$ of length $\lceil \rho_t^{-1} \rceil N_{t-1} \approx N$, which is denoted as $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$, is distributed approximately $g^*(\boldsymbol{x}, \widehat{m}_{t-1})$.
3. **Estimating** c_t Let $\widetilde{\mathcal{X}}_t = \{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_t}\}$ be the subset of the population $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ for which $S(\boldsymbol{X}_i) \geq \widehat{m}_t$. Take $\widehat{c}_t$ in Eq. 9 as an estimator of c_t given in Eq. 6. Note again as that $\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_t}$ is distributed approximately $g^*(\boldsymbol{x}, \widehat{m}_t)$.
4. **Stopping rule** If $m_t = m$ go to step 5, otherwise set, set $t = t + 1$ and repeat from step 2.
5. **Final Estimator** Deliver $\widehat{\ell}(m)$ given in Eq. 8 as an estimator of $\ell(m)$ and $|\widehat{\mathcal{X}^*}| = \widehat{\ell}(m)|\mathcal{X}|$ as an estimator of $|\mathcal{X}^*|$.

To proceed note first that the draw back of the approach for updating ρ based on Remark 6.1 is that one can often run into a situation, where $N_t = N$ and thus $\rho_t = 1$. This is the main reason that all three algorithms often stop before reaching the desired level m. The following example provides details. Assume that $N = 9$ and let the proposal (non-adaptive) $\rho = 1/3$. Consider the following two sample scenarios of the ordered values of $S(\boldsymbol{X}_i),\ i = 1, \ldots, 9$:

(1) $S(\boldsymbol{X}_i) = (1, 1, 1, 2, 2, 2, 2, 3, 3)$ and (2) $S(\boldsymbol{X}_i) = (1, 1, 1, 1, 1, 1, 1, 2, 3)$. Following Remark 6.1 it is readily seen that in cases (1) and (2) the actual values of ρ are $\rho_1 = 6/9$ and $\rho_2 = 1$, respectively. They are based on the elite sequences $\{2, 2, 2, 2, 3, 3\}$ and $\{1, 1, 1, 1, 1, 1, 1, 2, 3\}$, respectively. Note that both $\rho_1 > \rho$ and $\rho_2 > \rho$. Note, however, that we can not use $\rho_2 = 1$ (corresponding to $S(\boldsymbol{X}_i) = 1$), since as soon as we obtain $\rho = 1$ the cloning algorithm will stop and, thus the level m will be never reached. To prevent this we modify ρ_2 by moving from the level corresponding to $S(\boldsymbol{X}_i) = 1$ to the next level of $S(\boldsymbol{X}_i)$, that is to the level $S(\boldsymbol{X}_i) = 2$. This corresponds to the elite sequence $\{2, 3\}$ with the new modified $\rho_2 = N_t/N = 2/9 < \rho$. Based in this we shall further require that ρ should be in some fixed interval (a_1, a_2), say $(a_1, a_2) = (0.01, 0.25)$. This means that when the number of elites $N_t > a_2 N$ we automatically switch from a lower elite level to a higher one; if $a_1 N \leq N_t \leq a_2 N\ \ (\rho \in (a_1, a_2))\rho \in (a_1, a_2)$, and thus $a_1 \leq \rho_t \leq a_2$, we accept N_t as the size of the elites sample; and if $N_t < a_1 N$, we proceed sampling until $N_t = a_1 N$, that is until we obtain at least $a_1 N$ elites.

We summarize this as:

Remark 6.2 **Adaptive choice of** $\rho \in (a_1, a_2)$ For fixed N, some fixed ρ, say $\rho = 0.1$, called the proposal rarity parameter, let $S_{\lceil 1-\rho \rceil}$ be the largest elite value of the ordered sample $S(\boldsymbol{X}_i),\ i = 1, \ldots, N$, corresponding to that fixed ρ. The adaptive choice of $\rho \in (a_1, a_2)$, where (a_1, a_2) is some fixed interval, say $(a_1, a_2) = (0.01, 0.25)$ is performed as follows:

- *Include into the elite sample all additional values* $S_{\lceil 1-\rho \rceil} = S_{\min}$ (see Remark 6.1), provided that at the iteration t the number of elite samples $N_t \le a_2 N$.
- *Remove from the elite all values* $S_{\lceil 1-\rho \rceil} = S_{\min}$, provided the number of elite samples $N_t > a_2 N$. Note that by doing so we switch from a lower elite level to a higher one. If $a_1 N \le N_t \le a_2 N$, and thus $a_1 \le \rho_t \le a_2$, accept N_t as the size of the elites sample. If $N_t < a_1 N$, proceed sampling until $N_t = a_1 N$, that is until at least $a_1 N$ elite samples are obtained. The above guarantees that $\rho \in (a_1, a_2)$.

  Note that the main reason that Algorithm 6.1 *fails* to reach the target value m is that we used ρ based on Remark 6.1, rather than the adaptive ρ satisfying $a_1 < \rho \le a_2$.

We shall call Algorithm 6.1 with this additional step, the *Basic Algorithm*. It is important to note that adding the above step to all three above algorithms (Algorithm 2.1, Algorithm 3.1 and Algorithm 6.1), improves substantially their performance in the sense that all three ones were able to reach the desired level m in most of our experiments. In particular, while estimating rare-events for the sum of iid Bernoulli random variables we found that our Basic Algorithm outperforms Algorithm 3.1.

6.2 The Main Cloning Algorithm for Counting

To increase further the accuracy of the Basic Algorithm, we turn next to its main version, where we introduce two additional steps.

1. **Screening step**. Since the IS pdf $g^*(\boldsymbol{x}, m_t)$ must be *uniformly distributed* for each fixed m_t, the cloning algorithm checks at each iteration whether or not *all elite vectors* $\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_t}$ *are different*. If this is not the case, we screen out (eliminate) all redundant elite samples. We denote the resulting elite sample as $\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_t}$ and call it, *the screened elite sample*. Observe that this procedure prevents (at least partially) the empirical pdf associated with $\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_t}$ from deviation from the uniform. Note that alternatively, one may first employ screening to the entire sample and only then define the elite sampling. We shall use the first alternative.
2. **Cloning step**. This step is defined in details in Section 4.

Below we present the main cloning algorithm for counting, also called the *Gibbs cloner*. If not stated otherwise we shall use below Eq. 27 for choosing η_{t-1} and b.

Algorithm 6.2 (Main Cloning Algorithm for Counting)

Given the proposal rarity parameter ρ, say $\rho = 0.1$, the parameters a_1 and a_2, say $a_1 = 0.01$ and $a_2 = 0.25$, such that $\rho \in (a_1, a_2)$, and the sample size N, say $N = m \times n$, execute the following steps:

1. **Acceptance–Rejection**—the same as in Algorithm 6.1.
2. **Adaptive choice of** ρ For each iteration use the adaptive choice of $a_1 \leq \rho \leq a_2$, with a_1 and a_2 defined in Remark 6.2.
3. **Screening** Denote the elite sample obtained at iteration $(t-1)$ by $\{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_{t-1}}\}$. Screen out the redundant elements of the subset $\{\widetilde{\boldsymbol{X}}_1, \ldots, \widetilde{\boldsymbol{X}}_{N_{t-1}}\}$ and denote the resulting one as $\{\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$.
4. **Cloning** Given the size N_{t-1} of screened elites at iteration $(t-1)$, find the cloning and the burn-in parameters η_{t-1} and b_{t-1} according to Eq. 27. Reproduce η_{t-1} times each vector $\widehat{\boldsymbol{X}}_k = (\widehat{X}_{1k}, \ldots, \widehat{X}_{nk})$ of the screened elite sample $\{\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$, that is, take η_{t-1} identical copies of each vector $\widehat{\boldsymbol{X}}_k$ obtained at the $(t-1)$-th iteration. Denote the entire new population ($\eta_{t-1} N_{t-1}$ cloned vectors plus the original screened elite sample $\{\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$) by $\mathcal{X}_{cl} = \{(\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_1), \ldots, (\widehat{\boldsymbol{X}}_{N_{t-1}}, \ldots, \widehat{\boldsymbol{X}}_{N_{t-1}})\}$. To each of the cloned vectors of the population $\mathcal{X}_{cl}$ apply the MCMC (and in particular the Gibbs sampler) for b_{t-1} burn-in periods. Denote the *new entire* population by $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$. Observe that each member of $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ is distributed approximately $g^*(\boldsymbol{x}, \widehat{m}_{t-1})$.
5. **Estimating** c_t—the same as in Algorithm 6.1.
6. **Stopping rule**—the same as in Algorithm 6.1.
7. **Final Estimator**—the same as in Algorithm 6.1.

Remark 6.3 **The honesty of the algorithm** If at some fixed iteration t Algorithm 6.2 can not proceed from m_t to m_{t+1}, then we can try to increase the sample size N, say by factor of 5–10. If this does not help, then Algorithm 6.2 declares that it can not handle the problem. We call such property of Algorithm 6.2, the *honesty property*. Note that in all our numerical studies with Algorithm 6.2 it never stopped before reaching the level m.

Remark 6.4 **Improving uniformity**

To reduce further the deviation from the uniform of the sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ one can use at each iteration in parallel to the screening step an additional, the so-called *thinning* step. In the thinning step we accept, say only the even samples from the total population $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$, or, say only each third one. Clearly, by doing so

1. The total sample should be doubled (tripled) at each iteration.
2. The dependence between the components $\boldsymbol{X}_2, \boldsymbol{X}_4, \ldots, \boldsymbol{X}_{2N}$ (thinning using even samples) decreases.

Remark 6.5 **Alternative choice of** ρ As an alternative to Remark 6.2 for choice of ρ we can define ρ based on the maximum value of the ordered sample $S(\boldsymbol{X}_i)$, $i = 1, \ldots, N$, denoted as $S_{\max}$, rather than on $S_{\lceil 1-\rho \rceil}$, which corresponds to the largest

$\rho 100\%$ elite value. This is so, since the sample size N is typically larger than the number of levels m in the model. Thus, we expect that the number of $S_{\max}$ values, denoted as $\#S_{\max}$ will be greater than 1. The adaptive ρ in this case will correspond to $\rho = \frac{\#S_{\max}}{N}$.

As an example consider again as the following two sample scenarios of the ordered values of $S(\boldsymbol{X}_i),\ i = 1, \dots, 9$:

(1) $S(\boldsymbol{X}_i) = (1, 1, 1, 2, 2, 2, 2, 3, 3)$ and (2) $S(\boldsymbol{X}_i) = (1, 1, 1, 1, 1, 1, 1, 2, 3)$. We have (i) $\rho = 2/9$ and (2) $\rho = 1/9$. Note that since in case (2) we have that $\#S_{\max} = 1$, that is only a single value of $S_{\max} = 3$, we can include into the elite sample all additional smaller values, corresponding (in this case) to $S = 2$. By doing so we obtain $\rho = 2/9$ instead of $\rho = 1/9$.

In summary, in the alternative ρ approach we first select some *target* ρ value, say $\rho = 0.01$ and then we accumulate all the largest values of $S(\boldsymbol{X}_i)$ until we obtain that

$$\frac{\text{number of accumulated largest values of S}(\boldsymbol{X}_{\text{i}})}{N \geq 0.01.}$$

Remark 6.6 **The direct estimator** As an alternative to the estimator $|\widehat{\mathcal{X}^*}|$ obtained by Algorithm 6.2 we can use the one based on *direct counting* of the number of the screened samples obtained just after crossing the level m. Such counting estimator, denoted by $|\widehat{\mathcal{X}^*_{\text{dir}}}|$, is associated with the *empirical* distribution of the uniform distribution $g^*(\boldsymbol{x}, m)$. We found numerically that $|\widehat{\mathcal{X}^*_{\text{dir}}}|$ is extremely useful and very accurate. Note that it can be applied only for counting problems with $|\mathcal{X}^*|$ being not too large. In particular, $|\mathcal{X}^*|$ should be less than the sample size N, that is $|\mathcal{X}^*| < N$. Note also that counting problems with values small relative to $|\mathcal{X}|$ are known as the must difficult ones and in many problems one is indeed interested to count only if $|\mathcal{X}^*|$ is no greater then some fixed quantity, say $\mathcal{N}$. Clearly, this is possibly only if $N \geq \mathcal{N}$.

It is important to note that $|\widehat{\mathcal{X}^*_{\text{dir}}}|$ is typically much more accurate than its counterpart, the standard estimator $|\widehat{\mathcal{X}^*}| = \widehat{\ell}|\mathcal{X}|$. The reason is that $|\widehat{\mathcal{X}^*_{\text{dir}}}|$ is obtained *directly* by counting all distinct values of $\boldsymbol{X}_i,\ i = 1, \dots, N$, satisfying $S(\boldsymbol{X}_i) \geq m$, that is it can be written as

$$|\widehat{\mathcal{X}^*_{\text{dir}}}| = \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i^{(d)}) \geq m\}}, \tag{42}$$

where $\boldsymbol{X}_i^{(d)} = \boldsymbol{X}_i$, if $\boldsymbol{X}_i \neq \boldsymbol{X}_j,\ \forall j = 1, \dots, i-1$ and $\boldsymbol{X}_i^{(d)} = 0$, otherwise. Note that we set in advance $\boldsymbol{X}_1^{(d)} = \boldsymbol{X}_1$.

To increase further the accuracy of $|\widehat{\mathcal{X}^*_{\text{dir}}}|$ we can take a larger sample at the last step of Algorithm 6.2, that is after reaching the level m.

Since the direct estimator $|\widehat{\mathcal{X}^*_{\text{dir}}}|$ is extremely useful in counting we present next the relevant cloning algorithm.

Algorithm 6.3 (The Direct Cloning Algorithm for Counting)

Given the proposal rarity parameter ρ, say $\rho = 0.1$, the parameters a_1 and a_2, say $a_1 = 0.01$ and $a_2 = 0.25$, such that $\rho \in (a_1, a_2)$, and the sample size N, say $N = m \times n$, execute the following steps:

1. **Acceptance–Rejection**—as in Algorithm 6.1
2. **Adaptive choice of** ρ—the same as in Algorithm 6.3.
3. **Screening**—the same as in Algorithm 6.2.
4. **Cloning**—the same as in Algorithm 6.2.
5. **Estimating** m_t—the same as in Algorithm 5.1.
6. **Stopping rule**—the same as in Algorithm 6.2.
7. **Final Estimator** For $m_T = m$, take a sample of size N, and deliver $|\widetilde{\mathcal{X}}^*_{\text{dir}}|$ in Eq. 42 as an estimator of $|\mathcal{X}^*|$.

In Section 10.2 we discuss the complexity of the estimator $|\widehat{\mathcal{X}}^*_{\text{dir}}|$.

Table 1 presents comparative studies of the performance of Algorithm 6.2 for it different parameter configurations as well as the performance of the DHR and Botev–Kroese (BK) algorithms while estimating

$$\ell = \mathbb{E}_f \left[I_{\{\sum_{i=1}^n X_i \geq m\}} \right],$$

where the X_i's are iid Ber(1/2), $n = 20$ and $m = 19$. Table 2 presents similar data for $n = 100$ and $m = 99$. In both experiments we set $N = 1{,}000$ and $\rho = 0.05$ and the results were averaged over 100 independent runs.

