# On the Use of Smoothing to Improve the Performance of the Splitting Method

Frédéric Cérou[b], Arnaud Guyader[b,c], Reuven Rubinstein[a,1] and Radislav Vaisman[a]

[a] Faculty of Industrial Engineering and Management,
Technion, Israel Institute of Technology, Haifa, Israel
ierrr01@ie.technion.ac.il,slvaisman@gmail.com

[b] INRIA Rennes Bretagne Atlantique
Aspi project-team
Campus de Beaulieu, 35042 Rennes Cedex, France
Frederic.Cerou@irisa.fr

[c] Université Rennes II – Haute Bretagne
Campus Villejean
Place du Recteur Henri Le Moal, CS 24307
35043 Rennes Cedex, France
arnaud.guyader@uhb.fr

July 26, 2011

**Abstract**

We present an enhanced version of the splitting method, called the *smoothed splitting method* (SSM), for counting associated with complex

[1]Corresponding author (http://iew3.technion.ac.il/Home/Users/ierrr01.phtml).

[1†] The research of Reuven Rubinstein was supported by the BSF (Binational Science Foundation, grant No 2008482).

sets, such as the set defined by the constraints of an integer program and in particular for counting the number of satisfiability assignments. Like the conventional splitting algorithms, ours uses a sequential sampling plan to decompose a "difficult" problem into a sequence of "easy" ones. The main difference between SSM and splitting is that it works with an auxiliary sequence of continuous sets instead of the original discrete ones. The rationale of doing so is that continuous sets are easier to handle. We show that while the proposed method and its standard splitting counterpart are similar in their CPU time and variability, the former is more robust and more flexible than the latter.

**Keywords.** Combinatorial Optimization, Rare Event, Counting, Splitting.

**Mathematical Subject Classification.** Primary 65C05, 65C35; Secondary 68W20, 60C05.

# 1 Introduction: The Splitting Method

The goal of this work is to propose a novel and original way, called the *smoothed splitting method* (SSM), for counting on discrete sets associated with NP-hard discrete combinatorial problems and in particular counting the number of satisfiability assignments. The main idea of the SSM is to transform a combinatorial counting problem into a continuous integration problem using a type of "smoothing" of discrete indicator functions. Then we are in a position to apply a quite standard Sequential Monte Carlo/splitting method to this continuous integration problem. We show that although numerically the proposed method performs similar to the standard splitting one [15, 16] (in terms of CPU time and accuracy), the former one is more robust than the latter. In particular, tuning the parameters in SSM is simpler than in its standard splitting counterpart.

Before proceeding with SSM we present the splitting method for counting, following [15, 16]. For relevant references on the splitting method see [2], [4], [5], [7], [8], [9], [10], [11], which contain extensive valuable material as well as a detailed list of references. Recently, the connection between splitting for Markovian processes and *interacting particle methods* based on the Feynman-Kac model with a rigorous framework for mathematical analysis has been established in Del Moral's monograph [6].

The main idea of the splitting method for counting is to design a sequential sampling plan, with a view of decomposing a "difficult" counting problem defined on some set $\mathcal{X}^*$ into a number of "easy" ones associated with a sequence of related sets $\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_T$ and such that $\mathcal{X}_T = \mathcal{X}^*$. Similar to *randomized algorithms* [12], [13] splitting algorithms explore the connection between counting and sampling problems and in particular the reduction from approximate counting of a discrete set to approximate sampling of elements of this set, where the sampling is performed by the classic MCMC method [18]. Very recently, [1] discusses several splitting variants in a very similar setting, including a discussion on an empirical estimate of the variance of the rare event probability estimate.

A typical splitting algorithm comprises the following steps:

1. Formulate the counting problem as that of estimating the cardinality $|\mathcal{X}^*|$ of some set $\mathcal{X}^*$.

2. Find a sequence of sets $\mathcal{X} = \mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_T$ such that $\mathcal{X}_0 \supset \mathcal{X}_1 \supset \cdots \supset \mathcal{X}_T = \mathcal{X}^*$, and $|\mathcal{X}| = |\mathcal{X}_0|$ is known.

3. Write $|\mathcal{X}^*| = |\mathcal{X}_T|$ as

$$|\mathcal{X}^*| = |\mathcal{X}_0| \prod_{t=1}^{T} \frac{|\mathcal{X}_t|}{|\mathcal{X}_{t-1}|} = |\mathcal{X}_0| \ell, \tag{1}$$

where $\ell = \prod_{t=1}^{T} \frac{|\mathcal{X}_t|}{|\mathcal{X}_{t-1}|}$. Note that $\ell$ is typically very small, like $\ell = 10^{-100}$, while each ratio

$$c_t = \frac{|\mathcal{X}_t|}{|\mathcal{X}_{t-1}|} \tag{2}$$

should not be small, like $c_t = 10^{-2}$ or bigger. Clearly, estimating $\ell$ directly while sampling in $\mathcal{X}_0$ is meaningless, but estimating each $c_t$ separately seems to be a good alternative.