Here RE and RE_{dir} denote the relative error of the estimators $|\widehat{\mathcal{X}}^*|$ and $|\widehat{\mathcal{X}}^*_{\text{dir}}|$, respectively. Observe that same values of $|\widehat{\mathcal{X}}^*|$ and $|\widehat{\mathcal{X}}^*_{\text{dir}}|$ in our tables, here and below, are not integers. There reason is that they present point estimators of $|\mathcal{X}^*|$ and $|\mathcal{X}^*_{\text{dir}}|$ averaged over 10 independent replications. Note also that in the version N_o1 of Algorithm 6.2 we set $a_1 = 0.01$ and $a_2 = 0.25$, while we did not restrict the remaining ones that is they were $a_1 = 0$ and $a_2 = 1$. In the cloning Algorithm 6.2 we implemented the burn-in strategy (26). It follows from these table that Algorithm 6.2 with the burn-in period $b = 10$ is the best. Similar performance was obtained for $3 \leq b \leq 10$. Note that the version N_o2 and N_o3 of Algorithm 6.2 are somewhat similar to the version N_o4 and N_o5 of DHR and BK, respectively in the sense that the burn-in periods are $b = N$ and $b = 1$, respectively. Note also that for the case $n = 100$ and $m = 99$ Algorithm 3.1 did not converge to $m = 99$. It is interesting to note that while increasing the sample size from $N = 1{,}000$ to $N = 10{,}000$ for the Bernoulli model with $n = 100$ and $m = 99$, while leaving the remaining data the same, we found that the performance of all five versions substantially improved and in this

Table 1 Comparative performance of the algorithms for the sum of $n = 20$ iid Bernoulli random variables for $m = 19$, $N = 1{,}000$ and $\rho = 0.05$

N_o	Algorithm	b	η	$\lvert\widehat{\mathcal{X}}^*\rvert$	RE	$\lvert\widehat{\mathcal{X}}^*_{\text{dir}}\rvert$	RE_{dir}	CPU
1	New	10	$\eta_t = \lceil (N/bN_t) \rceil - 1$	21.20	0.047	21.0	0	0.8
2	New	N/N_t	0	21.92	0.147	21.0	0	0.9
3	New	1	N/N_t	21.78	0.063	21.0	0	0.7
4	BK	1	N/N_t	27.32	0.305	21.0	0	0.3
5	DHR	N	0	20.22	0.137	21.0	0	14.5

Table 2 Comparative performance of the algorithms for the sum of $n = 100$ iid Bernoulli random variables for $m = 99$, $N = 1{,}000$ and $\rho = 0.05$

N_o	Algorithm	b	η	$\lvert\widehat{\mathcal{X}^*}\rvert$	RE	$\lvert\widehat{\mathcal{X}^*_{\text{dir}}}\rvert$	RE_{dir}	CPU
1	New	10	$\eta_t = \lceil (N/bN_t) \rceil - 1$	101.7	0.041	101.0	0	33
2	New	N/N_t	0	122.6	0.206	100.7	0.01	27
3	New	1	N/N_t	118.2	0.072	101.0	0	26
4	BK	1	N/N_t	220	0.576	100,1	0.05	25
5	DHR	N	0	207.3	0.354	101	0.0	288

case Algorithm 3.1 also reaches the level $m = 99$, but the relative error is still large as compared to the remaining algorithms. In particular the relative error RE was decreased approximately by a factor of 5. Note that if the data would be independent one would obtain decrease in relative error by a factor of $10^{1/2} \approx 3$ only. Still the relative efficiencies of the 5 versions were similar to the case of $N = 1{,}000$. Finally, it follows from the tables, that in spite of the high relative error RE and substantial bias of some of the versions of $|\widehat{\mathcal{X}^*}|$, the direct estimators $|\widehat{\mathcal{X}^*_{\text{dir}}}|$ are very accurate for all 5 cases. The explanation for such nice behavior of $|\widehat{\mathcal{X}^*_{\text{dir}}}|$ is given in Remark 6.6 below.

We also run Algorithm 6.2 for the sum of Bernoulli random variables with $n = m$ and obtained similar results. Clearly, in this case $|\mathcal{X}^*| = 1$.

We finally define the most naive acceptance–rejection version of Algorithm 6.2, called the *$(N = 1)$-policy algorithm*. According to the $(N = 1)$-policy algorithm, at each fixed level m_{t-1} we use the acceptance–rejection (single trial) method, until for the first time we hit a higher level $m_t > m_{t-1}$. Using the $(N = 1)$-policy algorithm we always managed to reach any level $m \le n$, even for large n, like $n = 1{,}000$.

In Section 10 we discuss the complexity of the $(N = 1)$-policy algorithm.

For the continuous case Algorithm 6.2 simplifies substantially. In particular its steps 2 and 3 (adaptive ρ and screening) can be omitted. For details see (Rubinstein 2008b).

6.3 The Boltzmann Cloning Algorithm

Many problems including the classic Ising model are based on the Boltzmann distribution

$$g(\boldsymbol{x}) = Z(\lambda)^{-1} f(\boldsymbol{x}) \exp\{-\lambda S(\boldsymbol{x})\}, \tag{43}$$

where $Z(\lambda) = \mathbb{E}_f\left[\exp\{-\lambda S(\boldsymbol{X})\}\right]$ is called the partition function.

Note that in analogy to Eq. 4 and 5 we can write $Z = Z(m)$ as

$$Z(m) = \mathbb{E}_f\left[\exp\left\{-\sum_{i=1}^{m_0} \lambda C_i(\boldsymbol{X})\right\}\right] \times \prod_{t=1}^{T} \mathbb{E}_f\left[\exp\left\{-\sum_{i=1}^{m_t} \lambda C_i(\boldsymbol{X})\right\} \mid \exp\left\{-\sum_{i=1}^{m_{t-1}} \lambda C_i(\boldsymbol{X})\right\}\right]$$

$$\begin{aligned}
&= \mathbb{E}_f\left[\exp\left\{-\sum_{i=1}^{m_0}\lambda C_i(\boldsymbol{X})\right\}\right]\prod_{t=1}^{T}\mathbb{E}_{g_{t-1}}\left[\exp\left\{-\sum_{i=1}^{m_t}\lambda C_i(\boldsymbol{X})\right\}\right]\\
&= z_0\prod_{t=1}^{T} z_t,
\end{aligned} \tag{44}$$

where

$$\begin{aligned}
z_t &= \mathbb{E}_f\left[\exp\left\{-\sum_{i=1}^{m_t}\lambda C_i(\boldsymbol{X})\right\} \mid \exp\left\{-\sum_{i=1}^{m_{t-1}}\lambda C_i(\boldsymbol{X})\right\}\right]\\
&= \mathbb{E}_{g_{t-1}}\left[\exp\left\{-\sum_{i=1}^{m_t}\lambda C_i(\boldsymbol{X})\right\}\right],
\end{aligned} \tag{45}$$

$z_0 = \mathbb{E}_f\left[\exp\left\{-\sum_{i=1}^{m_0}\lambda C_i(\boldsymbol{X})\right\}\right]$ and as usual, the sequence $\{m_t,\ t=0,1,\ldots,T\}$ satisfies $0<m_0<m_1<\ldots<m_T=m$ and it is chosen on-line (adaptively), f denotes the original pdf $f(\boldsymbol{x})$, and $g_{t-1}=g(\boldsymbol{x},m_t,\lambda=-\infty)$ denotes the zero variance (Boltzmann) pdf at iteration t.

The estimator of $Z(m)$ can be written in analogy to Eqs. 8 and 9 as

$$\widehat{Z}(m)=\prod_{t=0}^{T}\widehat{z}_t, \tag{46}$$

where

$$\widehat{z}_t=\frac{1}{N}\sum_{j=1}^{N}\exp\left\{-\sum_{i=1}^{m_t}\lambda C_i(\boldsymbol{X}_j)\right\} \tag{47}$$

and $\boldsymbol{X}_j \sim g(\boldsymbol{x},\lambda,m_{t-1})$.

Here we present a modified version of Algorithm 6.2 for estimation of the partition function $Z(\lambda)$.

To proceed, note (Rubinstein 2008a, b) that Eq. 43 can be derived from the solution of the following single-constrained MinxEnt program

$$\begin{aligned}
&\min_g \mathcal{D}(g,h)=\min_g \int \ln\tfrac{g(\boldsymbol{x})}{f(\boldsymbol{x})}\,g(\boldsymbol{x})\mathrm{d}\boldsymbol{x}=\min_g \mathbb{E}_g\left[\ln\tfrac{g(\boldsymbol{X})}{f(\boldsymbol{X})}\right]\\
&\text{s.t. } \mathbb{E}_g\left[\sum_{i=1}^{m}C_i(\boldsymbol{X})\right]=m\\
&\int g(\boldsymbol{x})\mathrm{d}\,\boldsymbol{x}=1.
\end{aligned} \tag{48}$$

It is also important to note (Rubinstein 2008a, b) that for $\lambda=-\infty$ the pdf $g(\boldsymbol{x})$ in Eq. 43 coincides with the zero-variance IS pdf (7), that is with

$$g^*(\boldsymbol{x},m)=\ell^{-1}f(\boldsymbol{x})I_{\{S(\boldsymbol{x})\geq m\}}. \tag{49}$$

Because of this relation one can easily *switch* from the Boltzmann pdf (43) to the IS zero-variance pdf (49) and thus to apply the original Algorithm 6.2 instead of the Boltzmann type algorithm below and vise-versa, provided $\lambda=-\infty$ in Eq. 43. This, for example, means that when $\lambda = -\infty$ the classic Ising model can be treated by using the IS zero-variance pdf (49) rather than via the original Boltzmann pdf (43).

To proceed note that similar to the parametric MinxEnt algorithm (Rubinstein 2008a, b) no acceptance–rejection step is explicitly involved in the Boltzmann cloning algorithm below. As result neither the *rarity parameter* ρ nor *elite sampling* are explicitly involved here. In fact, the elite sampling will be hidden, and it will become explicitly available only after the sequence $\{\widehat{m}_t,\ t = 1, \dots, T\}$ is generated. Observe that here $\widehat{m}_t$ corresponds to the *maximal* level reached so for at iteration t.

To clarify, consider a rare event probability ℓ associated with the sample performance $S(\boldsymbol{X})$ being the sum of iid Bernoulli random variables, that is $S(\boldsymbol{X}) = \sum_{i=1}^m X_i$. Assume for concreteness that we took a sample $N = 1{,}000$ and while running the simulation we obtained (at some iteration t) that all 1,000 sample values $S_k \in (a, b),\ k = 1, \dots, 1000$, where, for concreteness, say $a = 20$ and $b = 50$. Since $\widehat{m}_t$ corresponds to the maximum level reached at iteration t, we clearly have that $\widehat{m}_t = b = 50$ and since $\lambda = -\infty$, all the terms of S_k which are less than b are negligible in the exponent $\exp\left\{-\sum_{i=1}^m \lambda X_{ik}\right\}$. Assume, finally that the number of terms, which we also call here, the number of elite samples, say N_t for which $S_k = 50$

Algorithm 6.4 (The Boltzmann Cloning Algorithm)

Given the sample size N, execute the following steps:

1. **Acceptance–Rejection** Set a counter $t = 1$. Generate a sample $\boldsymbol{X}_1, \dots, \boldsymbol{X}_N$ from the proposal density $f(\boldsymbol{x})$. Let $\widetilde{\mathcal{X}}_0 = \{\widetilde{\boldsymbol{X}}_1, \dots, \widetilde{\boldsymbol{X}}_{N_0}\}$ be the largest subset of the population $\{\boldsymbol{X}_1, \dots, \boldsymbol{X}_N\}$ (elite samples) for which $S(\boldsymbol{X}_i) \geq \widehat{m}_0$. Take $\widehat{z}_0 = \widehat{z}(\widehat{m}_0)$ in Eq. 47 is an *unbiased* estimator of z_0. Note that $\widetilde{\boldsymbol{X}}_1, \dots, \widetilde{\boldsymbol{X}}_{N_0} \sim g(\boldsymbol{x}, \lambda, \widehat{m}_0)$.
2. **Screening**—the same as in Algorithm 6.2.
3. **Adaptive choice of** ρ Estimate ρ according to the adaptive implicit rule $\widehat{\rho}_t = N_t/N$, where N_t includes all elite samples $S(\boldsymbol{X}_i)$ corresponding to the maximum level $\widehat{m}_t$ of the entire sample $S(\boldsymbol{X}_i),\ i = 1, \dots, N$.
4. **Cloning** Given the size N_{t-1} of screened elites at iteration $(t-1)$, find the cloning and the burn-in parameters η_{t-1} and b_{t-1} according to Eq. 27. Reproduce η_{t-1} times each vector $\widehat{\boldsymbol{X}}_k = (\widehat{X}_{1k}, \dots, \widehat{X}_{nk})$ of the screened elite sample $\{\widehat{\boldsymbol{X}}_1, \dots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$, that is, take η_{t-1} identical copies of each vector $\widehat{\boldsymbol{X}}_k$ obtained at the $(t-1)$-th iteration. Denote the entire new population ($\eta_{t-1} N_{t-1}$ cloned vectors plus the original screened elite sample $\{\widehat{\boldsymbol{X}}_1, \dots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$) by $\mathcal{X}_{cl} = \{(\widehat{\boldsymbol{X}}_1, \dots, \widehat{\boldsymbol{X}}_1), \dots, (\widehat{\boldsymbol{X}}_{N_{t-1}}, \dots, \widehat{\boldsymbol{X}}_{N_{t-1}})\}$. To each of the cloned vectors of the population $\mathcal{X}_{cl}$ apply the MCMC (and in particular the Gibbs sampler) for b_{t-1} burn-in periods. Denote the *new entire* population by $\{\boldsymbol{X}_1, \dots, \boldsymbol{X}_N\}$. Observe that each member of $\{\boldsymbol{X}_1, \dots, \boldsymbol{X}_N\}$ is distributed approximately $g(\boldsymbol{x}, \lambda, \widehat{m}_{t-1})$.
5. **Estimating** z_t Let $\widetilde{\mathcal{X}}_t = \{\widetilde{\boldsymbol{X}}_1, \dots, \widetilde{\boldsymbol{X}}_{N_t}\}$ be the subset of the population $\{\boldsymbol{X}_1, \dots, \boldsymbol{X}_N\}$ for which $S(\boldsymbol{X}_i) \geq \widehat{m}_t$. Take $\widehat{z}_t$ in Eq. 47 as an estimator of z_t given in Eq. 45. Note again as that $\widetilde{\boldsymbol{X}}_1, \dots, \widetilde{\boldsymbol{X}}_{N_t}$ is distributed approximately $g(\boldsymbol{x}, \lambda, \widehat{m}_t)$.
6. **Stopping rule**—the same as in Algorithm 6.2.
7. **Final Estimator** Deliver $\widehat{Z}(m)$ according to Eq. 46 as an estimator of $Z(m)$ given in Eq. 44.

equals 25. So, as before, we can define $\widehat{\rho}_t = N_t/N$ and call it *adaptive implicit* ρ, (for our case it is $\widehat{\rho}_t = \frac{25}{1000} = 1/40$).

Note that typically we expect $\widehat{\rho}_t > 10^{-2}$, since we take at each iteration a sample size $N > m/\rho$, where say $\rho = 10^{-1}$. If however, at some iteration t, we occasionally obtain that $\widehat{\rho}_t < \rho$, we can reject the largest value $\widehat{m}_t$ obtained so far and take instead the value $\widehat{m}_t - 1$, provided $N > (\widehat{m}_t - 1)/\rho$ and so far. In our example $\widehat{m}_t - 1 = 49$.

The Boltzmann cloning algorithm basically coincides with Algorithm 6.2. Here we sample from the Boltzmann pdf $g(\boldsymbol{x}, m_t, \lambda = -\infty)$ (see Eq. 43) rather than from the IS pdf $g^*(\boldsymbol{x}, m_t)$ (see Eq. 7) and the step 2 of Algorithm 6.2 for choosing the adaptive $\rho \in (a_1, a_2)$ is replaced by the corresponding adaptive implicit one defined above. Also, as in Algorithm 6.2 we use the term *screened elites*. This is regardless of the fact that the elite samples are obtained only in an implicit way.

6.4 Counting Multiple Events

As for application of the cloning Algorithm 6.2 consider counting on the set

$$\mathcal{X}^* = \{\boldsymbol{x} \in \mathbb{R}^n : S_i(\boldsymbol{x}) \geq b_i,\ i = 1, \ldots, m\}, \tag{50}$$

where $S_i(\boldsymbol{x}),\ i = 1, \ldots, m$ are arbitrary functions. We can associate with Eq. 50 the following *multiple-event* probability

$$\ell = \mathbb{P}_{\boldsymbol{u}} \left\{ \bigcap_{i=1}^{m} [S_i(\boldsymbol{X}) \geq b_i] \right\} = \mathbb{E}_{\boldsymbol{u}} \left[\prod_{i=1}^{m} I_{\{S_i(\boldsymbol{X}) \geq b_i\}} \right]. \tag{51}$$

Here $\boldsymbol{u}$ means the uniform distribution. Note that some of the events may be given as equalities, that is as, $\{S_i(\boldsymbol{X}) = b_i\}$. Note also that Eq. 51 has some interesting applications in rare-event simulation. For example, in a queueing model one might be interested in estimating the probability of the simultaneous occurrence of two events, $\{S_1(\boldsymbol{X}) \geq b_1\}$ and $\{S_2(\boldsymbol{X}) \geq b_2\}$, where the first is associated with buffer overflow (the number of customers S_1 is at least b_1), and the second is associated with the sojourn time (the waiting time of the customers S_2 in the queuing system is at least b_2).

We assume that each individual event $\{S_i(\boldsymbol{X}) \geq b_i\},\ i = 1, \ldots, m$, is *not rare*, that is each probability $\mathbb{P}_{\boldsymbol{u}}\{S_i(\boldsymbol{X}) \geq b_i\}$ is not a rare-event probability, say $\mathbb{P}_{\boldsymbol{u}}\{S_i(\boldsymbol{X}) \geq b_i\} \geq 10^{-4}$, but their intersection forms a *rare-event probability* ℓ. We are interested in efficient estimation of ℓ defined in Eq. 51 and counting the number of points (volume) on the set $\mathcal{X}^*$.