4. Develop an efficient estimator $\widehat{c}_t$ for each $c_t$ and estimate $|\mathcal{X}^*|$ by

$$\widehat{|\mathcal{X}^*|} = |\mathcal{X}_0|\,\widehat{\ell} = |\mathcal{X}_0| \prod_{t=1}^{T} \widehat{c}_t, \tag{3}$$

where $\widehat{\ell} = \prod_{t=1}^{T} \widehat{c}_t$ is an estimator of $\ell = \prod_{t=1}^{T} \frac{|\mathcal{X}_t|}{|\mathcal{X}_{t-1}|}$.

It is readily seen that in order to obtain a meaningful estimator of $|\mathcal{X}^*|$, we have to resolve the following two major problems:

(i) Put the well known NP-hard counting problems into the framework (1) by making sure that $\mathcal{X}_0 \supset \mathcal{X}_1 \supset \cdots \supset \mathcal{X}_T = \mathcal{X}^*$ and each $c_t$ is not a rare-event probability.

(ii) Obtain a low variance estimator $\widehat{c}_t$ of each $c_t = |\mathcal{X}_t|/|\mathcal{X}_{t-1}|$.

In Section 2, we briefly recall the SAT problem, which we will focus on in order to present our new method. In Section 3, which is our main one, we show how to resolve problems (i) and (ii) for the SAT problem by using the smoothed splitting method (SSM), which presents an enhanced version of the splitting method [15, 16]. Section 4 is devoted to the theoretical analysis of SSM in an idealized version, which we call i.i.d. SSM. In Section 5 numerical results

for both the SSM and splitting algorithm are presented. Their efficiencies are compared for several SAT instances.

## 2 Presentation of the SAT problem

The most common SAT problem comprises the following two components:

- A set of $n$ Boolean variables $\{x_1, \ldots, x_n\}$, 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, $\overline{\texttt{TRUE}} = \texttt{FALSE}$. A variable or its negation is called a *literal*.

- A set of $m$ distinct *clauses* $\{S_1, S_2, \ldots, S_m\}$ of the form $S_j = z_{j_1} \vee z_{j_2} \vee \cdots \vee z_{j_q}$, where the $z$'s are literals and the $\vee$ denotes the logical `OR` operator. For example, $0 \vee 1 = 1$.

The binary vector $\mathbf{x} = (x_1, \ldots, x_n)$ 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 $i = 1, \ldots, n$. The simplest SAT problem can now be formulated as: find a truth assignment $\mathbf{x}$ such that *all* clauses are true.

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

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

where the $S_j$'s 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.

**Toy Example** Let us consider the following toy SAT problem with two clauses and two variables: $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$. It is straightforward by considering all the four possible assignments, that this formula is satisfiable, with two valid assignments $x_1 = 1, x_2 = 0$ and $x_1 = 0, x_2 = 1$. If now we consider the three clauses formula $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2) \wedge (\bar{x}_2)$, then it is clearly unsatisfiable.

It is shown in [18] that the SAT-assignment problem can be modeled via rare-events with $\ell$ given by

$$\ell = \mathbb{E}\left[\mathbb{1}_{\left\{\sum_{j=1}^{m} C_j(\boldsymbol{X})=m\right\}}\right], \tag{4}$$

where $\boldsymbol{X}$ has a "uniform" distribution on the finite set $\{0,1\}^n$. It is important to note that here each $C_j(\mathbf{x}) = \mathbb{1}_{\{\sum_{k=1}^{n} a_{jk} x_k \geq b_j\}}$ can be also written alternatively as

$$C_j(\mathbf{x}) = \max_k \{0,\ (2\,x_k - 1)\,a_{jk}\}.$$

Here $C_j(\mathbf{x}) = 1$ if clause $S_j$ is `TRUE` with truth assignment $\mathbf{x}$ and $C_j(\mathbf{x}) = 0$ if it is `FALSE`, $A = (a_{jk})$ is a given clause matrix that indicates if the literal corresponds to the variable (+1) , its negation (-1), or that neither appears in the clause (0). If for example $x_k = 0$ and $a_{jk} = -1$, then the literal $\overline{x}_j$ is `TRUE`. The entire clause is `TRUE` if it contains at least one true literal. In other words, $\ell$ in (4) is the probability that a uniformly generated SAT assignment (trajectory) $\boldsymbol{X}$ is valid, that is, all clauses are satisfied, that is

$$S(x) = \min_{1 \leq j \leq m} C_j(x) \geq 1,$$

which is typically very small.

# 3 Smoothed Splitting Method

Before presenting the SSM algorithm we shall discuss its main features having in mind a SAT problem.