To proceed note that under the IS pdf $g^*(\boldsymbol{x})$ all constraints $\{S_i(\boldsymbol{x}) \geq b_i,\ i = 1, \ldots, m\}$ must be fulfilled, provided the set is not empty. This is equivalent of saying that the rare-event probability ℓ in Eq. 51 becomes certain under $g^*(\boldsymbol{x})$, that is,

$$\mathbb{E}_{g^*} \left[\prod_{i=1}^{m} I_{\{S_i(\boldsymbol{X}) \geq b_i\}} \right] = 1. \tag{52}$$

In other words, Eq. 52 states that under such *ideal* IS pdf $g^*(\boldsymbol{x})$ all m indicators must be equal to unity with probability 1. This can also be written as

$$\mathbb{P}_{\boldsymbol{u}} \left\{ \left(\sum_{i=1}^{m} C_i(\boldsymbol{X}) \right) = m \right\} = \mathbb{E}_{g^*} \left[I_{\{\mathcal{C}(\boldsymbol{X}) = m\}} \right] = 1, \tag{53}$$

where

$$\mathcal{C}(\boldsymbol{X}) = \sum_{i=1}^{m} C_i(\boldsymbol{X}), \tag{54}$$

and $C_i(\boldsymbol{X}) = I_{\{S_i(\boldsymbol{X}) \geq b_i\}}$. Similar to Eq. 52, formula (53) states that under $g^*(\boldsymbol{x})$ the probability of the sum of m indicator random variables $C_i(\boldsymbol{X})$ being equal to m, (m is the number of constraints) must be equal to 1.

We next deal with counting the number of feasible solutions in an integer program with both equality and inequality constraints, which reads as

$$\begin{aligned} & \max c'\boldsymbol{x}, \\ \text{s.t.} & \sum_{k=1}^{n} a_{ik} x_k = b_i, \; i = 1, \ldots, m_1, \\ & \sum_{k=1}^{n} a_{jk} x_k \geq b_j, \;\; j = m_1 + 1, \ldots, m_1 + m_2, \\ & \boldsymbol{x} \geq 0, \;\; x_k \text{ integer } \forall k = 1, \ldots, n, \end{aligned} \tag{55}$$

where $\boldsymbol{c}$ and $\boldsymbol{x}$ are n-dimensional vectors.

In particular we shall show how to use the Algorithm 6.2 for counting on the set containing both equality and inequality constraints defined in Eq. 55, that is, on the set (12)

$$\begin{aligned} & \sum_{k=1}^{n} a_{ik} x_k = b_i, \; i = 1, \ldots, m_1, \\ & \sum_{k=1}^{n} a_{jk} x_k \geq b_j, \;\; j = m_1 + 1, \ldots, m_1 + m_2, \\ & \boldsymbol{x} \geq 0, \;\; x_k \text{ integer } \forall k = 1, \ldots, n. \end{aligned}$$

It is readily seen that in this case the rare event probability ℓ reduces to Eq. 13, that is to

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

where the first m_1 terms $C_i(\boldsymbol{X})$'s in Eq. 54 can be written as in Eq. 14, that is as

$$C_i(\boldsymbol{X}) = I_{\{\sum_{k=1}^{n} a_{ik} X_k = b_i\}}, \; i = 1, \ldots, m_1,$$

while the remaining m_2 ones are as in Eq. 15, that is as

$$C_i(\boldsymbol{X}) = I_{\{\sum_{k=1}^{n} a_{ik} X_k \geq b_i\}}, \; i = m_1 + 1, \ldots, m_1 + m_2.$$

Indeed, in order to count the number $|\mathcal{X}^*|$ of feasible solutions of the program (55), that is, on the set (12), we associate with it the following rare-event probability

$$\ell = \mathbb{P}_{\boldsymbol{u}}\{\boldsymbol{X} \in \mathcal{X}^*\} = \mathbb{E}_{\boldsymbol{u}} \left[\prod_{i=1}^{m_1} I_{(\sum_{k=1}^{n} a_{ik} X_k = b_i)} \prod_{j=m_1+1}^{m_1+m_2} I_{(\sum_{k=1}^{n} a_{jk} X_k \geq b_j)} \right], \tag{56}$$

which can be also written as (see Eq. 13) $\ell = \mathbb{E}_{\boldsymbol{u}} \left[I_{\{\sum_{i=1}^{m} C_i(\boldsymbol{X}) \geq m\}} \right]$.

It follows from the above that counting multiple events and in particular estimating the cardinality of the set (12) can be efficiently performed using the cloning Algorithm 6.2, that is one can estimate $|\mathcal{X}^*|$ via $|\widehat{\mathcal{X}^*}| = \widehat{\ell}|\mathcal{X}|$, where $\widehat{\ell}$ by itself is an estimator of ℓ given in Eq. 8. Similar arguments can be applied for estimating the volumes of bodies given by the set

$$\mathcal{X}^* = \{\boldsymbol{x} \in \mathbb{R}^n : S_i(\boldsymbol{x}) = b_i,\ i = 1, \dots, m_1;\ S_j(\boldsymbol{x}) \geq b_j,\ \ j = 1, \dots, m_2\}. \tag{57}$$

In particular, for a polyhedron, $\mathcal{X}^*$ reduces to the set associated with the following linear programming constraints

$$\begin{aligned} &\sum_{k=1}^{n} a_{ik} x_k = b_i,\ i = 1, \dots, m_1, \\ &\sum_{k=1}^{n} a_{jk} x_k \geq b_j,\ \ j = 1, \dots, m_2, \\ &x_k \in (a, b),\ \forall k = 1, \dots, n. \end{aligned} \tag{58}$$

Figure 2 presents the dynamics of the $(N = 1)$-policy algorithm to hit the polytope $1 \to 2 \to 3 \to 4 \to 5 \to 6$. The total number of samples required is $M = 4$. It is readily seen that while generating

1. The first (green) sample, the $(N = 1)$-policy algorithm hits the polytope $1 \to 2 \to 3 \to 4$.
2. The second (black) sample, it hits the polytope $1 \to 2 \to 3 \to 4 \to 6$.
3. The third (black) sample, it remains at the same polytope $1 \to 2 \to 3 \to 4 \to 6$.
4. The fourth (red) sample, it hits the desired polytope $1 \to 2 \to 3 \to 4 \to 5 \to 6$.

Example 6.1 (*SAT Example*) We shall show how to apply Algorithm 3.1 for counting the number of assignment in the following simple SAT problem

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

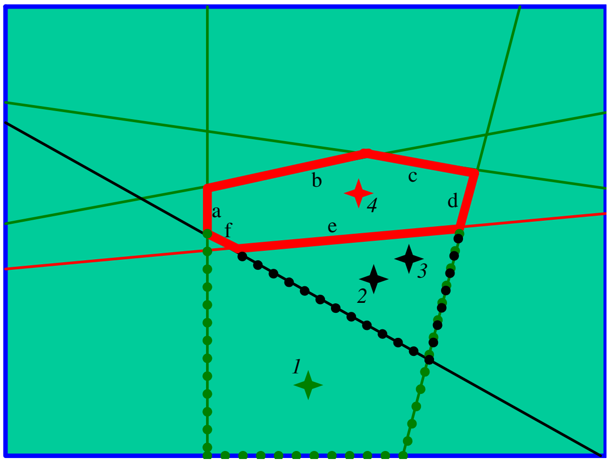


Fig. 2 Dynamics of the $(N = 1)$-policy algorithm to hit the polytope

Note that the associated set of linear integer constraints can be written as

$$x_1 + (1 - x_2) \geq 1$$

$$(1 - x_1) + (1 - x_2) + x_3 \geq 1,$$

$$x_2 + x_3 \geq 1,$$

where each $x_1, x_2, x_3 \in \{0, 1\}$. For more details on SAT problems see Section 9.

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

We have

$$\begin{aligned} g^*(x_1, \widehat{m}_{t-1} | \boldsymbol{x}_{-1}) &= f_1(x_1) I_{\{x_1 \geq \widehat{m}_{t-1} - I_{\{-x_2 \geq 0\}} - I_{\{x_2 - x_3 \leq 1\}} - C_3\}}, \\ g^*(x_2, \widehat{m}_{t-1} | \boldsymbol{x}_{-2}) &= f_2(x_2) I_{\{x_2 \geq \widehat{m}_{t-1} - I_{\{x_1 \geq 0\}} - I_{\{x_1 - x_3 \leq 1\}} - I_{\{x_3 \geq 1\}}\}}, \\ g^*(x_3, \widehat{m}_{t-1} | \boldsymbol{x}_{-3}) &= f_3(x_3) I_{\{x_1 \geq \widehat{m}_{t-1} - C_1 - I_{\{x_1 + x_2 \leq 1\}} - I_{\{x_2 \geq 1\}}\}}, \end{aligned} \tag{59}$$

where $f_i(x_i),\ i = 1, 2, 3$ are independent Ber(1/2) distributions.

Note that $I_{\{X_1 \geq 0\}} = 1$ in $g^*(x_2, \widehat{m}_{t-1} | \boldsymbol{x}_{-2})$.

It readily follows from the above example, that in order to count on a quite general set of linear (integer or continuous) constraints (12) with given matrix $\boldsymbol{A} = \{a_{ij}\}$, one only needs to apply the Gibbs method, while sampling from the following simple one-dimensional conditional pdfs

$$g^*(x_i, \widehat{m}_{t-1} | \boldsymbol{x}_{-i}) = \text{Ber}(1/2) I_{\left\{\sum_{r \in R_i} C_r(\boldsymbol{X}) \geq \widehat{m}_{t-1} - \sum_{r \notin R_i} C_r(\boldsymbol{X})\right\}}, \tag{60}$$

where $R_i = \{j : \ a_{ij} \neq 0\}$.

Recall that sampling a random variable $\widetilde{X}_i$ from Eq. 60 using the Gibbs sampler can be performed as follows. Generate $Y \sim \text{Ber}(1/2)$. If

$$\sum_{r \in R_i} C_r(x_1, \ldots, x_{i-1}, Y, x_{i+1}, \ldots, x_n) \geq \widehat{m}_{t-1},$$

then set $\widetilde{X}_i = Y$, otherwise set $\widetilde{X}_i = 1 - Y$.

Note also that before performing the simulation one has to store the corresponding set of indexes R_i associated with each conditional marginal pdf $g^*(x_i, \widehat{m}_{t-1} | \boldsymbol{x}_{-i})$.

7 Sampling Uniformly on Different Regions

Since the sequences produced by Algorithm 6.2 a uniform, so it should be suitable for uniform sampling on $\mathcal{X}^*$. This means, for example, that while counting the number of feasible solutions defined on the set of the linear integer program with the constraints (12), one can sample according to a *discrete uniform* pdf inside the corresponding region $\mathcal{X}^*$. As for another example, if we have linear programming constraints (58) instead of Eq. 12, then the continuous version (Rubinstein 2008b) of Algorithms 6.2 can be used to sample uniformly inside the corresponding polyhedron.

Since to sample on $\mathcal{X}^*$ only the elites at level $m_T = m$ matter, we can run the cloning Algorithms 6.2 at all intermediate levels $m_0, m-1, \dots, m_{T-1}$ with less samples, while at the final level, $m_T = m$, we can take a larger sample. The corresponding elites could be further employed to generate an approximate uniform sampling on the entire region $\mathcal{X}^*$.

As a simple example, consider sampling on the region

$$\mathcal{X}^* = \left\{ \boldsymbol{X} : \sum_{i=1}^{n} X_i \geq m \right\},$$

where the X_i's are iid each distributed Ber (1/2). Let $n = 100$ and $m = 99$. We have $|\mathcal{X}^*| = 101$. For this example we employed Algorithm 6.2 with $N = 100$, $\rho = 0.1$ and $b = 1$ for the first $T-1$ iterations. By doing so we obtain at each iteration, on average, ten uniform elites. At the last iteration we increased the sample to 1,000 using the same $\rho = 0.1$ and the same $b = 1$ and thus obtained on average, 100 truly uniform elites. We finally run each of these 100 Markov chains (in steady-state) for a long burn-in, say $b = 50$ and, thus generating a total of $N = 5{,}000 = 100 \times 50$ Gibbs samples, for which statistics was collected. For this simple example we found that

1. The direct estimator $|\widehat{\mathcal{X}^*_{\text{dir}}}|$ with the above sample $N = 5,000$ found all 101 different Bernoulli points.
2. The resulting sample is distributed approximately uniformly on the set $\mathcal{X}^* = \left\{ \boldsymbol{X} : \sum_{i=1}^{100} X_i \geq 99 \right\}$, that is its histogram over these 101 points satisfies uniformity in the sense that it passes successfully the χ-square statistical test.

For a relevant paper see Botev (2007). More research on uniform sampling on different regions is under way.

8 Combining Cloning with CE: The CLONCE Algorithms

Here we show that the efficiency of the cloning method can be often improved while combining it with CE.

Consider again as a fixed sequence of $\{c_t,\ t = 0, 1, \dots, T\}$, given in Eq. 6, that is

$$c_t = \frac{\mathbb{E}_f[I_{\{S(\boldsymbol{X}) \geq m_t\}}]}{\mathbb{E}_f[I_{\{S(\boldsymbol{X}) \geq m_{t-1}\}}]} = \mathbb{E}_{g^*_{t-1}}[I_{\{S(\boldsymbol{X}) \geq m_t\}}]. \tag{61}$$

Recall that $g^*_{t-1} = g^*(\boldsymbol{x}, \boldsymbol{p}, m_{t-1})$ denotes the IS pdf for *fixed* $\boldsymbol{p}$ in $f(\boldsymbol{x}, \boldsymbol{p})$.

We shall show next how to get better estimators of c_t than these given in Eq. 9. To do so we shall apply an additional change of measure, while estimating the c_t's. In particular we shall introduce a sequence of parametric families $\{g^*_{t-1}(\boldsymbol{p}^*_t) = g^*(\boldsymbol{x}, \boldsymbol{p}^*_t, m_{t-1})\}$ instead of just $g^*_{t-1}(\boldsymbol{p}) = g^*(\boldsymbol{x}, \boldsymbol{p}, m_{t-1})$. Note that in the latter pdf the parameter vector $\boldsymbol{p}$ is *fixed*, while in the former it varies; the parameter vector $\boldsymbol{p}^*_t = \boldsymbol{p}^*(m_t)$ *will be updated adaptively at each level* m_t, say by using the CE method. We shall show numerically that by doing so the efficiency of Algorithm 6.2 can be often improved. In particular it is shown in Section 10.1.1 that for the sum of iid Bernoulli random variables with $p = 1/2$ the average complexity (the average number of samples) of the $(N = 1)$-policy algorithms to reach the set $\mathcal{X}^* = \{\boldsymbol{x} : \sum_{k=m}^{n} X_k \geq m\}$, is $R_m = O(\ln m)$ as compared to $R_m = O(n \ln \frac{n}{n-m})$ without using CE. However, in general, one has take into account the degeneracy property (Rubinstein and Kroese 2007) of the likelihood ratios, while combining cloning with CE. In fact, in high dimensions, the effect of CE can be even negative.

To this end, in analogy to the pdf $g_t^*(\boldsymbol{p}) = g^*(\boldsymbol{x}, \boldsymbol{p}, m_t)$ (see Eq. 7) define

$$g_t^{**} = g^*(\boldsymbol{x}, \boldsymbol{p}_t^*, m_t) = \frac{1}{\ell_t^*(\boldsymbol{p}_t^*)} f(\boldsymbol{x}, \boldsymbol{p}_t^*) I_{\{S(\boldsymbol{x}) \geq m_t\}}, \tag{62}$$

where $\ell_t^* = \mathbb{E}_{\boldsymbol{p}_t^*}\left[I_{\{S(\boldsymbol{X}) \geq m_t\}}\right]$ denotes the normalization constant under the pdf $f(\boldsymbol{x}, \boldsymbol{p}_{t-1}^*)$. Thus, an estimator of $\ell(m)$ can be obtained by taking again as the product of c_t's, where

$$c_t = \mathbb{E}_{g_{t-1}^{**}}\left[I_{\{S(\boldsymbol{X}) \geq m_t\}} W_{**}(\boldsymbol{X}, \boldsymbol{p}, \boldsymbol{p}_{t-1}^*)\right], \tag{63}$$

$\mathbb{E}_{g_{t-1}^{**}}$ denotes the expectation with respect to $g_{t-1}^{**} = g^*(\boldsymbol{x},\ m_{t-1},\ \boldsymbol{p}_{t-1}^*)$ and

$$W_{**}(\boldsymbol{X}, \boldsymbol{p}, \boldsymbol{p}_{t-1}^*) = \frac{g^*(\boldsymbol{X}, \boldsymbol{p}, m_{t-1})}{g^*(\boldsymbol{X}, \boldsymbol{p}_{t-1}^*, m_{t-1})} = \frac{\ell_t^*}{\ell_t} \frac{f(\boldsymbol{X}, \boldsymbol{p})}{f(\boldsymbol{X}, \boldsymbol{p}_{t-1}^*)} \tag{64}$$

is the *likelihood ratio* (LR). Substituting Eqs. 64 into 63 we obtain

$$c_t = \frac{\ell_{t-1}^*}{\ell_{t-1}} \mathbb{E}_{g_{t-1}^{**}}\left[I_{\{S(\boldsymbol{X}) \geq m_t\}} \frac{f(\boldsymbol{X}, \boldsymbol{p})}{f(\boldsymbol{X}, \boldsymbol{p}_{t-1}^*)}\right]. \tag{65}$$

But $\frac{\ell_{t-1}^*}{\ell_{t-1}}$ is unknown, so c_t can not be estimated directly from Eq. 65.