To proceed, recall that the main idea of SSM is to work within a continuous space rather than a discrete one. As a result this involves a continuous random vector $\mathbf{Y}$ instead of the discrete random vector $\boldsymbol{X}$ with i.i.d. components with law Ber($p = 1/2$). For example for a SAT problem one needs to adopt the following steps:

1. Choose a random vector $\mathbf{Y}$ of the same size as $\boldsymbol{X}$, such that the components $Y_1, \ldots, Y_n$, are i.i.d. uniformly distributed on the interval $(0, 1)$. Clearly the Bernoulli components $X_1, \ldots, X_n$ can be written as $X_1 = \mathbb{1}_{\{Y_1 > 1/2\}}, \ldots, X_n = \mathbb{1}_{\{Y_n > 1/2\}}$.

2. Instead of the former $0 - 1$ variables $x$ or $\bar{x}$ we will use for each clause a family of functions from $(0, 1)$ to $(0, 1)$. In particular, for each occurrence of $x$ or $\bar{x}$, we consider two functions, say $g_\varepsilon(y)$ and $h_\varepsilon(y) = g_\varepsilon(1 - y)$ indexed by $\varepsilon \geq 0$. These functions need to be increasing in $\varepsilon$, which means that

$$0 < \varepsilon \leq \varepsilon' \;\Rightarrow\; g_\varepsilon(y) \leq g_{\varepsilon'}(y),\ \forall y \in (0, 1). \tag{5}$$

   and for $\varepsilon = 0$, $g_0(y) = \mathbb{1}_{\{y > 1/2\}}$, $h_0(y) = g_0(1 - y) = \mathbb{1}_{\{y \leq 1/2\}}$. Possible choices of $g_\varepsilon(y)$ are:

$$g_\varepsilon(y) = (2y)^{1/\varepsilon} \mathbb{1}_{\{0 < y < \frac{1}{2}\}} + \mathbb{1}_{\{y > \frac{1}{2}\}} \tag{6}$$

or

$$g_\varepsilon(y) = \mathbb{1}_{\{\frac{1}{2}-\varepsilon<y<\frac{1}{2}\}}\left(\frac{y}{\varepsilon}+1-\frac{1}{2\varepsilon}\right)+\mathbb{1}_{\{y>\frac{1}{2}\}}. \tag{7}$$

or

$$g_\varepsilon(y) = \mathbb{1}_{[1/2-\varepsilon,1]}(y). \tag{8}$$

3. For each clause $C_j$, we consider the approximate $\varepsilon$-clause $C_{j\varepsilon}$, where we replace $x$ by $g_\varepsilon(y)$, $\bar{x}$ by $h_\varepsilon(y)$, and $\vee$ by $+$. Note also that the statement "$C_j$ is true" is replaced in the new notations by $C_{j\varepsilon} \geq 1$.

4. **Nested sets**. For each $\varepsilon \geq 0$, consider the subset (or event) $B_\varepsilon$ of $(0,1)^n$ defined as

$$B_\varepsilon = \{\mathbf{y} \in (0,1)^n : \forall j \in \{1,\ldots,m\}, C_{j\varepsilon}(\mathbf{y}) \geq 1\} = \{\mathbf{y} \in (0,1)^n : S_\varepsilon(\mathbf{y}) \geq 1\},$$

where $S_\varepsilon(\mathbf{y}) = \min_{1\leq j\leq m} C_{j\varepsilon}(\mathbf{y})$. Then it is clear from the above that for $\varepsilon_1 \geq \varepsilon_2 \geq 0$, we have the inclusions $B_0 \subset B_{\varepsilon_2} \subset B_{\varepsilon_1}$. Note that $B_0$ is the event for which *all* the *original* clauses are satisfied and $B_\varepsilon$ is an event on which *all* the *approximate* $\varepsilon$-clauses are satisfied. Note also that $\varepsilon_t,\ t = 1,\ldots,T$, should be a decreasing sequence, with $T$ being the number of nested sets, and $\varepsilon_T = 0$. In our SSM algorithm below (see section 3.2), we shall choose the sequence $\varepsilon_t,\ t = 1,\ldots,T$, adaptively, similar as the sequence $m_t,\ t = 1,\ldots,T$, is chosen in the Basic Splitting Algorithm of [16].

## 3.1 The SSM Algorithm with fixed nested subsets

Below we outline the main steps of the SSM algorithm.

1. **Initialization** Generate $N$ i.i.d. samples $\mathbf{Y}_1^1, \ldots, \mathbf{Y}_N^1$ of distribution $\mathcal{U}((0,1)^n)$.

2. **Selection** Keep only those samples for which all the $\varepsilon_1$-clauses (constructed with $g_{\varepsilon_1}$ and $h_{\varepsilon_1}$) are satisfied. Reindex them $1, \ldots, N_1$. Set $\widehat{p}_1 = N_1/N$.

3. **Cloning** Draw $N - N_1$ clones from the previous sample (with equal probabilities). Together with it we have again a sample of size $N$.

4. **Mutation** For all $N - N_1$ new samples apply the Gibbs sampler (see subsection 3.3 below) one or several times.