To overcome this difficulty we shall use the following well known trick. Taking into account that $\mathbb{E}_{g_{t-1}^{**}}[W_{**}(\boldsymbol{X}, \boldsymbol{p}, \boldsymbol{p}_{t-1}^*)] = 1$, we can write c_t in Eq. 63 as

$$c_t = \frac{\mathbb{E}_{g_{t-1}^{**}}\left[I_{\{S(\boldsymbol{X}) \geq m_t\}} W_{**}(\boldsymbol{X}, \boldsymbol{p}, \boldsymbol{p}_{t-1}^*)\right]}{\mathbb{E}_{g_{t-1}^{**}}\left[W_{**}(\boldsymbol{X}, \boldsymbol{p}, \boldsymbol{p}_{t-1}^*)\right]}. \tag{66}$$

Now substituting Eq. 64 into Eq. 66 we finally obtain

$$c_t = \frac{\mathbb{E}_{g_{t-1}^{**}}\left[I_{\{S(\boldsymbol{X}) \geq m_t\}} \frac{f(\boldsymbol{X}, \boldsymbol{p})}{f(\boldsymbol{X}, \boldsymbol{p}_{t-1}^*)}\right]}{\mathbb{E}_{g_{t-1}^{**}}\left[\frac{f(\boldsymbol{X}, \boldsymbol{p})}{f(\boldsymbol{X}, \boldsymbol{p}_{t-1}^*)}\right]}. \tag{67}$$

Observe that c_t in Eq. 67 does not contain the term $\frac{\ell^*_{t-1}}{\ell_{t-1}}$ any more, so it can be estimated as

$$\widetilde{c}_t = \frac{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i)\geq m_t\}} W(\boldsymbol{X}_i,\ \boldsymbol{p},\ \widehat{\boldsymbol{p}}^*_{t-1})}{\sum_{i=1}^{N} W(\boldsymbol{X}_i,\ \boldsymbol{p},\ \widehat{\boldsymbol{p}}^*_{t-1})} = \frac{\sum_{i=1}^{N_t} W(\boldsymbol{X}_i,\ \boldsymbol{p},\ \widehat{\boldsymbol{p}}^*_{t-1})}{\sum_{i=1}^{N} W(\boldsymbol{X}_i,\ \boldsymbol{p},\ \widehat{\boldsymbol{p}}^*_{t-1})}, \tag{68}$$

where $\boldsymbol{X}_i \sim g^{**}_{t-1},\ N_t$ is the number of elites and

$$W(\boldsymbol{X}_i,\ \boldsymbol{p},\ \widetilde{\boldsymbol{p}}^*_{t-1}) = \frac{f(\boldsymbol{X}_i,\ \boldsymbol{p})}{f(\boldsymbol{X}_i,\ \widetilde{\boldsymbol{p}}^*_{t-1})}. \tag{69}$$

Finally, we can estimate $\ell(m)$ as

$$\widetilde{\ell}(m) = \prod_{t=1}^{T} \widetilde{c}_t = \prod_{t=1}^{T} \frac{\sum_{i=1}^{N_t} W(\boldsymbol{X}_i,\ \boldsymbol{p},\ \widehat{\boldsymbol{p}}^*_{t-1})}{\sum_{i=1}^{N} W(\boldsymbol{X}_i,\ \boldsymbol{p},\ \widehat{\boldsymbol{p}}^*_{t-1})}. \tag{70}$$

As mentioned the sequence $\{\boldsymbol{p}^*_{t-1}\}$ and $\{\widetilde{\boldsymbol{p}}^*_{t-1}\}$ can be obtained by applying the standard CE method to $\mathbb{E}_{g^*_{t-1}}[I_{\{S(\boldsymbol{X})\geq m_t\}}]$. In particular the components of $\boldsymbol{p}^*_{t-1}$ can be written as

$$p^*_{t,k} = \frac{\mathbb{E}_{g^{**}_{t-1}}\left[X_k I_{\{S(\boldsymbol{X})\geq m_t\}} W_{**}(\boldsymbol{X},\ \boldsymbol{p},\ \boldsymbol{p}^*_{t-1})\right]}{\mathbb{E}_{g^{**}_{t-1}}\left[I_{\{S(\boldsymbol{X})\geq m_t\}} W_{**}(\boldsymbol{X},\ \boldsymbol{p},\ \boldsymbol{p}^*_{t-1})\right]}, \tag{71}$$

where, as before, $\boldsymbol{X} \sim g^{**}_{t-1}$ and $\boldsymbol{p}$ denotes the original parameter vector. Arguing similarly to Eq. 67 it can be simplified as

$$p^*_{t,k} = \frac{\mathbb{E}_{g^{**}_{t-1}}\left[X_k I_{\{S(\boldsymbol{X})\geq m_t\}} \frac{f(\boldsymbol{X},\boldsymbol{p})}{f(\boldsymbol{X},\boldsymbol{p}^*_{t-1})}\right]}{\mathbb{E}_{g^{**}_{t-1}}\left[I_{\{S(\boldsymbol{X})\geq m_t\}} \frac{f(\boldsymbol{X},\boldsymbol{p})}{f(\boldsymbol{X},\boldsymbol{p}^*_{t-1})}\right]}, \tag{72}$$

The final CE estimator of Eq. 72 can be written as

$$\widehat{p}^*_{t,k} = \frac{\sum_{i=1}^{N} X_{ki} I_{\{S(\boldsymbol{X}_i)\geq \widehat{m}_t\}} W(\boldsymbol{X}_i,\ \boldsymbol{p},\ \widehat{\boldsymbol{p}}^*_{t-1})}{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i)\geq \widehat{m}_t\}} W(\boldsymbol{X}_i,\ \boldsymbol{p},\ \widehat{\boldsymbol{p}}^*_{t-1})}, \tag{73}$$

where, as before $\boldsymbol{X}_i \sim g^{**}_{t-1}$ and $W(\boldsymbol{X}_i,\ \boldsymbol{p},\ \widehat{\boldsymbol{p}}^*_{t-1})$ is given in Eq. 69.

It is important to not that for optimization problems there is no need for the LR $W(\boldsymbol{X}_i,\ \boldsymbol{p},\ \widehat{\boldsymbol{p}}^*_{t-1})$ and thus Eq. 73 can be simplified as

$$\widetilde{p}^*_{t,k} = \frac{\sum_{i=1}^{N} X_{ki} I_{\{S(\boldsymbol{X}_i)\geq \widehat{m}_t\}}}{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i)\geq \widehat{m}_t\}}}. \tag{74}$$

Below we present the combined cloning-CE algorithm, which we call the *CLONCE* algorithm and which differs from the original Algorithm 6.2 that it contains two additional steps: (1) an adaptive CE step for updating the vector $\widehat{\boldsymbol{p}}^*$ according to

Eq. 73 and (2) the standard smoothing step with injection (Rubinstein and Kroese 2007). The essential difference is that at each step we use the Gibbs sampler to generate from $g^*(\boldsymbol{x},\ \widehat{\boldsymbol{p}}^*_{t-1},\ \widehat{m}_{t-1})$ rather than from $g^*(\boldsymbol{x},\ \boldsymbol{p},\ \widehat{m}_{t-1})$.

Remark 8.1 Since the cloning part is the major one in the CLONCE Algorithm 8.1, we shall further restrict the components $\widetilde{p}^*_{t,k}$ of the vector $\widetilde{\boldsymbol{p}}^*_t$ by $b_1 \le \widetilde{p}^*_{t,k} \le b_2$, where say $b_1 = 0.1$ and $b_2 = 0.9$. This can be done by choosing say $\beta = 0.1$. Without this constraint, that is allowing $0 \le \widetilde{p}^*_{t,k} \le 1$, the CE part might become the major one. As result, the CLONCE Algorithm 8.1 might converge to a local optimum.

Algorithm 8.1 (The CLONCE Algorithm for Counting)

Given the proposal rarity parameter ρ, say $\rho = 0.1$, the parameters a_1 and a_2, say $a_1 = 0.01$ and $a_2 = 0.25$, such that $\rho \in (a_1, a_2)$, and the sample size N, say $N = m \times n$, execute the following steps:

1. **Acceptance–Rejection**—the same as in Algorithm 6.1.
2. **Adaptive choice of $\widehat{\boldsymbol{p}}^*_{t-1}$** Generate a sample $\boldsymbol{X}_1, \dots, \boldsymbol{X}_N$ from the pdf $g^*(\boldsymbol{x},\ \widehat{\boldsymbol{p}}^*_{t-1},\ \widehat{m}_{t-1})$ and compute the components of $\widehat{\boldsymbol{p}}^*_t$ according to Eq. 73.
3. **Smoothing** Smooth out the vector $\widehat{\boldsymbol{p}}^*_t$ according to
$$\widetilde{\boldsymbol{p}}_t = \alpha\widehat{\boldsymbol{p}}_t + \beta\boldsymbol{p}_0 + (1-\alpha-\beta)\widetilde{\boldsymbol{p}}_{t-1}, \tag{75}$$
where α, $(0 < \alpha < 1)$ is called the *smoothing* parameter and β is called, the *injection* parameter. Typically $0.5 \le \alpha \le 0.9$ and $0.05 \le \beta \le 0.2$.
4. **Adaptive choice of ρ**—the same as in Algorithm 6.2.
5. **Screening**—the same as in Algorithm 6.2.
6. **Cloning** Given the size N_{t-1} of screened elites at iteration $(t-1)$, find the cloning and the burn-in parameters η_{t-1} and b_{t-1} according to Eq. 27. Reproduce η_{t-1} times each vector $\widehat{\boldsymbol{X}}_k = (\widehat{X}_{1k}, \dots, \widehat{X}_{nk})$ of the screened elite sample $\{\widehat{\boldsymbol{X}}_1, \dots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$, that is, take η_{t-1} identical copies of each vector $\widehat{\boldsymbol{X}}_k$ obtained at the $(t-1)$-th iteration. Denote the entire new population ($\eta_{t-1}N_{t-1}$ cloned vectors plus the original screened elite sample $\{\widehat{\boldsymbol{X}}_1, \dots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$) by $\mathcal{X}_{cl} = \{(\widehat{\boldsymbol{X}}_1, \dots, \widehat{\boldsymbol{X}}_1), \dots, (\widehat{\boldsymbol{X}}_{N_{t-1}}, \dots, \widehat{\boldsymbol{X}}_{N_{t-1}})\}$. To each of the cloned vectors of the population $\mathcal{X}_{cl}$ apply the MCMC (and in particular the Gibbs sampler) for b_{t-1} burn-in periods. Denote the *new entire* population by $\{\boldsymbol{X}_1, \dots, \boldsymbol{X}_N\}$. Observe that each member of $\{\boldsymbol{X}_1, \dots, \boldsymbol{X}_N\}$ is distributed approximately $g^*(\boldsymbol{x}, \widehat{\boldsymbol{p}}^*_{t-1}, \widehat{m}_{t-1})$.
7. **Estimating c_t** Let $\widetilde{\mathcal{X}}_t = \{\widetilde{\boldsymbol{X}}_1, \dots, \widetilde{\boldsymbol{X}}_{N_t}\}$ be the subset of the population $\{\boldsymbol{X}_1, \dots, \boldsymbol{X}_N\}$ for which $S(\boldsymbol{X}_i) \ge \widehat{m}_t$. Take $\widetilde{c}_t$ in Eq. 68 as an estimator of c_t given in Eq. 63. Note again as that $\widetilde{\boldsymbol{X}}_1, \dots, \widetilde{\boldsymbol{X}}_{N_t}$ is distributed approximately $g^*(\boldsymbol{x},\ \widehat{\boldsymbol{p}}^*_t,\ \widehat{m}_t)$.
8. **Stopping rule** If $m_t = m$ go to step 9, otherwise set, set $t = t+1$ and repeat from step 2.
9. **Final Estimator** Deliver $\widetilde{\ell}(m)$ given in Eq. 70 as an estimator of $\ell(m)$ and $|\widetilde{\mathcal{X}}^*| = \widetilde{\ell}(m)|\mathcal{X}|$ as an estimator of $|\mathcal{X}^*|$.

Tables 3 and 4 presents the dynamics of the cloning Algorithm 6.2 and the CLONCE Algorithm 8.1, respectively for the sum of $n = 100$ Bernoulli rv's, with $m = 99,\ p = 0.5,\ \rho = 0.05$ and $N = 10{,}000$ samples. One can clearly see that

Table 3 Dynamics of the cloning Algorithm 6.2 for the sum of $n = 100$ Bernoulli rv's, with $m = 99$, $p = 0.5$, $\rho = 0.05$ and $N = 10{,}000$ samples

t	$\lvert\widetilde{\mathcal{X}}^*\rvert$	$\lvert\widetilde{\mathcal{X}}^*_{\text{dir}}\rvert$	N_t	$N_t^{(s)}$	m_t^*	m_{*t}	ρ_t
1	8.56e+028	6.75e+002	675	675	69	58	0.07
3	1.07e+027	1.52e+003	1,519	1,519	74	66	0.15
5	4.83e+025	2.40e+003	5,596	2,399	78	70	0.19
7	9.81e+023	1.46e+003	4,230	1,463	79	74	0.13
9	9.58e+021	9.00e+002	900	900	82	78	0.09
11	1.76e+020	2.59e+003	2,586	2,586	86	81	0.24
13	8.53e+018	2.50e+003	2,500	2,500	88	83	0.21
15	3.09e+017	2.18e+003	2,175	2,175	89	85	0.19
17	8.52e+015	1.80e+003	11,340	1,798	90	87	0.16
19	1.68e+014	1.49e+003	11,032	1,493	92	89	0.14
21	2.23e+012	1.11e+003	1,108	1,108	94	91	0.11
23	1.79e+010	8.88e+002	10,630	888	95	93	0.08
25	7.89e+007	6.11e+002	611	611	97	95	0.06
27	1.71e+005	4.48e+002	10,420	448	98	97	0.04
29	1.16e+002	8.90e+001	10,240	89	100	99	0.02
30	1.16e+002	1.01e+002	10,235	101	100	99	1.00
31	1.16e+002	1.01e+002	10,100	101	100	99	1.00

CLONCE Algorithm is faster. We also found that CLONCE is the most accurate among the two.

In Tables 3 and 4 we used the following notations

1. N_t and $N_t^{(s)}$ denotes the actual number of elites and the one after screening, respectively.
2. m_t^* and m_{*t} denotes the upper and the lower elite levels reached, respectively.
3. $\rho_t = N_t/N$ denotes the adaptive proposal rarity parameter.

Below, for completeness, we present the CLONCE algorithm for optimization with the objective function given in Eq. 36. Unlike Algorithm 5.1 we also kept here the screening step introduced in our counting algorithms to produce screened samples at each step. The cloning Algorithm 5.1 follow automatically from the CLONCE one by removing the CE and the screening steps from it. As in Algorithm 5.1 we shall use formula (26) for choosing η and b.