5. **Selection/Cloning/Mutation** for $\varepsilon_2, \ldots, \varepsilon_T$. This yields the estimates $\widehat{p}_2, \ldots, \widehat{p}_{T-1}$.

6. **Final Estimator and solutions to the original SAT problem** Select the samples that satisfy all the original clauses. Let $N_T$ be their number. Estimate $\widehat{p}_T = N_T/N$. From this last sample, construct a discrete sample $X_1, \ldots, X_{N_T}$ by $X_{j,k} = \mathbb{1}_{\{Y_{j,k} > 1/2\}}$, $1 \leq k \leq n$, which is not independent, but identically distributed on the instances of $\mathbf{x}$ that satisfy all the original clauses. An estimate of $\ell$ is given by $\widehat{\ell} = \prod_{t=1}^{T} \widehat{p}_t$, so that an estimate of $|\mathcal{X}^*|$ is given by $2^n \widehat{\ell} = 2^n \prod_{t=1}^{T} \widehat{p}_t$.

A crucial issue in this algorithm is to choose the successive levels $\varepsilon_1, \varepsilon_2$, etc., so that the variance of the estimator $\widehat{\ell}$ is as small as possible. The following subsection explains how to do it adaptively.

## 3.2 The SSM Algorithm with adaptive nested subsets

Say that we implemented the algorithm up to iteration $t$, and want to choose $\varepsilon_{t+1}$. Let $\mathbf{Y}_1^t, \ldots, \mathbf{Y}_N^t$ the current sample satisfying all the $\varepsilon_t$-clauses. Choose (as

usual in adaptive rare-event simulation) a given rate of success $\rho$, with $0 < \rho < 1$. Then the appropriate choice for $\varepsilon_{t+1}$ would be a value $\varepsilon > 0$ such that the number of replicas in the current sample $\mathbf{Y}_1^t, \ldots, \mathbf{Y}_N^t$ that satisfy all the $\varepsilon$-clauses is equal (close) to $\rho\, N$. A simple way of doing this is to perform a binary search in the interval $[0, \varepsilon_t]$ bearing in mind that $\varepsilon_t \geq \varepsilon_{t+1}$.

The following algorithm summarizes the above.

**Algorithm 3.1.** [Adaptive Choice of $\varepsilon_{t+1}$] Given the parameters $\rho$ and $\varepsilon_t$ proceed as follows:

1. Set $\varepsilon_{low} = 0$, $\varepsilon_{high} = \varepsilon_t$ and $\varepsilon_{t+1} = \frac{\varepsilon_{high}}{2}$.

2. While the proportion of replicas in the current sample $\mathbf{Y}_1^t, \ldots, \mathbf{Y}_N^t$ that satisfy all $\varepsilon_{t+1}$-clauses is not close to $\rho$, do the following:

   (a) Calculate the $\varepsilon_{t+1}$ performance $S_{\varepsilon_{t+1}}(\mathbf{Y})$ of the trajectories conveniently defined as the minimum over all $C_{\varepsilon_{t+1}}(\mathbf{Y})$ corresponding to the trajectory $\mathbf{Y}$. [Recall that by saying that $\mathbf{Y}$ is a satisfying trajectory, we mean that $S_{\varepsilon_{t+1}}(\mathbf{Y}) \geq 1$].

   (b) If the number of $\varepsilon_{t+1}$ satisfying trajectories is larger than $\rho\, N$ set $\varepsilon_{high} = \varepsilon_{t+1}$.

   (c) If the number of $\varepsilon_{t+1}$ satisfying trajectories is smaller than $\rho\, N$ set $\varepsilon_{low} = \varepsilon_{t+1}$.

   (d) Set $\varepsilon_{t+1} = \frac{\varepsilon_{low} + \varepsilon_{high}}{2}$.

3. Deliver $\varepsilon_{t+1}$ as the new adaptive level.

We are now in a position to describe the adaptive smoothed splitting algorithm, which is the one that will be used in the simulations.

**Algorithm 3.2.** [SSM Algorithm for Counting]

Fix the parameter $\rho$, say $\rho \in (0.01, 0.5)$ and the sample size $N$ such that $N_e = \rho N$ is an integer which denotes the size of the elite sample at each step. Choose also the function $g_\varepsilon(y)$, say the one given in (8), and $\varepsilon_0$ accordingly (e.g. $\varepsilon_0 = 1/2$ for (8)). Then execute the following steps:

1. **Acceptance-Rejection** Set a counter $t = 1$. Generate an i.i.d. sample $\mathbf{Y}_1^1, \ldots, \mathbf{Y}_N^1$ each uniformly on $(0,1)^n$. Obtain the first $\widehat{\varepsilon}_1$ using Algorithm 3.1 and let $\widehat{\mathcal{Y}}^1 = \{\widehat{\mathbf{Y}}_1^1, \ldots, \widehat{\mathbf{Y}}_{N_e}^1\}$ be the elite sample. Note that $\widehat{\mathbf{Y}}_1^1, \ldots, \widehat{\mathbf{Y}}_{N_e}^1 \sim \mathcal{U}(B_{\widehat{\varepsilon}_1})$, the uniform distribution on $B_{\widehat{\varepsilon}_1}$.