Table 4 Dynamics of the CLONCE Algorithm 8.1 for the sum of $n = 100$ Bernoulli rv's, with $m = 99$, $p = 0.5$, $\rho = 0.05$ and $N = 10{,}000$ samples

t	$\lvert\widetilde{\mathcal{X}}^*\rvert$	$\lvert\widetilde{\mathcal{X}}^*_{\text{dir}}\rvert$	N_t	$N_t^{(s)}$	m_t^*	m_{*t}	ρ_t
1	9.19e+028	7.25e+002	725	725	68	58	0.07
2	1.21e+027	8.66e+002	866	866	76	66	0.09
3	8.66e+024	1.33e+003	1,326	1,326	82	72	0.13
4	1.15e+022	1.15e+003	1,150	1,150	88	78	0.11
5	9.36e+018	1.33e+003	1,334	1,334	90	83	0.13
6	8.85e+015	1.81e+003	1,814	1,814	96	87	0.17
7	2.31e+012	1.53e+003	1,527	1,527	97	91	0.14
8	1.42e+009	1.96e+003	1,958	1,958	99	94	0.18
9	1.86e+005	1.26e+003	3,285	1,264	100	97	0.11
10	1.12e+002	1.01e+002	1,317	101	100	99	0.13
11	1.12e+002	1.01e+002	10,100	101	100	99	1.00

Algorithm 8.2 (The CLONCE Algorithm for Optimization)

Given the proposal rarity parameter ρ, say $\rho = 0.1$, the parameters b_1 and b_2, say $b_1 = 0.1$ and $b_2 = 0.9$, such that $\rho \in (b_1, b_2)$, the sample size N, say $N = m \times n$, and the burn in period b, say $3 \leq b \leq 10$ execute the following steps:

1. **Acceptance–Rejection** - the same as in Algorithm 6.1.
2. **Adaptive choice of $\widehat{\boldsymbol{p}}^*_{t-1}$** Generate a sample $\boldsymbol{X}_1, \dots, \boldsymbol{X}_N$ from the pdf $g^*(\boldsymbol{x},\ \widehat{\boldsymbol{p}}^*_{t-1},\ \widehat{m}_{t-1})$ and compute the components of $\widehat{\boldsymbol{p}}^*_t$ according to Eq. 74.
3. **Smoothing** Smooth out the vector $\widehat{\boldsymbol{p}}^*_t$ according to Eq. 75.
4. **Screening**—the same as in Algorithm 6.2.
5. **Cloning** Given the number of burn in periods b and the size N_{t-1} of screened elites at iteration $(t-1)$, find the cloning parameter η_{t-1} according to $\eta_{t-1} = \left\lceil \frac{N}{b\, N_{t-1}} \right\rceil - 1$. Reproduce each vector $\widehat{\boldsymbol{X}}_k = (\widehat{X}_{1k}, \dots, \widehat{X}_{nk})$ of the screened elite sample $\{\widehat{\boldsymbol{X}}_1, \dots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$ η_{t-1} times, that is take η_{t-1} identical copies of each vector $\widehat{\boldsymbol{X}}_k$ obtained at the $(t-1)$-th iteration. Denote the entire new population ($\eta_{t-1} N_{t-1}$ cloned vectors plus the original screened elite sample $\{\widehat{\boldsymbol{X}}_1, \dots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$) by $\mathcal{X}_{cl} = \{(\widehat{\boldsymbol{X}}_1, \dots, \widehat{\boldsymbol{X}}_1), \dots, (\widehat{\boldsymbol{X}}_{N_{t-1}}, \dots, \widehat{\boldsymbol{X}}_{N_{t-1}})\}$. To each of the cloned vectors of the population $\mathcal{X}_{cl}$ apply the MCMC (and in particular the Gibbs sampler) for b_{t-1} burn-in periods. Denote the *new entire* population by $\{\boldsymbol{X}_1, \dots, \boldsymbol{X}_N\}$. Observe that each member of $\{\boldsymbol{X}_1, \dots, \boldsymbol{X}_N\}$ is distributed approximately $g^*(\boldsymbol{x},\ \widehat{\boldsymbol{p}}^*_{t-1},\ \widehat{m}_{t-1})$.
6. **Estimating $\boldsymbol{m_t}$** Let $\widetilde{\mathcal{X}}_t = \{\widetilde{\boldsymbol{X}}_1, \dots, \widetilde{\boldsymbol{X}}_{N_t}\}$ corresponds to the largest $(1-\rho)\%$ subset of the population $\{\boldsymbol{X}_1, \dots, \boldsymbol{X}_N\}$ for which $S(\boldsymbol{X}_i) \geq \widehat{m}_t$. Deliver $\widehat{m}_t$. Note again as that $\widetilde{\boldsymbol{X}}_1, \dots, \widetilde{\boldsymbol{X}}_{N_t}$ is distributed approximately $g^*(\boldsymbol{x},\ \widehat{\boldsymbol{p}}^*_t,\ \widehat{m}_t)$.
7. **Stopping rule**—the same as in Algorithm 6.2.

9 Applications to Optimization and Counting

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

Knapsack Problem We consider here well known, so-called *multiple knapsack* problem, which reads as

$$\begin{aligned}
&\max \sum_{i=1}^{m} \sum_{k=1}^{n} c_k x_{ik} \\
&\text{s.t.} \sum_{k=1}^{n} a_k x_{ik} \leq b_i,\ \forall\text{i=1}, \dots, \text{m} \\
&\sum_{i=1}^{m} x_{ik} \leq 1,\ \forall k = 1, \dots, n \\
&x_{ik} \in \{0, 1\}, , \forall i = 1, \dots, m;\ k = 1, \dots, n.
\end{aligned} \tag{76}$$

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

Set Covering, Set Packing and Set Partitioning Note that set partition program reduces to the program Eq. 55, provided A is a 0–1 matrix, $x_i \in \{0, 1\},\ \forall i = 1, \dots, n$ and the vector b $= 1$, provided $m_2 = 0$ and the minimization is replaced by maximization. The set covering and set packing problems are similar to the set partition one, provided the equality constraints $\boldsymbol{Ax} = \mathbf{1}$ is replaced by $\boldsymbol{Ax} \geq \mathbf{1}$ and $\boldsymbol{Ax} \leq \mathbf{1}$, respectively.

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

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

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

and

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

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

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

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

and

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

respectively.

The SAT Problem Because of the importance of the SAT problem we recapitulate some material from Rubinstein et al. (2007). The most common SAT problem comprises the following two components:

- A set of n Boolean variables $\{x_1, \dots, x_n\}$, representing statements that can either be TRUE (=1) or FALSE (=0). The negation (the logical NOT) of a variable x is denoted by $\overline{x}$. For example, $\overline{\text{TRUE}} = \text{FALSE}$. A variable or its negation is called a *literal*.
- A set of m distinct *clauses* $\{S_1, S_2, \dots, S_m\}$ of the form $S_i = z_{i_1} \vee z_{i_2} \vee \cdots \vee z_{i_k}$, where the z's are literals and the $\vee$ denotes the logical OR operator. For example, $0 \vee 1 = 1$.

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

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

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

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

The problem of deciding whether there *exists* a valid assignment, and, indeed, providing such a vector, is called the *SAT-assignment* problem (Rubinstein et al. 2007).

It is shown in Rubinstein et al. (2007) that the SAT-assignment problem can be modeled via rare-events with ℓ given in Eq. 53, that is,

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

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

$$C_i(\boldsymbol{x}) = \max_j \{0,\ (2\,x_j - 1)\, a_{ij}\}. \tag{77}$$

Here $C_i(\boldsymbol{x}) = 1$ if clause S_i is TRUE with truth assignment $\boldsymbol{x}$ and $C_i(\boldsymbol{x}) = 0$ if it is FALSE, $A = (a_{ij})$ is a given clause matrix that indicate if the literal corresponds to the variable (+1) , its negation (−1), or that neither appears in the clause (0). If for example $x_j = 0$ and $a_{ij} = -1$, then the literal $\overline{x}_j$ is TRUE. The entire clause is TRUE if it contains at least one true literal. In other words, ℓ in Eq. 53 is the probability that a uniformly generated SAT assignment (trajectory) $\boldsymbol{X}$ is valid, that is, all clauses are satisfied, which is typically very small.

Counting 0–1 Tables with Fixed Margins In this case the set $\{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$ is given by

$$\sum_{i=1}^{n} x_{ij} = b_j^{(1)}, \;\; j = 1, \ldots, m$$
$$\sum_{j=1}^{m} x_{ij} = b_i^{(2)}, \;\; i = 1, \ldots, n$$
$$x_{ij} \in \{0, 1\}, \;\; \forall i, j. \tag{78}$$

The goal is, for given $b_j^{(1)}$, $j = 1, \ldots, m$ and $b_i^{(2)}$, $i = 1, \ldots, n$ to find the binary matrix (x_{ij}) satisfying Eq. 78 and to count the total number of 0–1 tables. Note that if, for example A is 20×20 matrix, then $n = 20^2 = 400$ and $m = 2 \cdot 20 = 40$.

10 Complexity Properties

We consider in this section the complexity properties of the $(N = 1)$-policy algorithm and of the direct estimator.

10.1 Complexity Properties of the $(N = 1)$-policy Algorithm

Recall that according to the $(N = 1)$-policy algorithm, at each level m_{t-1} we use the Gibbs sampler until we reach for the first time the next level m_t. By complexity of the $(N = 1)$-policy algorithm we mean here the *average complexity* (the average number of required samples), denoted as $R_m = \mathbb{E}(\mathcal{N}_m)$ and the *associated variance* $\sigma_m^2 = \mathrm{Var}(\mathcal{N}_m)$ to hit the desired level m. Here $\mathcal{N}_m$ denotes the total number of trials required to reach the level m. We consider the complexity for the following two basic cases: the sum of iid Bernoulli random variables and for multiple events associated with the sample objective function $S(\boldsymbol{X}) = \sum_{i=1}^{m} C_i(\boldsymbol{X})$, where the $C_i(\boldsymbol{X})$'s are given in Eqs. 14, 15. While deriving the complexity we shall assume that the Gibbs sampler is *perfect*, that is at every level m_t we are able to generate samples from the corresponding IS pdf g_{t-1}^* at level m_{t-1}.

10.1.1 Sum of Independent Bernoulli Random Variables

Consider estimation of

$$\ell = \mathbb{E}_f\left[I_{\{\sum_{i=1}^{n} X_i \geq m\}}\right], \tag{79}$$

where X_i are iid $\mathrm{Ber}(p)$ random variables. We have

$$\ell = \sum_{k=m}^{n} \binom{n}{k} p^k (1-p)^{n-k}. \tag{80}$$

As before, let $\{c_t, \; t = 0, 1, \ldots, T\}$ be a fixed sequence given in Eq. 6, which can be also written as

$$c_t = \frac{\mathbb{E}_f\left[I_{\{\sum_{k=1}^{n} X_k \geq m_t\}}\right]}{\mathbb{E}_f\left[I_{\{\sum_{k=1}^{n} X_k \geq m_{t-1}\}}\right]}. \tag{81}$$

We consider separately two cases: (1) $p = 1/2$ and (2) $p \neq 1/2$.

(1) $p = 1/2$.

Lemma 10.1 *The average number of samples $R_m = \mathbb{E}(\mathcal{N}_m)$ required by the $(N = 1)$-policy algorithm to reach the level m, while estimating ℓ in Eq. 79 with $p = 1/2$ is at most*

$$R_m = (n+1)\ln\frac{n}{n+1-m}. \tag{82}$$

Note that R_m in Eq. 82, which is also called the *average complexity* of the $(N = 1)$-policy algorithm, means that in order to reach, say the level $m \approx n$ for $n = 100$ and $n = 1{,}000$ we need at most $R_m = 460$ and $R_m = 6908$ samples, respectively. Our numerical results below support these numbers.

Proof To prove Eq. 82 consider a sequence of levels $\{m_0, m_1, \ldots, m_t, \ldots, m\}$, where m_t denotes the number of ones at the Bernoulli trials at the iteration t. Note that for large n, while flipping n symmetric coins we expect to obtain on average $m_0 = n/2$ ones at the first iteration of the algorithm. We shall, however, remove the condition $m_0 = n/2$, assuming that m_0 is any fixed integer, say $m_0 = 0$ and, thus consider the worst case.

We proceed by showing that while starting from any fixed m_0, a *typical* sequence $\{m_0, m_1, \ldots, m_t, \ldots, m\}$ generated by the $(N = 1)$-policy algorithm will be the following one $\{m_0,\ m_0+1,\ m_0+2, \ldots, m\}$, while any alternative sequence, like the one $\{m_0,\ m_0+r_1,\ m_0+r_2, \ldots,\ m_0+r_t, \ldots, m\}$, where $r_t > 1$, will be *not a typical* one. To see this observe that for fixed m_t, the average number of Bernoulli trials to reach the level $m_t = m_{t-1}+1$ (see Eq. 81) is

$$\begin{aligned}
1/c_t &= \frac{\mathbb{E}_f[I_{\{\sum_{k=1}^n X_k \geq m_{t-1}\}}]}{\mathbb{E}_f[I_{\{\sum_{k=1}^n X_k \geq m_{t-1}+1\}}]} = \frac{\sum_{k=m_{t-1}}^n \binom{n}{k}}{\sum_{k=m_{t-1}+1}^n \binom{n}{k}} \\
&= \frac{\sum_{k=m_{t-1}}^n \frac{n!}{k!(n-k)!}}{\sum_{k=m_{t-1}+1}^n \frac{n!}{k!(n-k)!}} \\
&= 1 + \frac{\frac{n!}{m_{t-1}!(n-m_{t-1})!}}{\sum_{k=m_{t-1}+1}^n \frac{n!}{k!(n-k)!}} \leq 1 + \frac{m_{t-1}+1}{n-m_{t-1}}.
\end{aligned} \tag{83}$$

For example, for $n = 100$ and $m_{t-1} = 1,\ 50,\ 97$ we obtain

$$1/c_t - 1 \leq 2/99,\ 51/50,\ 98/3.$$

Observe that the probability of remaining at the same level m_{t-1} (rejecting the trial) is $1 - c_t$.

Observe also that for $m_t = m_{t-1}+2$, $(r = 2)$ we have

$$1/c_t \leq 1 + \frac{(m_{t-1}+1)(m_{t-1}+2)}{(n-m_{t-1})(n-m_{t-1}-1)}. \tag{84}$$

In particular, for $n = 100$ and $m_{t-1} = 1,\ 50,\ 97$ we obtain that

$$1/c_t - 1 \leq \frac{2}{99}\frac{3}{98},\ \frac{51}{50}\frac{52}{49},\ \frac{98}{3}\frac{99}{2}.$$

Clearly, that for m close to n, the quantity $1/c_t(r)$ in Eq. 84 is much smaller than that $c_t(1)$ in Eq. 83 and, thus the sequence $\{m_0,\ m_0 + r_1,\ m_0 + r_2, \ldots,\ m_0 + r_t, \ldots, m\}$ is not a typical one, provided m is close to n and $r_t > 1$. This in turn implies that while using the $(N = 1)$-policy it is much more likely that (starting at some $m_{t-1} > n/2$) we shall climb level by level, rather than skipping some levels before we reach the level m. Note again that by considering only the case $r = 1$ and by removing the all (non typical) cases with $r > 1$ we can only increase the average complexity R_m.

To proceed note that the average complexity R_m can be written as

$$R_m = \sum_{t=m_0}^{m} \frac{1}{c_t} \leq \sum_{t=0}^{m} \frac{1}{c_t}, \tag{85}$$

where $1/c_t$ given in Eq. 83. Taking into account that $1/c_t$ can be written as

$$1/c_t = 1 + \frac{m_{t-1} + 1}{n - m_{t-1}} = \frac{n + 1}{n - m_{t-1}} \tag{86}$$

we can rewrite R_m as

$$R_m \leq \sum_{t=0}^{m} \frac{n+1}{n+1-t}.$$

For large m the above sum can be approximated by the harmonic series, resulting in

$$R_m \leq \sum_{t=0}^{m} \frac{n+1}{n+1-t} \approx (n+1) \ln \frac{n}{n+1-m} = O\left(n \ln \frac{n}{n-m}\right). \tag{87}$$

□

Remark 10.1 The hazard rate approach We shall show now that the expression (87) of Lemma 10.1 can be derived by using the classic hazard rate approach described in Appendix. Note that for a discrete random variable the hazard rate is defined in Eq. 116, that is

$$\lambda(t) = \mathbb{P}\{X = t | X \geq t\} = \frac{p_t}{1 - \sum_{j=1}^{t-1} p_j},\ t = 0, 1, \ldots, \infty. \tag{88}$$

To proceed, observe the close the connection between $\lambda(t)$ and $1/c_t$. In particular, note that the quantity $1/c_{t+1}$ in Eq. 83 can be also written as

$$1/c_{t+1} = 1 + \lambda(t) = 1 + \frac{p_t}{1 - \sum_{j=1}^{t-1} p_j},\ t = 0, 1, \ldots, \infty. \tag{89}$$

Thus, we could obtain the bound (87) by using the hazard rate $\lambda(t)$ in Eq. 88 instead of $1/c_{t+1} - 1$.

An alternative way to obtain the bound (87) is to combine Eq. 85 with Eqs. 88 and 89. This results in

$$R_m \leq n + \sum_{t=0}^{n} \lambda(t) = n + \Lambda(t), \tag{90}$$

where $\Lambda(t)$ is the cumulative hazard function defined in Eq. 115. It also follows from the definitions of IFR and IFRA in Appendix that the Binomial random variable $Z = \sum_{i=1}^{n} X_i$ used in Lemma 10.1 is both an IFR and IFRA variable.

Lemma 10.2 *The variance of the random variable $\mathcal{N}_m$ associated with the $(N = 1)$-policy algorithm, while estimating ℓ in Eq. 79 with $p = 1/2$ is at most $C_{nm}(n+1)^2$, that is*

$$Var(\mathcal{N}_m) \leq C_{nm}(n+1)^2, \tag{91}$$

= *where C_{nm} is a constant depending on n and m and such that $C_{nm} \leq \frac{\pi^2}{6}$.*

Proof To calculate the variance $\mathcal{N}_m$ of the $(N = 1)$-policy algorithm observe that

1. The variance of a geometric random variable X with the parameter p is

$$\mathrm{Var}(X) = \frac{1-p}{p^2} \tag{92}$$

2. The number of trials to hit the level m_t from the level m_{t-1} is distributed Geom(c_t).
3. The trials in of the $(N = 1)$-policy algorithm are independent.