2. **Splitting (Cloning)** Given the elite sample $\{\widehat{\mathbf{Y}}_1^t, \ldots, \widehat{\mathbf{Y}}_{N_e}^t\}$ at iteration $t$, reproduce $\rho^{-1}$ times each vector $\widehat{\mathbf{Y}}_i^t$. Denote the entire new population by

$$\mathcal{Y}_{cl} = \{(\widehat{\mathbf{Y}}_1^t, \ldots, \widehat{\mathbf{Y}}_1^t), \ldots, (\widehat{\mathbf{Y}}_{N_e}^t, \ldots, \widehat{\mathbf{Y}}_{N_e}^t)\}.$$

   To each of the cloned vectors of the population $\mathcal{Y}_{cl}$ apply the MCMC (and in particular the Gibbs sampler Algorithm 3.3) for $b_t$ burn-in periods. Denote the *new entire* population by $\{\mathbf{Y}_1^{t+1}, \ldots, \mathbf{Y}_N^{t+1}\}$. Note that each vector in the sample $\mathbf{Y}_1^{t+1}, \ldots, \mathbf{Y}_N^{t+1}$ is distributed uniformly in $B_{\widehat{\varepsilon}_t}$.

3. **Adaptive choice** Obtain $\widehat{\varepsilon}_{t+1}$ using Algorithm 3.1. Note again that each vector of $\widehat{\mathbf{Y}}_1^{t+1}, \ldots, \widehat{\mathbf{Y}}_{N_e}^{t+1}$ of the elite sample is distributed uniformly in $B_{\widehat{\varepsilon}_{t+1}}$.

4. **Stopping rule** If $\widehat{\varepsilon}_{t+1} = 0$ go to step 5, otherwise set $t = t + 1$ and repeat from step 2.

5. **Final Estimator** Denote $\widehat{T} + 1$ the current counter, and

$$\widehat{r} = \frac{|\{i \in \{1, \ldots, N\} : S_0(\mathbf{Y}_i^{\widehat{T}+1}) \geq 1\}|}{N} > \rho,$$

and deliver $\widehat{\ell} = \widehat{r} \times \rho^{\widehat{T}}$ as an estimator of $\ell$ and $|\widehat{\mathcal{X}}^*| = 2^n \ \widehat{\ell}$ as an estimator of $|\mathcal{X}^*|$.

**Remark: Differences between Basic Splitting and SSM Algorithms**

1. SSM Algorithm 3.2 operates on a continuous space, namely $(0,1)^n$, while the Basic Splitting Algorithm of [16] operates on a discrete one, namely $\{0,1\}^n$. As a consequence their MCMC (Gibbs) samplers are different.

2. In the discrete case the performance function $S(\boldsymbol{X})$ represents the number of satisfied clauses, while in the continuous one it depends on both $\varepsilon$ and the $g_\varepsilon$. It is crucial to note that in the discrete case all clauses are satisfied *at the last iteration only* while in the continuous case *each clause is $\varepsilon_t$-satisfied at each iteration $t$.*

3. The stopping rules in both algorithms are the same. In particular, at the last iteration the SSM Algorithm 3.2 transforms its vectors from the continuous space to the discrete one.

## 3.3 Gibbs Sampler

Starting from $\mathbf{Y} = (Y_1, \ldots, Y_n)$, which is uniformly distributed on

$$B_\varepsilon = \{\mathbf{y} \in (0,1)^n : \ \forall j \in \{1, \ldots, m\}, C_{j\varepsilon}(\mathbf{y}) \geq 1\} = \{\mathbf{y} \in (0,1)^n : \ S_\varepsilon(\mathbf{y}) \geq 1\},$$

a possible way to generate $\widetilde{\mathbf{Y}}$ with the same law as $\mathbf{Y}$ is to use the following general systematic Gibbs sampler:

**Algorithm 3.3.** [Systematic Gibbs Sampler]

1. Draw $\widetilde{Y}_1$ from the conditional pdf $g(y_1|y_2, \ldots, y_n)$.

2. Draw $\widetilde{Y}_k$ from the conditional pdf $g(y_k|\widetilde{y}_1,\ldots,\widetilde{y}_{k-1},y_{k+1},\ldots,y_n)$, $2\le k\le n-1$.

3. Draw $\widetilde{Y}_n$ from the conditional pdf $g(y_n|\widetilde{y}_1,\ldots,\widetilde{y}_{n-1})$.

where $g$ is the target distribution. In our case, $g$ is the uniform distribution on $B_\varepsilon$, and the conditional distribution of the $k$th component given the others is simply the uniform distribution on some interval $(r,R)$ given as explained below.

**Toy Example** Let us consider first a small example with four variables and two clauses: $(X_1\vee X_2)\wedge(\bar{X}_2\vee X_3\vee\bar{X}_4)$. For a given $\varepsilon>0$, this gives the two $\varepsilon$-clauses:

$$\begin{aligned} g_\varepsilon(Y_1)+g_\varepsilon(Y_2) &\ge 1\\ h_\varepsilon(Y_2)+g_\varepsilon(Y_3)+h_\varepsilon(Y_4) &\ge 1. \end{aligned}$$

Let us say we want the distribution of $Y_2$ given $Y_1,Y_3,Y_4$. If we want the first one to be satisfied, we need $g_\varepsilon(Y_2)\ge 1-g_\varepsilon(Y_1)$, that is $Y_2\ge g_\varepsilon^{-1}(1-g_\varepsilon(Y_1))=r$. Similarly, the second clause gives $h_\varepsilon(Y_2)\ge 1-g_\varepsilon(Y_3)-h_\varepsilon(Y_4)$, and because $h_\varepsilon$ is decreasing, $Y_2\le h_\varepsilon^{-1}(1-g_\varepsilon(Y_3)-h_\varepsilon(Y_4))=1-g_\varepsilon^{-1}(1-g_\varepsilon(Y_3)-h_\varepsilon(Y_4))=R$. Thus the conditional distribution of $Y_2$ is uniform on the interval $(r,R)$.

The generalization is straightforward, and is given below.

**Algorithm 3.4.** [Conditional sampling of $\widetilde{Y}_k$]

- Denote by $\mathcal{I}_k$ the set of $\varepsilon$-clauses $C_{j\varepsilon}$ in which $g_\varepsilon(Y_k)$ is involved.
- For all $j\in\mathcal{I}_k$, denote by $Z_1,\ldots,Z_{q-1}$ the other $g_\varepsilon(Y_i)$'s or $h_\varepsilon(Y_i)$'s involved in clause $C_{j\varepsilon}$. Denote $r_j=g_\varepsilon^{-1}(1-Z_1-\cdots-Z_{q-1})$, and $r=\sup_{j\in\mathcal{I}_k}r_j$.
- Denote by $\mathcal{J}_k$ the set of $\varepsilon$-clauses $C_{j\varepsilon}$ in which $h_\varepsilon(Y_k)=g_\varepsilon(1-Y_k)$ is involved.

- For all $j \in \mathcal{J}_k$, denote by $Z_1, \ldots, Z_{q-1}$ the other $g_\varepsilon(Y_i)$'s or $h_\varepsilon(Y_i)$'s involved in clause $C_{j\varepsilon}$. Denote $R_j = 1 - g_\varepsilon^{-1}(1 - Z_1 - \cdots - Z_{q-1})$, and $R = \inf_{j \in \mathcal{I}_k} R_j$.
- Sample $\widetilde{Y}_k$ uniformly in the interval $[r, R]$.

**Remark:** It is readily seen that $r < R$ and $\widetilde{\mathbf{Y}} = (\widetilde{Y}_1, \ldots, \widetilde{Y}_n)$ has the same distribution as $\mathbf{Y}$. This is so since the initial point $\mathbf{Y} = (Y_1, \ldots, Y_n)$ belongs to and is uniformly distributed in $B_\varepsilon$. Note that our simulation results clearly indicate that one round of the Gibbs Algorithm 3.3 suffices for good experimental results. Nonetheless, if one wants the new vector $\widetilde{\mathbf{Y}}$ to be independent of its initial position $\mathbf{Y}$, then in theory the Gibbs sampler would have to be applied an infinite number of times. This is what we call the i.i.d. SSM in section 4, and this is the algorithm that we will analyze from a theoretical point of view.

# 4 Statistical Analysis of i.i.d. SSM

It is possible to obtain exact results about the estimator $\widehat{\ell}$ in an assumed situation (never encountered in practice) that each step begins with an $N$ i.i.d. sample. We call this idealized version "the *i.i.d.* smoothed splitting algorithm" - *i.i.d. SSM*. This would typically correspond to the situation where at each step the Gibbs sampler is applied an infinite number of times, which is not realistic but will be our main hypothesis in Subsection 4.1. The following theoretical results do not exactly match the algorithm which is used in practice, but can be expected to provide insight.

## 4.1 Statistical Analysis of i.i.d. SSM

The aim of this subsection is to precise the statistical properties of the estimator $\widehat{\ell}$ obtained by the *i.i.d. SSM*.

Let us denote by $s$ the number of solutions of the SAT problem at hand, and by $\mathcal{S}$ the union of $s$ hypercubes (with edge length $1/2$) which correspond to these solutions in the continuous version: this means that for all $\mathbf{y} = (y_1, \ldots, y_n) \in (0,1)^n$, $\mathbf{y}$ belongs to $\mathcal{S}$ if and only if $\mathbf{x} = (\mathbb{1}_{y_1 \geq 1/2}, \ldots, \mathbb{1}_{y_n \geq 1/2})$ is a solution of the SAT problem.