It follows from the above that the variance of the $(N = 1)$-policy algorithm is

$$\mathrm{Var}(\mathcal{N}_m) \leq \sum_{t=0}^{m} \frac{1-c_t}{c_t^2}. \tag{93}$$

Taking again into account Eq. 86, that is $c_t = \frac{n-t}{n+1}$ and $1 - c_t = \frac{t+1}{n+1}$ we obtain that

$$\mathrm{Var}(\mathcal{N}_m) \leq \sum_{t=0}^{m} \frac{(n+1)t}{(n+1-t)^2} = (n+1)^2 \sum_{n+1-m}^{n} \frac{1}{t^2} - (n+1) \sum_{n+1-m}^{n} \frac{1}{t}.$$

For large m the above sum can be again approximated by harmonic series. We finally obtain

$$\mathrm{Var}(\mathcal{N}_m) \leq (n+1)^2 C_{nm} - (n+1) \ln \frac{n}{n+1-m}, \tag{94}$$

where $C_{nm} = C_{n+1-m} \leq \pi^2/6$. In particular

$$C_1 = \pi^2/6,\ C_2 = (\pi^2/6) - 1,\ C_3 = (\pi^2/6) - 1 - (1/2)^2,$$
$$C_4 = (\pi^2/6) - 1 - (1/2)^2 - (1/3)^{1/2}, \ldots$$

Thus,

$$\mathrm{Var}(\mathcal{N}_m) \leq (\pi^2/6)(n+1)^2 = O(n^2). \tag{95}$$

□

Again, the bound Eq. 95 could be obtained by taking into account that $1/c_t = 1 + \lambda(t)$, where $\lambda(t)$ is the hazard rate and then proceeding with Eq. 93 in terms of $\lambda(t)$ rather than in terms of c_t. Note finally that for $n = m$ the expressions for R_m and $\mathrm{Var}(\mathcal{N}_m)$ coincide with the expressions for the classic *coupon collection problem* (Ross 2002a, b).

Using the 3σ rule we can conclude that with probability 0.99 the number of trials required to reach the level m is at most

$$R_m + 3(\mathrm{Var}\mathcal{N}_m)^{1/2} = n \ln \frac{n}{n+1-m} + 3\left(\frac{\pi}{\sqrt{6}}\right)(n+1).$$

As an example, for $n = 100$ and $m = 99$ we have that $R_m + 3(\mathrm{Var}\mathcal{N}_m)^{1/2} \approx 650$.

Finally, for large n and m one can apply the CLT (central limit theorem) for the random variable

$$\frac{\mathcal{N}_m - \mathbb{E}\mathcal{N}_m}{\sigma(\mathcal{N}_m)}.$$

(2) $p \neq 1/2$. Let us extend now the complexity analysis to the case where $p \neq 1/2$.

We consider here two cases **(a)** $p < 1/2$ and (b) $p > 1/2$. Note that $b = a + 1$. (a) $p < 1/2$. Assume that p is small. In particular, assume that $p = (1/n)^a$. In this case similar to Eqs. 86 and 87 analyses yield that as

$$1/c_t \leq n^a \frac{n+1}{n - m_{t-1}} \tag{96}$$

$$R_m \leq n^{a+1} \ln \frac{n}{n+1-m}. \tag{97}$$

If, for example, $p = 1/n$ we have that

$$R_m \leq n^2 \ln \frac{n}{n+1-m}. \tag{98}$$

Also, in analogy to Eq. 95 we obtain that

$$\mathrm{Var}(\mathcal{N}_m) \leq \sum_{t=0}^{m} \frac{1-c_t}{c_t^2} = O(n^{2(a+1)}). \tag{99}$$

This means, that in order to reach the level $m = n = 100$ with $p = 1/100$ using the $(N = 1)$-policy one needs on average at most $R_m = 46{,}000$ samples and in order to reach the level $m = n = 1{,}000$ with $p = 1/1{,}000$ one needs on average at most $R_m = 6.9 \cdot 10^6$ samples. Note that in the last case a huge sample is needed due to the fact that the probability p is small (rare event).

It is also not difficult to show that if $1/2 \leq p \leq 1/n$ then we obtain in analogy to Eqs. 98 and 99

$$R_m \leq n^b \ln \frac{n}{n+1-m}. \tag{100}$$

and

$$\mathrm{Var}(\mathcal{N}_m) = O(n^{2b}), \tag{101}$$

where1 $\leq b = b(p) \leq 2$ and again, $b(p) = 1$ for $p = 1/2$; $b(p) = 2$ for $p = 1/n$. Note that for $m = n$, Eqs. 100 and 101 reduce to

$$R_m \leq m^b \ln m. \tag{102}$$

and

$$\mathrm{Var}(\mathcal{N}_m) = O(m^{2b}), \tag{103}$$

respectively. **(b)** $p > 1/2$. Assume that p is large. In particular, assume that $p = (n - 1/n)$. In this case similar analyses yield

$$R_m \leq \ln \frac{n}{n+1-m}. \tag{104}$$

This means, that in order to reach the level $m = n = 100$ with $p = 99/100$ using the $(N = 1)$-policy one needs on average at most $R_m = 4.6$ samples and in order to reach the level $m = n = 1{,}000$ with $p = 999/1000$ one needs on average at most $R_m = 10$ samples.

Also in analogy to Eq. 94 the resulting variance is

$$\mathrm{Var}(\mathcal{N}_m) \leq C_{nm}. \tag{105}$$

That is the variance is bounded by the constant C_{nm}.

Consider now the complexity of the $(N = 1)$-policy algorithm, while estimating the following extended version of Eq. 79

$$\ell = \mathbb{E}_f\left[I_{\{\sum_{i=1}^n a_i X_i \geq m\}}\right], \tag{106}$$

where a_i are fixed coefficients and X_i are again iid $\mathrm{Ber}(p)$ random variables. Assume for concreteness that all a_i's are positive and $p = 1/2$. Denote $a_* = \min_{i=1,\ldots,n} a_i$ and assume that $a_* \geq 2/n$. Then it is readily seen that the complexity of the $(N = 1)$-policy algorithm coincides with Eqs. 100, 101, where the parameter b in Eqs. 100, 101 depends on a_*. In particular, we have $b = 1$, if all $a_i = 1$ and $b = 2$, if all $a_i = 2/n$. Note that if a_* is small, say $a_* < 2/n$, that is the corresponding $p_* = \mathbb{E}(a_* X_*) < 1/n$, then the $(N = 1)$-policy algorithm becomes much less efficient, since its complexity is defined by Eqs. 98 and 99.

Consider next the case of dependent Bernoulli random variables. It is difficult to define the complexity for general distribution $f(\boldsymbol{x}) = f(x_1, \ldots, x_n)$. We shall provide details in Section 10.1.2 while considering multiple events, while imposing some restrictions of the mode of dependency between the Bernoulli random variables.

Consider finally the complexity of the CLONCE version using the $(N = 1)$-policy. To see the benefit of CE consider for example the case $p = 1/2$. Let each parameter $p_k^*,\ k = 1, \ldots, n$ of the optimal CE parameter vector $\boldsymbol{p}^*$ be $> 1/2$. Assume for concreteness that while applying CE we obtain that each $p_k^* = \frac{(n-1)}{n}$. In this case we have again the results (104), (105), that is $R_m \leq \ln \frac{n}{n+1-m}$ and $\mathrm{Var}(\mathcal{N}_m) \leq C_{nm}$.

This is in contrast to $R_m \leq (n+1)\ln\frac{n}{n+1-m}$ and $\mathrm{Var}(\mathcal{N}_m) \leq C_{nm}(n+1)^2$, which was obtained for the $(N=1)$-policy without CE. Thus, by using CE the complexity can be substantially reduced.

10.1.2 Multiple Events

Consider now the average complexity $R_m = \mathbb{E}(\mathcal{N}_m)$ and the associated variance $\sigma_m^2 = \mathrm{Var}(\mathcal{N}_m)$ of the $(N=1)$-policy algorithm to hit the desired level m, while estimating (see Eq. 13)

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

for multiple events. Here $C_i,\ i=1,\ldots,n$ are given in Eqs. 14, 15 and the components $X_j,\ j=1,\ldots,n$ are iid Ber(1/2) random variables.

We shall prove that under some mild conditions the complexity (the average number of trials and the associated variance) of the $(N=1)$-policy algorithm, while estimating ℓ is given by Eqs. 102, 103, that is

$$R_m = \mathbb{E}(\mathcal{N}_m) \leq m^b \ln m, \text{ and } \mathrm{Var}(\mathcal{N}_m) = O(m^{2b}), \tag{107}$$

where the parameter $b = b(n)$ satisfies $1 \leq b \leq 2$.

To obtain R_m we shall introduce an auxiliary variable $Z = \sum_{i=1}^m X_i$, where the X_i's are iid Ber(p), with p defined below, and such that

$$R_m(Z) \geq R_m(Y).$$

Here $Y = \sum_{i=1}^m C_i$. Note that $R_m(Z)$ will be available from the results of Section 10.1.1, since $Z = \sum_{i=1}^m X_i$ and since the X_i's are *iid* Ber(p). Thus, instead of working directly with *dependent* random variables C_i's we shall work with iid X_i's ensuring that $R_m(Z) \geq R_m(Y)$ and therefore (see Eq. 102)

$$R_m \leq m^b \ln m.$$

The part $\mathrm{Var}(\mathcal{N}_m) = O(m^{2b})$ of Eq. 107 will be established similarly.

To this end note that in order for Eq. 102 to hold we need to impose some conditions on the parameters $p_i = \mathbb{E}(C_i),\ i=1,\ldots,m$ and on the dependency between the random variables $C_i,\ i=1,\ldots,m$. These conditions should be such that a in $1/c_t \leq m^a \frac{m+1}{m-m_{t-1}}$ (see Eq. 96 for $n=m$) satisfies $0 \leq a \leq 1$. (Recall that $b = a+1$ and that the case $a=0$ and $a=1$ correspond to iid Bernoullis with $p=1/2$ and $p=1/m$, respectively). Note that as soon as a in Eq. 96 satisfies $0 \leq a \leq 1$ we automatically obtain the part (103) of (107) as well, that is we obtain (by taking into account (102)) that $\mathrm{Var}(\mathcal{N}_m) = O(m^{2b})$. Thus, the proof of Eq. 107 reduces, in fact, to finding conditions on the random variables C_i, under which a in Eq. 96 satisfies $0 \leq a \leq 1$.

It is not difficult to see that in order for $a \in (0, 1)$ we need to impose the following two *basic* conditions on the indicator random variables C_i's:

(1) The probability $p_i = \mathbb{E}(C_i)$ of each indicator random variables C_i is not small. In particular it should be larger than $p = 1/m$ (see below).
(2) The dependency (correlation) between the random variables C_i should not be too strong (see below).

We next specify the basic conditions (1) and (2) by pointing out a couple of typical (large) sets of C_i's under which $0 \leq a \leq 1$ holds. Our conditions on the sets of C_i's assume that

(a) All probabilities $p_i = \mathbb{E}(C_i)$ are bounded from below by same p, say by $p = 1/2$, regardless of the value of m. The maximum number of non-zero terms a_{ik} ($a_{ik} \neq 0$) at each $C_i,\ i = 1, \ldots, m$, denoted by r , is small, say it is less than 10 regardless of m. To see that in this case a satisfies $0 \leq a \leq 1$, note first that for $r = 1$, which corresponds to iid Ber$(1/2)$, we have $b = 1$ and thus $a = 0$. Using now direct calculation for $1/c_t$ (with small parameter r and with $p_i = \mathbb{E}(C_i) = 1/2$) and taking into account that $m_t = m_{t-1} + 1$ we readily obtain that a in Eq. 96 satisfies $0 \leq a \leq 1$. This is also intuitively clear since while assuming that r is small we automatically impose weak correlation between the C_i's. Note that this is regardless of the actual location of each non-zero coefficients a_{ij} at their corresponding indicator C_i. Note also that it readily follows that $0 \leq a \leq 1$ will be still satisfied if instead of $p = 1/2$ we assume, say that $p \geq 1/10$.
(b) All probabilities $p_i = \mathbb{E}(C_i)$ are large relative to $1/m$, say all satisfy $p_i \geq (1/m)^{1/2}$. The number r of non-zero terms a_{ik} ($a_{ik} \neq 0$) at each $C_i,\ i = 1, \ldots, m$ is again small, say, as before, we assume that $r \leq 10$, regardless of m. To see that in this case a satisfies again $0 \leq a \leq 1$, note first that for $r = 1$, which corresponds to iid Ber$((1/m)^{1/2})$ we have in Eq. 107 that $b < 2$, and thus $a < 1$ (recall that the case $b = 2$ corresponds to $p = 1/m$). Using again direct calculation for $1/c_t$ with a small value r and taking into account that $m_t = m_{t-1} + 1$ we readily obtain that for $p_i \geq (1/m)^{1/2}$ the parameter a in Eq. 96 satisfies $0 \leq a \leq 1$.

Since the case (b) generalizes the case (a), we can summarize the above by the following

Theorem 10.1 *The complexity of the ($N = 1$)-policy algorithm to hit the level m, while estimating ℓ in Eq. 13 is defined by the expressions (102), (103), where the parameter b satisfies $1 \leq b \leq 2$, provided the following conditions hold:*

1. *All probabilities $p_i = \mathbb{E}(C_i)$ are large relative to $1/m$, say all satisfy $p_i \geq (1/m)^{1/2}$.*
2. *The number r of non-zero terms a_{ik} ($a_{ik} \neq 0$) at each $C_i,\ i = 1, \ldots, m$ is small, say, $r \leq 10$, regardless of m.*

Note that to prove Eq. 102 we could employ our general expression (90) for R_m in terms of the cumulative hazard function (CHF) $\Lambda(t)$ (see Appendix), that is

$$R_m \leq m + \sum_{t=0}^{m} \lambda(t) = m + \Lambda(t), \tag{108}$$

rather than in terms of c_t and similarly for $\mathrm{Var}\mathcal{N}_m$.

Remark 10.2 Since

$$\frac{1}{c_t} = \frac{|\mathcal{X}_{m_{t-1}}|}{|\mathcal{X}_{m_t}|} \tag{109}$$

the results of Theorem 10.1 are directly applicable for counting on the set $\mathcal{X}^*$ defined be the constraints of continuous and integer programs.

This in particular means that in order for the expressions (102), (103) to hold, while calculating a volume of a polytope defined on the set $\mathcal{X}^*$ of linear constraints, one should assume no outliers constraints in the sense that $p_{\min} = \min_{i=1,\ldots,m} \mathbb{E}(C_i)$ should be not too small relative to n. In other words, for Eqs. 102, 103 to hold the contribution of each indicator C_i should be similar while considering all possible combinations of volumes of polyhedrons containing m_{t-1} and $m_{t-1}+1$ constraints. Also, similar conditions should be imposed while counting the number of points defined on the set $\mathcal{X}^*$ of integer constraints.

All the above results on complexity of the $(N=1)$-policy algorithm to hit any desired level m_t can be extended for the cloning Algorithm 6.2, provided the sample size N and ρ are fixed. In the extended version instead of the $\mathrm{Geom}(c_t)$ distribution for the number of trials to hit the level m_t from the level m_{t-1} one needs to apply the order statistic theory. In particular, one needs to find the distribution of the number of trials (to hit the level m_t from the level m_{t-1}) associated with the order statistic $S_{\lceil 1-\rho \rceil}$, that is to find the distribution of the number of trials of the largest elite value of the ordered sample $S(\boldsymbol{X}_i),\ \ i=1,\ldots,N$. The above clearly presents a natural extension of the $\mathrm{Geom}(c_t)$ distribution for the number of trials of the $(N=1)$-policy algorithm. This subject is under current investigation.

Note finally, that as soon as formulas (86) for $1/c_t$ and the corresponding formulas (93) and (94) related to $\mathrm{Var}(\mathcal{N}_m)$ are available, we can compute the complexity of our estimator $\widehat{\ell}(m) = \prod_{t=0}^{T} \widehat{c}_t$ in Eq. 8 based on the product of $\widehat{c}_t$'s given in Eq. 9. We shall rather consider here the complexity of the direct estimator $|\widehat{\mathcal{X}^*_{\mathrm{dir}}}|$, considering the complexity of the standard estimator $|\widehat{\mathcal{X}^*}|$ in some further works.

10.2 The Complexity of the Direct Estimator

By the complexity of $|\widehat{\mathcal{X}^*_{\mathrm{dir}}}|$ we mean here the required sample size M of the cloning Algorithm 6.2 to identify all different points in $|\mathcal{X}^*|$ with high probability as soon as it hits the level m. As before, we assume that Gibbs sampler is perfect in the sense that it produces indeed a uniform distribution over $|\mathcal{X}^*|$. In our calculations we shall adopt the classic *coupon collection problem* (Ross 2002a, b) according to which there are ζ different types of coupons, and that it is equally likely that each time one obtains a coupon of any of these types. The problem is to calculate the average number of different coupons $\mathbb{E}(M)$ and the associated variance $\mathrm{Var}(M)$. Observe that instead of the coupon model one can consider an urn one with replacement containing ζ different balls. In this case one needs to calculate the average number of draws in order to have at least one of each type of the balls and the associated variance.

The solution of the problem is given in Ross (2002a, b). It is $\mathbb{E}(M) \sim \zeta \ln(\zeta)$ and $\mathrm{Var}(M) \sim \frac{\zeta^2\pi^2}{6}$.

Using the 3σ rule we can conclude that in order to allocate all different coupons with probability 0.99 the total number of samples N_{total} required is at most

$$N_{\text{total}} = \mathbb{E}(M) + 3\mathrm{Var}(M)^{1/2} = \zeta \ln(\zeta) + 3\left(\frac{\zeta\pi}{\sqrt{6}}\right) = \zeta\left(\ln(\zeta) + 3\frac{\pi}{\sqrt{6}}\right). \tag{110}$$

Thus, in order to sample all different points of the set $\mathcal{X}^*$ with probability 0.99 the total number of samples required is at most $\left(\ln(\zeta) + 3\frac{\pi}{\sqrt{6}}\right)$ times larger than ζ. This means that if, for example, $|\mathcal{X}^*| = \zeta = 100$ we need a total of at most $N_{\text{total}} = 900$ samples and if $|\mathcal{X}^*| = 1{,}000$ we need a total of at most $N_{\text{total}} = 12{,}000$ samples.

But, in our case the parameter $\zeta = |\mathcal{X}^*|$ (the number of coupons) is unknown. We shall show next that $|\mathcal{X}^*|$ can be easily estimated from simulation by taking into account the 3σ rule (110), while arguing as follows.

We assume that the range of $|\mathcal{X}^*|$ is $|\mathcal{X}^*| \in (0, K)$, where K is a *known* number. Let us limit K by a larger number, say $K = 1{,}000$. Then in order to estimate $|\mathcal{X}^*| \in (0, K)$ we can proceed by using the following "bisection" method.

1. For $K = 1{,}000$ start with $\zeta_0 = 500$. According to Eq. 110 this corresponds to $N_{\text{total},0} = 6,000$ trials. This means that while taking $N_{\text{total},0} = 6,000$ samples, we can estimate reliably the unknown number $|\mathcal{X}^*|$, provided $|\mathcal{X}^*| \le 500$. In short, if $|\mathcal{X}^*|$ is indeed any number ≤ 500, we can estimate it with $|\widehat{\mathcal{X}}^*_{\text{dir}}|$ using a sample $N_{\text{total},0} = 6,000$ and stop, otherwise we go to next step.
2. Employ the "bisection" method for $\zeta \in (500, 1{,}000)$, that is we set $\zeta_1 = 750$. In this case the corresponding sample (see Eq. 110) is at most $N_{\text{total},1} = 9,000$. If according to Algorithm 6.2 we obtain that $|\widehat{\mathcal{X}}^*_{\text{dir}}|$ is any number ≤ 750, deliver it as an estimator of $|\mathcal{X}^*|$ and stop, otherwise go to next bisection level, that is to the level $\zeta_2 = 875$, which corresponds to $N_{\text{total},2} = 10,500$ samples, etc. In short, in order to find a proper estimator $|\widehat{\mathcal{X}}^*_{\text{dir}}|$ of $|t\mathcal{X}^*|$ we need to find the first index k, for which given ζ_k and given the sample size $N = N_{\text{total},k}$ calculated according to Eq. 110, that is according

$$N_{\text{total},k} = \zeta_k\left(\ln(\zeta_k) + 3\frac{\pi}{\sqrt{6}}\right) \tag{111}$$

   our Algorithm 6.2 delivers $|\widehat{\mathcal{X}}^*_{\text{dir}}| \le \zeta_k$.

Remark 10.3 A related coupon collection problem is to find the expected value and variance of the number of distinct types, denoted as R, in a collection of N coupons. For our case this problem reads as: given a uniform sample of size N on $|\mathcal{X}^*|$, find the expected value and variance of R, presenting the number of distinct points on $|\mathcal{X}^*|$. In other words, taken a uniform sample of size N in $|\mathcal{X}^*|$, what would be the average number and the variance of distinct points one might expect? This problem can be considered as kind of dual to the former coupon collection problem.

The answer can be found again in Ross (2002a, b). It is

$$\mathbb{E}(R) = \zeta \left[1 - \left(\frac{\zeta - 1}{\zeta} \right)^N \right] \tag{112}$$

and

$$\mathrm{Var}(R) = \zeta \left[\left(\frac{\zeta - 1}{\zeta} \right)^N \right] + \zeta(\zeta - 1) \left[\left(\frac{\zeta - 2}{\zeta} \right)^N \right] - \zeta^2 \left[\left(\frac{\zeta - 1}{\zeta} \right)^{2N} \right]. \tag{113}$$

Based on Eqs. 112 and 113 we can use again the 3σ rule and estimate the most appropriate ζ and thus, the most appropriate estimator $|\widehat{\mathcal{X}}^*_{\text{dir}}|$ of $|\mathcal{X}^*|$, for a given budget (sample size) N.

11 Numerical Results

We present here numerical results with the proposed algorithms for counting and optimization.

A huge collection of instances (including real-world) is available on sites :

- http://people.brunel.ac.uk/~mastjjb/jeb/orlib/scpinfo.html
- http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/set-benchmarks.htm
- http://www.mat.univie.ac.at/~neum/glopt/test.html
- http://www.caam.rice.edu/~bixby/miplib/miplib.html
- For multiple-knapsack on http://hces.bus.olemiss.edu/tools.html, http://elib.zib.de/pub/Packages/mp-testdata/ip/sac94-suite/index.html, http://www.diku.dk/~pisinger/codes.html
- SAT problems is given on SATLIB website www.satlib.org

To study the variability in the solutions we run each problem 10 times and report the statistic. We used the following notations:

1. "Mean, max and min $|\widetilde{\mathcal{X}}^*|$" denote the sample mean, maximum and minimum and minimal values of the 10 estimates of $|\mathcal{X}^*|$.
2. "Mean, max and min $|\widehat{\mathcal{X}}^*_{\text{dir}}|$ " denote the sample mean, maximum and minimum values of the direct estimator (see Remark 6.6) found in each of the 10 samples of size N. Note that the maximum value of the "direct estimator" can be viewed as the lower bound of the true unknown quantity $|\mathcal{X}^*|$.
3. RE denotes the mean relative error for $|\widetilde{\mathcal{X}}^*|$, averaged over the 10 runs.
4. CPU denotes the mean relative error for $|\widetilde{\mathcal{X}}^*|$, averaged over the 10 runs.

11.1 Counting

The SAT Problem Table 5 presents the performance using the 2–0 pt method of Algorithm 6.2 for a random 3-SAT problem with an instance matrix $A = (20 \times 80)$ for $N = 1{,}000$ and $\rho = 0.05$. We found that the average relative error is RE = 0.08 and the average CPU time is 67 sec.

Table 5 Performance of Algorithm 6.2 for 3-SAT with the matrix $A = (20 \times 80)$

t	$\lvert\widetilde{\mathcal{X}}^*\rvert$			$\lvert\widehat{\mathcal{X}}^*_{\text{dir}}\rvert$			m_t
	Mean	Max	Min	Mean	Max	Min	
1	8,753.0	1.0e+004	7,077.9	0.6	2.0	0.0	75
2	1,879.8	2,126.3	1,501.9	1.9	4.0	0.0	77
3	219.2	250.7	195.1	7.2	12.0	4.0	78
4	15.1	16.9	11.5	14.9	15.0	14.0	79
5	15.1	16.9	11.5	15.0	15.0	15.0	80
6	15.1	16.9	11.5	15.0	15.0	15.0	80

The results are self- explanatory. Table 6 presents the dynamics for one of the runs of Algorithm 6.2 for the same model.

As before, we used the following notations

1. N_t and $N_t^{(s)}$ denotes the actual number of elites and the one after screening, respectively.
2. m_t^* and m_{*t} denotes the upper and the lower elite levels reached, respectively.
3. $\rho_t = N_t/N$ denotes the adaptive proposal rarity parameter.

In addition we found numerically that

- The naive ($N = 1$)-policy algorithm always reaches the target level $m = 80$. It, however, converges to a local extrema, delivering $|\mathcal{X}^*_{\text{dir}}| = 9$ instead of $|\mathcal{X}^*_{\text{dir}}| = 15$ obtained in Table 6 for $N = 1{,}000$. Note that the result $|\mathcal{X}^*_{\text{dir}}| = 9$ we obtained by slightly modifying the ($N = 1$)-policy algorithm as follows: until it reaches the level m we indeed used the ($N = 1$)-policy, but after reaching the level m we used a sample of size $N = 1{,}000$ (implementing the cloning and burn-in parameters according to Eq. 27) instead of $N = 1$.
- For $N \geq 150$ the cloning Algorithm 6.2 always delivers $|\mathcal{X}^*_{\text{dir}}| = 15$, while for $N < 150$ it converges to a local minima, delivering $|\mathcal{X}^*_{\text{dir}}| < 15$.

Tables 7 and 8 present data similar to Tables 5 and 6, respectively for the random 3-SAT problem with the instance matrix $A = (75 \times 325)$ taken from www.satlib.org. We set $N = 10{,}000$ and $\rho = 0.1$ for all iterations until Algorithm 6.2 reached the desired level 325. After that we switched to $N = 100{,}000$ for the last iteration. The results are self- explanatory.

We found that the average relative error is RE = 0.08 and the average CPU time is 10 minutes for each run. It is readily seen that at iteration 21 we obtained $\rho = 1$

Table 6 Dynamics of Algorithm 6.2

t	$\lvert\widetilde{\mathcal{X}}^*\rvert$	$\lvert\widehat{\mathcal{X}}^*_{\text{dir}}\rvert$	N_t	$N_t^{(s)}$	m_t^*	m_{*t}	ρ_t
1	8753	0.0	68	68	78	75	0.07
2	1879	1.0	146	146	80	77	0.11
3	219.0	7.0	292	266	80	78	0.22
4	14.0	14.0	118	91	80	79	0.13
5	14.0	15.0	78	15	80	80	0.08
6	14.0	15.0	1080	15	80	80	1.00

Table 7 Performance of Algorithm 6.2 for the random 3-SAT with the clause matrix $A = (75 \times 325)$, $N = 10{,}000$ and $\rho = 0.1$

t	$\lvert\widetilde{\mathcal{X}}^*\rvert$			$\lvert\widehat{\mathcal{X}}^*_{\text{dir}}\rvert$			m_t
	Mean	Max	Min	Mean	Max	Min	
1	5.4e+020	5.6e+020	5.1e+020	0.0	0.0	0.0	292
2	5.6e+019	6.0e+019	5.1e+019	0.0	0.0	0.0	297
3	6.5e+018	6.9e+018	6.0e+018	0.0	0.0	0.0	301
4	1.2e+018	1.3e+018	1.1e+018	0.0	0.0	0.0	304
5	1.7e+017	1.9e+017	1.6e+017	0.0	0.0	0.0	306
6	2.0e+016	2.2e+016	1.8e+016	0.0	0.0	0.0	308
7	6.1e+015	6.8e+015	5.7e+015	0.0	0.0	0.0	310
8	4.6e+014	5.2e+014	4.1e+014	0.0	0.0	0.0	312
9	2.5e+013	2.8e+013	2.2e+013	0.0	0.0	0.0	314
10	5.0e+012	5.7e+012	4.4e+012	0.0	0.0	0.0	315
11	9.5e+011	1.1e+012	8.2e+011	0.0	0.0	0.0	316
12	1.6e+011	1.8e+011	1.4e+011	0.0	0.0	0.0	317
13	2.5e+010	2.8e+010	2.1e+010	0.0	0.0	0.0	318
14	3.3e+009	3.9e+009	2.7e+009	0.0	0.0	0.0	319
15	3.9e+008	4.7e+008	3.2e+008	0.0	0.0	0.0	320
16	3.5e+008	4.7e+008	4.2e+007	0.0	0.0	0.0	321
17	3.8e+007	4.6e+007	3.1e+007	0.4	1.0	0.0	322
18	2.8e+006	3.4e+006	2.4e+006	8.9	12.0	5.0	323
19	1.4e+005	1.7e+005	1.1e+005	179.2	230.0	148.0	324
20	2,341.2	2,924.0	1,749.9	2,203.5	2,224.0	2,181.0	325
21	2,341.2	2,924.0	1,749.9	2,225.0	2,247.0	2,197.0	325

Table 8 Dynamics of Algorithm 6.2 for the random 3-SAT with the clause matrix $A = (75 \times 325)$

t	$\lvert\widetilde{\mathcal{X}}^*\rvert$	$\lvert\widehat{\mathcal{X}}^*_{\text{dir}}\rvert$	N_t	$N_t^{(s)}$	m_t^*	m_{*t}	ρ_t
1	5.4e+020	0.0	1,020	1,020	305	292	0.11
2	5.6e+019	0.0	1,714	1,714	307	297	0.14
3	6.5e+018	0.0	1,070	1,070	309	301	0.10
4	1.2e+018	0.0	1,462	1,462	310	304	0.12
5	1.7e+017	0.0	2,436	2,436	312	306	0.18
6	2.0e+016	0.0	2,166	2,166	314	308	0.14
7	6.1e+015	0.0	1,501	1,501	316	310	0.12
8	4.6e+014	0.0	1,115	1,115	316	312	0.09
9	2.5e+013	0.0	636	636	319	314	0.06
10	5.0e+012	0.0	2,213	2,213	320	315	0.23
11	9.5e+011	0.0	2,674	2,674	321	316	0.20
12	1.6e+011	0.0	1,969	1,969	320	317	0.19
13	2.5e+010	0.0	1,962	1,962	321	318	0.17
14	3.3e+009	0.0	1,775	1,775	322	319	0.16
15	3.9e+008	0.0	1,350	1,350	323	320	0.13
16	3.5e+008	0.0	1,437	1,437	324	321	0.12
17	3.8e+007	0.0	1,270	1,270	324	322	0.10
18	2.8e+006	8.0	924	924	325	323	0.08
19	1.4e+005	179.0	537	534	325	324	0.05
20	2,341.0	2,203.0	196	187	325	325	0.01
21	2,341.0	2,225.0	10,472	2,199	325	325	1.00

for $m_t = 324$; after that Algorithm 6.2 switches automatically from $\rho = 0.05$ to $\rho = 0.01$. This in turn results in switching from $m_t = 324$ to $m_t = m = 325$. Note again that without the adaptive ρ mechanism Algorithm 6.2 would terminate at $m_t = 324$ without reaching the final destination $m = 325$. In addition we found numerically that

- The naive $(N = 1)$-policy algorithm always reaches the target level $m = 325$. It, however, converges to a local extrema, delivering $|\mathcal{X}^*_{\text{dir}}| = 1{,}400$ instead of $|\mathcal{X}^*_{\text{dir}}| = 2,225$ obtained in Table 8 for $N = 10{,}000$.
- The cloning Algorithm 6.2 with the adaptive ρ as per Remark 6.5 performs similar to the one with the standard adaptive ρ as per Remark 6.2, which is, in fact, implemented in the cloning Algorithm 6.2 including also the above SAT models. In particular, we found some speed up, while using ρ as per Remark 6.5 versus ρ as per Remark 6.2. For example, using ρ as per Remark 6.5 we found for the SAT model with $A = (75 \times 325)$ that the Mean, Max, and Min values of the direct estimator $|\widehat{\mathcal{X}}^*_{\text{dir}}|$ are (2,215, 2,242, 2,193) instead of (2,225, 2,247, 2,197) as per Table 7, respectively. That is, although, the former version is a little faster than the latter, but at the same time it is a little less exact. The reason for that is that in the latter version we use a more conservative ρ, namely it is $0.25 \geq \rho \geq 0.01$, as compared to $\rho \approx 0.01$.

We also applied the CLONCE Algorithm 8.1 to a broad variety of SAT problems. We found that for small and moderate sizes, like the above problem with the matrix $A = (20 \times 80)$, CLONCE is faster and more accurate than the cloning Algorithm 6.2, while for large sizes, like the above problem with the matrix $A = (75 \times 325)$, CLONCE is slower and less accurate than the cloning Algorithm 6.2. As we mentioned, the reason for such inaccurate behavior of CLONCE is the degeneracy of the likelihood ratio term W in $\boldsymbol{p}^*$, especially for large n.