With these notations, the probability that we are trying to estimate is

$$\ell = \mathbb{P}(\mathbf{Y} \in \mathcal{S})$$

where $\mathbf{Y}$ is a uniform random vector in the hypercube $(0,1)^n$. Recall that for any $\varepsilon \geq 0$

$$B_\varepsilon = \{\mathbf{y} \in (0,1)^n : \ \forall j \in \{1, \ldots, m\}, C_{j\varepsilon}(\mathbf{y}) \geq 1\} = \{\mathbf{y} \in (0,1)^n : \ S_j(\mathbf{y}) \geq \varepsilon\},$$

so that we have the following Bayes formula for the splitting algorithm

$$\ell = \mathbb{P}(B_0) = \mathbb{P}(B_0 | B_{\varepsilon_T}) \times \cdots \times \mathbb{P}(B_{\varepsilon_1} | B_{\varepsilon_0}),$$

where $\varepsilon_0$ is large enough (possibly infinite) so that $\mathbb{P}(B_{\varepsilon_0}) = 1$ (for example $\varepsilon_0 = 1/2$ when $g_\varepsilon$ is defined by formula (8) and $\varepsilon_0 = +\infty$ when $g_\varepsilon$ is defined by formula (6) or (7)).

Let us now describe briefly the smoothed splitting algorithm in this framework. As previously, $\rho$ is the fixed proportion of the elite sample at each step. For simplicity, we will assume that $\rho N$ is an integer.

Starting with an $N$ i.i.d. sample $(\mathbf{Y}_1^1, \ldots, \mathbf{Y}_N^1)$, with $\mathbf{Y}_i^1$ uniformly distributed in $(0,1)^n$ for all $i \in \{1, \ldots, N\}$, the first step consists in applying a binary search to find $\widehat{\varepsilon}_1$ such that

$$\frac{|\{i \in \{1, \ldots, N\} : \mathbf{Y}_i^1 \in B_{\widehat{\varepsilon}_1}\}|}{N} = \rho.$$

Such an $\widehat{\varepsilon}_1$ is not unique, but this will not matter from the theoretical point of view, as will become clear in the proof of Theorem 4.1 below.

Knowing $\widehat{\varepsilon}_1$ and using a Gibbs sampler, the elite sample of size $\rho N$ allows ideally (which means: for the i.i.d. SSM) to draw an $N$ i.i.d. sample $(\mathbf{Y}_1^2, \ldots, \mathbf{Y}_N^2)$, with $\mathbf{Y}_i^2$ uniformly distributed in $B_{\widehat{\varepsilon}_1}$. Using a binary search , one can then find $\widehat{\varepsilon}_2$ such that

$$\frac{|\{i \in \{1, \ldots, N\} : \mathbf{Y}_i^2 \in B_{\widehat{\varepsilon}_2}\}|}{N} = \rho,$$

and iterate the algorithm, with only the last step being different: the algorithm stops when for an $N$ i.i.d. sample $(\mathbf{Y}_1^{\widehat{T}+1}, \ldots, \mathbf{Y}_N^{\widehat{T}+1})$, with $\mathbf{Y}_i^{\widehat{T}+1}$ uniformly distributed in $B_{\widehat{\varepsilon}_{\widehat{T}}}$, the proportion of points which satisfy the SAT problem is larger than $\rho$:

$$\frac{|\{i \in \{1, \ldots, N\} : \mathbf{Y}_i^{\widehat{T}+1} \in B_0\}|}{N} = \widehat{r} > \rho.$$

In summary, the "ideal" smoothed splitting estimator is defined as

$$\widehat{\ell} = \widehat{r}\ \rho^{\widehat{T}}, \text{with } \widehat{r} \in (\rho, 1],$$

whereas the true probability of the rare event may be decomposed as

$$\ell = r\ \rho^T, \text{with } T = \left\lfloor \frac{\log \ell}{\log \rho} \right\rfloor \text{ and } r = \ell \rho^{-T} \in (\rho, 1].$$

Let us summarize now the statistical properties of this "ideal" estimator.

**Theorem 4.1.** *The ideal estimator $\widehat{\ell}$ has the following properties:*

1. *Strong consistency:* $\widehat{\ell} \xrightarrow[N\to\infty]{a.s.} \ell$
2. *Number of steps:* $\mathbb{P}(\widehat{T} \neq T) \leq 2\ (T+1)\ e^{-2N\alpha^2}$ *where* $\alpha = \min(\rho - \ell^{\frac{1}{T}}, \ell^{\frac{1}{T+1}} - \rho)$.
3. *Asymptotic normality:* $\sqrt{N}\ \frac{\widehat{\ell}-\ell}{\ell} \xrightarrow[N\to\infty]{\mathcal{D}} \mathcal{N}(0,\sigma^2)$ *where* $\sigma^2 = T\ \frac{1-\rho}{\rho} + \frac{1-r}{r}$.
4. *Positive bias:* $N\ \frac{\mathbb{E}[\ \widehat{\ell}\ ]-\ell}{\ell} \xrightarrow[N\to\infty]{} T\ \frac{1-\rho}{\rho}$.