Counting 0–1 tables Table 9 presents dynamics of the cloning Algorithm 6.2 to count the number of 0–1 tables for the data set given in Table 4.3 from Lui (2001),

Table 9 Dynamics of Algorithm 6.2 for counting 0–1 tables the $A = (12 \times 17)$ binary matrix

t	$\|\mathcal{X}^*\|$	Empirical	$N_{t,e}$	$N^{(s)}_{t,e}$	m^*_t	m_{*t}	ρ_t
1	6.09e+060	6.00e+002	2,370	2,370	−68	−92	0.24
3	3.18e+059	1.13e+003	2,474	2,474	−62	−78	0.22
5	8.97e+057	1.20e+003	2,093	2,093	−56	−68	0.20
7	1.83e+056	1.04e+003	4,222	1,497	−48	−60	0.13
10	3.07e+053	1.90e+003	2,461	2,461	−42	−50	0.24
13	2.71e+051	1.80e+003	11,350	2,167	−36	−44	0.19
16	1.04e+049	1.30e+003	10,884	1,498	−32	−38	0.14
19	1.59e+046	1.05e+003	10,926	1,158	−24	−32	0.11
22	7.61e+042	6.70e+002	10,272	716	−22	−26	0.07
29	2.01e+032	1.48e+002	10,314	150	−10	−12	0.01
33	2.07e+023	2.40e+001	10,070	24	−4	−4	0.00
34	2.05e+020	1.00e+001	10,080	10	−2	−2	0.00
35	8.11e+016	4.00e+000	10,120	4	0	0	0.00
36	8.11e+016	4.00e+000	10,032	4	0	0	1.00
37	8.11e+016	4.00e+000	10,080	4	0	0	1.00

Table 10 Performance of Algorithm 6.2 for the permutation example

t	$\lvert\widetilde{\mathcal{X}}^*\rvert$			$\lvert\widehat{\mathcal{X}}^*_{\text{dir}}\rvert$			m_t
	Mean	Max	Min	Mean	Max	Min	
1	3.76E+05	3.83E+05	3.64E+05	8.6	15	4	339
2	40,698	43,345	36,778	104.1	119	91	362
3	4,881.6	5,444.8	3,898.9	647.1	749	574	373
4	2,873.8	3,146.2	2,674	2,504.9	2,621	2,415	375
5	2,873.8	3,146.2	2,674	2,903	2,903	2,903	375
6	2,873.8	3,146.2	2,674	2,903	2,903	2,903	375
7	2,873.8	3,146.2	2,674	2,903	2,903	2,903	375

presenting a $A = (12 \times 17)$ binary matrix. We set $\rho = 0.1$ and $N = 10{,}000$. Note that full enumeration results into 6.71×10^{16} different 0–1 tables. We found that the relative error based on 10 independent runs is about 20% and the CPU time for each run is about 5 minutes.

Counting Permutations Consider the problem of estimating the number $|\mathcal{X}^*|$ of permutations in $\mathcal{X}$, presenting the set of all permutations $\boldsymbol{x} = (x_1, \dots, x_n)$ of integers $1, \dots, n$, for which $\sum_{i=1}^n i x_i \geq m$, that is estimating the size $|\mathcal{X}^*|$ of the set (see also Botev and Kroese 2008)

$$\mathcal{X}^* = \left\{ \boldsymbol{x} \in \mathcal{X} : \sum_{i=1}^{n} i x_i \geq m \right\}.$$

As an example consider the case $n = 10$ and $m = 375$. For this case using full enumeration gives $|\mathcal{X}^*| = 2903$.

Table 10 presents the performance of using the 2–0pt method Algorithm 6.2 based on 10 independent runs with $N = 10{,}000$ and $\rho = 0.1$. As expected direct estimator $|\widehat{\mathcal{X}}^*_{\text{dir}}|$ is much more accurate than the standard one $|\widehat{\mathcal{X}}^*|$. In fact, we found that $|\widehat{\mathcal{X}}^*_{\text{dir}}|$ is exact, that is $|\widetilde{\mathcal{X}}^*_{\text{dir}}| = 2{,}903$, while the standard one is quite variable. Note that to obtain $|\widehat{\mathcal{X}}^*_{\text{dir}}| = 2{,}903$ we took a sample of $N = 50{,}000$ instead of $N = 10{,}000$ when Algorithm 6.2 reached the level $m = 375$. Table 11 presents the dynamics for one of the runs of Algorithm 6.2

Table 11 Dynamics of Algorithm 6.2 for the permutation example

t	$\lvert\widetilde{\mathcal{X}}^*\rvert$	$\lvert\widehat{\mathcal{X}}^*_{\text{dir}}\rvert$	N_t	$N_t^{(s)}$	m_t^*	m_{*t}	ρ_t
1	3.76E+05	8.6	1,034.8	1,033.6	380.3	339	0.10348
2	40,698	104.1	1,344.3	1,310.9	383	362	0.10836
3	4,881.6	647.1	1,283	1,096.5	384.3	373	0.11971
4	2,873.8	2,504.9	7,252	2,504.9	384.8	375	0.59455
5	2,873.8	2,903	54,008	2,903	385	375	1
6	2,873.8	2,903	58,060	2,903	385	375	1
7	2,873.8	2,903	58,060	2,903	385	375	1

11.2 Optimization

The Max-Cut problem Here we considered the max-cut problems taken from http://dimacs.rutgers.edu/Challenges/Seventh/Instances/lib.ps.

In particular we run the problems (1) TorusSetg-3-8 (*TSg38*) and (2) TorusSetpm-3-8-50 (*TSpm3850*) with both, the cloning Algorithm 6.2 and CLONCE Algorithm 8.2. Since the size of the problems are quite large we applied parallel computation with ten computers. We found that the relative error of both algorithms does not exceed 1%, but the CLONCE Algorithm is faster two to three times than the cloning algorithm. The reason is that in optimization CE is useful since we do not use likelihod ratios while updating $\boldsymbol{p}$.

TSP We run the cloning Algorithm 6.2 for the TSP models taken from http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/atsp/.

We run Algorithm 6.2 with $N = 10{,}000$ and $\rho = 0.1$ using systematic and random Gibbs sampler based on the $2 - opt$ heuristics. We found that the performance of the cloning Algorithm 6.2 in terms of the accuracy and the CPU time is similar to that of the CE algorithm. Below we present the solutions found by Algorithm 6.2 based on ten independent runs for the ftv33 model with the best known solution equal 1286 and $n = 34$ nodes using the 2–0pt method.

1,286, 1,329, 1,286, 1,302, 1,311, 1,323, 1,286, 1,311, 1,302, 1,311.

It takes approximately 40 iterations for each run. Using the alternative to the 2–0pt method we obtained 1,286 for most of the runs.

Knapsack Problem As a particular problem consider the Sento1.dat knapsack problem given in http://people.brunel.ac.uk/mastjjb/jeb/orlib/files/mknap2.txt.

The problem has 30 constraints and 60 variables. We ran the cloning Algorithm 5.1 and the CLONCE 8.2 for 10 independent runs with $\rho = 0.1$, $N = 1{,}000$, $b = 5$ and we found that both always found the best known solution. We also run it for different integer problems and the results were exact, that is they coincided with the best known solution.

Table 12 presents a typical dynamics of the CLONCE Algorithm 5.1 with $N = 1{,}000$ $\rho = 0.05$ for the problem $pb7$ taken from the website with the best known solution equal to 1035.

As one can see Algorithm 5.1 finds the best known solution after 8 iterations.

Table 12 A typical dynamics of the cloning Algorithm 5.1 with $N = 1{,}000$ $\rho = 0.05$ for the problem $pb7$

t	N_t	$N_t^{(s)}$	m_t^*	m_{*t}
1	51	51	130	769
2	51	51	693	859
3	51	51	808	920
4	51	51	884	955
5	51	41	939	986
6	65	32	972	1,014
7	58	21	1,006	1,035
8	52	1	1,035	1,035
9	1,000	1	1,035	1,035
10	1,000	1	1,035	1,035

12 Conclusion and Further Research

In this paper we presented several randomized algorithms for approximating the solutions of quite general NP-hard combinatorial optimization problems, counting, rare-event estimation and uniform sampling on complex regions. As usual for randomized algorithms, they use a sequential sampling plan such that the original "difficult" problem is decomposed into a sequence of "easy" ones. We showed that the main difference between the existing algorithms and the proposed ones is the latter have a special mechanisms, called the "cloning" device, which makes them very fast and accurate. In particular, they are well suited for solving problems associated with the Boltzmann distribution, like estimating the partition functions in an Ising model and for sampling random variables uniformly distributed on different convex bodies. We

1. Showed that all the cloning algorithms contain only *three parameters*: the rarity parameter ρ, the sample size N, and either the cloning parameter η, or the burn-in one b.
2. Presented a combined version of the cloning and cross-entropy (CE) algorithms, called the CLONCE algorithm and we discussed its efficiency.
3. Considered the complexity properties of a particular case of the cloning algorithm, the so-called $(N=1)$-policy algorithm, and proved its polynomial complexity in the sense of reaching a designed level m, while estimating the rare event probability $\ell(m)=\mathbb{E}_f\left[I_{\{S(\boldsymbol{X})\geq m\}}\right]$.
4. Discussed the complexity properties of the direct estimator obtained from Gibbs cloner.
5. Presented numerical results with both the cloning and the CLONCE algorithms and we showed that for optimization problems CLONCE typically outperforms the cloning algorithm, while for counting it is vise versa because of the degeneracy of the likelihood ratios. Note, however that for optimization problems CE is faster than both the cloning and the CLONCE algorithms. The main reason is that the calculation of the sample function in CE is less time consuming.

Further Research As for further research, we suggest considering the following issues:

1. Establish rigorous mathematical foundations for the general version of the cloning algorithm, providing rigorous proofs on the polynomial complexity and the speed of its convergence for rare-event probability estimation, counting and optimization. In particular, to extend the complexity results for the $(N=1)$-policy algorithm to the cloning Algorithm 6.2 in the sense that in the extended version instead of the Geom(c_t) distribution one needs to apply the order statistic theory for $S_{\lceil 1-\rho\rceil}$, presenting the largest elite value of the ordered sample $S(\boldsymbol{X}_i),\ i=1,\ldots,N$.
2. Apply the cloning algorithms to a broad variety of optimization and counting problems, like Hamiltonian cycles, self-avoiding walks, counting problems associated with graph coloring, cliques and counting the number of multiple extrema in a multi-extremal function.

3. Apply the cloning Algorithm 6.2 for rare event probability estimation in queueing models with heavy tails, like estimation of the probability of buffer overflow in the $GI/G/1$ queue.
4. Present rigorous statistical treatment on the performance of the cloning Algorithm 6.2 for generating samples uniformly distributed on different convex regions $\mathcal{X}^*$.

Acknowledgements I would like to thank Thomas Taimre from the University of Queensland, Brisbane, Australia for many valuable suggestions on the earlier draft, Andrey Dolgin and Alexander Libster for performing the computational part of the paper and to Ishay Wessman and Leonid Mytnick for several valuable remarks regarding the proof of complexity of the algorithm.

13 Appendix

13.1 Ross' Algorithm

Algorithm 13.1 (Ross' Algorithm)

```
1. Set J=N=0.
2. Choose a vector $\boldsymbol{x}$ such that $S(\boldsymbol{x}) \geq m_{t-1}$.
3. Generate a random vector $U \sim U(0,1)$ and set $I$=Int$(nU)+1$.
4. If $I = k$, generate $X$ given the conditional distribution of $X_k$, given that $X_j = x_j,\ j \neq k$.
5. If $S(x_1, \ldots, x_{k-1}, X, x_{k+1}, \ldots, x_n) < m_{t-1}$, return to 4.
6. $N = N + 1,\ x_k = X$.
7. If $S(\boldsymbol{x}) \geq m_t$, then $J = J + 1$.
8. Go to 3.
```

13.2 Hazard Rate

Definition: Hazard Rate The *hazard rate* or the *failure rate* of a random variable X is denoted by $\lambda(t)$ and is defined as

$$\lambda(t) = f(t)/(1 - F(t)) = f(t)/\bar{F}(t), \tag{114}$$

where $\bar{F}(t) = 1 - F(t)$ and $F(t)$ is the cumulative distribution function (cdf).

If we let

$$\Lambda(t) = \int_0^t \lambda(u)\,du \tag{115}$$

be the *cumulative hazard function*, we then have $F(t) = 1 - e^{-\Lambda(t)}$. Two other useful identities that follow from these formulas are:

$$\lambda(t) = -d \ln \bar{F}(t)/dt; \ \ \Lambda(t) = -\ln \bar{F}(t).$$

From $\Lambda(t) = -\ln \bar{F}(t)$ we see that $\Lambda(t)$ increases without bound as t tends to infinity (assuming $\bar{F}(t)$ tends to zero). This implies that $\lambda(t)$ must not decrease too quickly, since the cumulative hazard diverges. For example, e^{-t} is not the hazard function of any survival distribution, because its integral converges to 1.

Definition: Hazard Rate for a Discrete Random Variable Similar to Eq. 114 the hazard rate for an integer (countable) random variable X is defined as

$$\lambda(t) = \mathbb{P}\{X = t | X \geq t\} = \frac{f(t)}{R(t-1)} = \frac{p_t}{1 - \sum_{j=1}^{t-1} p_j}, \ t = 1, \ldots, \infty, \tag{116}$$

where $f(t) = \mathbb{P}\{X = t\}$. Note that $\lambda(0) = f(0)$. It is readily seen that

$$f(t) = \lambda(t) \prod_{j=0}^{t-1} (1 - \lambda(j)), \ t \geq 1.$$

Definition: IFR Distribution We say that the cdf (cumulative distribution function) $F(t)$ is an *increasing failure rate* (IFR) distribution if $\lambda(t)$ is an increasing function in t. Similarly, we say that $F(t)$ is an *decreasing failure rate* (DFR) distribution if $\lambda(t)$ is a decreasing function in t.

For example the Weibull distribution $f(x) = ab\ (bx)^{a-1}\, \mathrm{e}^{-(bx)^a}$ is IFR when $a \geq 1$ and DFR when $0 < a \leq 1$. When $a = 1$, we obtain the exponential pdf, which is both IFR and DFR.

Definition: IFRA distribution A distribution is said to have an *increasing failure rate on average* (IFRA) if

$$\frac{\Lambda(t)}{t} = \frac{\int_0^t \lambda(u)\, du}{t} \tag{117}$$

increases in t. Similarly, if $\frac{\Lambda(t)}{t}$ decreases in t we say that the distribution $F(t)$ has *decreasing failure rate on average* (DFRA).

It is well known that if $S(\boldsymbol{x})$ is a monotone function with independent components and each component x_k in $S(\boldsymbol{x})$ has an IFRA distribution, then the distribution of the random variable $S(\boldsymbol{X})$ itself is IFRA.

References

Bezakova I, Stefankovic D, Vazirani V, Vigoda E (2007) Accelerating simulated annealing for the permanent and combinatorial counting problems. SIAM J Comput (to appear)

Botev ZI (2007) Three examples of a practical exact markov chain sampling. Manuscript, November

Botev ZI, Kroese DP (2008) An efficient algorithm for rare-event probability estimation, combinatorial optimization, and counting. Methodol Comput Appl Probab (to appear)

Del Moral P (2004) Feynman–Kac formulae genealogical and interacting particle systems. Springer

Diaconis P, Andersen H (2007) Hit and run as a unifying device. Manuscript

Diaconis P, Holmes S (1994) Three examples of the Markov chain Monte Carlo method. In: Aldous D, et al (eds) Discrete probability and algorithms. Springer, New York, pp 43–56

Ghate AR, Smith RL (2008) A dynamic programing approach to efficient sampling from boltzmann distribution. OR Lett (to appear)

Lui JS (2001) Monte Carlo strategies in scientific computing. Springer

Mitzenmacher M, Upfal E (2005) Probability and computing: randomized algorithms and probabilistic analysis. Cambridge Univ. Press, New York (NY)

Motwani R, Raghavan R (1997) Randomized algorithms. Cambridge University Press

Ross SM (2002a) Simulation, 3rd edn. Academic

Ross SM (2002b) Probability models for computer science. Academic

Rubinstein RY (2008a) Semi-iterative minimum cross-entropy algorithms for rare-events, counting, combinatorial and integer programing. Methodol Comput Appl Probab (2):1–45

Rubinstein RY (2008b) Entropy and cloning methods for combinatorial optimization, sampling and counting using the gibbs sampler. Information theory and statistical learning. Springer

Rubinstein RY, Kroese DP (2004) The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning. Springer

Rubinstein RY, Kroese DP (2007) Simulation and the Monte Carlo method, 2nd edn. Wiley

Rubinstein RY, Kroese DP, Dolgin A, Glynn PW (2007) Parametric minimum cross-entropy method for counting the number of satisfiability assignments. Manuscript, Technion, Israel

Smith RL (1984) Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. Oper Res 32:1296–1308

Stefankovic D, Vempala S, Vigoda E (2007) Adaptive simulated annealing: a near-optimal connection between sampling and counting. FOCS