**Proof.** We first prove the strong consistency. Let us denote by $F(\varepsilon)$ the Lebesgue measure of $B_\varepsilon$: $\forall \varepsilon \in \mathbb{R},\ F(\varepsilon) = \mathbb{P}(\mathbf{Y} \in B_\varepsilon)$. By convention, we will assume that $B_\varepsilon = \emptyset$ for $\varepsilon < 0$. One can readily see that $F(\varepsilon)$ has the following properties:

- $F(\varepsilon) = 0$ when $\varepsilon < 0$,
- $F(0) = \ell$,
- $F(\varepsilon) = 1$ when $\varepsilon \geq \varepsilon_0$, or $\lim_{\varepsilon\to+\infty} F(\varepsilon) = 1$ in the infinite case (cf. for example formulae (6) or (7)),
- $F$ is a non decreasing and continuous function on $(0, \varepsilon_0)$.

We will also make use of the mapping $F(\varepsilon, \varepsilon')$, defined for $0 \leq \varepsilon' \leq \varepsilon \leq \varepsilon_0$ as

$$F(\varepsilon, \varepsilon') = \mathbb{P}(\mathbf{Y} \in B_{\varepsilon'} | \mathbf{Y} \in B_\varepsilon) = \frac{F(\varepsilon')}{F(\varepsilon)}.$$

With these notations, let us recall the following point: by construction and by assumption on the i.i.d. SSM, given $\widehat{\varepsilon}_{t-1}$, the random vectors $\mathbf{Y}_1^t, \ldots, \mathbf{Y}_N^t$ are

i.i.d. with uniform distribution in $B_{\widehat{\varepsilon}_{t-1}}$. For all $i = 1, \ldots, N$, let us define

$$\varepsilon(\mathbf{Y}_i^t) = \inf\{\varepsilon \in [0, \widehat{\varepsilon}_{t-1}] : S_\varepsilon(\mathbf{Y}_i^t) \geq 1\}.$$

Then the random variables $D_1 = \varepsilon(\mathbf{Y}_1^t), \ldots, D_N = \varepsilon(\mathbf{Y}_N^t)$ are i.i.d. with cdf $F(\widehat{\varepsilon}_{t-1}, .)$.

Thus, given $\widehat{\varepsilon}_{t-1}$, $\widehat{\varepsilon}_t$ is an empirical quantile of order $\rho$ for the i.i.d. sample $(D_1, \ldots, D_N)$. Denoting by $F_N(\widehat{\varepsilon}_{t-1}, .)$ the empirical cdf of $F$ with this sample, we have

$$|F(\widehat{\varepsilon}_{t-1}, \widehat{\varepsilon}_t) - \rho| \leq |F(\widehat{\varepsilon}_{t-1}, \widehat{\varepsilon}_t) - F_N(\widehat{\varepsilon}_{t-1}, \widehat{\varepsilon}_t)| + |F_N(\widehat{\varepsilon}_{t-1}, \widehat{\varepsilon}_t) - \rho|.$$

By construction of $\widehat{\varepsilon}_t$, we know that the second term of this inequality is less than $1/N$, so that the almost sure convergence to 0 follows for it. For the first term, denoting by $\|f\|_\infty$ the supremum norm of $f$, and using the Dvoretsky-Kiefer-Wolfowitz inequality (see for example [19] p. 268), we know that for any $\eta > 0$

$$\mathbb{P}(\|F(\widehat{\varepsilon}_{t-1}, .) - F_N(\widehat{\varepsilon}_{t-1}, .)\|_\infty \geq \eta) \leq 2e^{-2N\eta^2},$$

which guarantees the almost sure convergence via the Borel-Cantelli Lemma. Thus we have proved that for all $t$

$$F(\widehat{\varepsilon}_{t-1}, \widehat{\varepsilon}_t) \xrightarrow[N\to\infty]{a.s.} \rho$$

Next, since the product of a finite and deterministic number of random variables will almost surely converge to the product of the limits, we conclude that for all

$t$

$$\rho^t - \prod_{k=1}^{t} F(\widehat{\varepsilon}_{k-1}, \widehat{\varepsilon}_k) \xrightarrow[N\to\infty]{a.s.} 0.$$

Finally we have to proceed with the last step. We will only focus on the general case where $\log \ell / \log \rho$ is not an integer. Recall that $T = \lfloor \log l / \log \rho \rfloor$ is the "correct" (theoretical) number of steps *i.e.* the number of steps that "should" be done, whereas $\widehat{T}$ is the true and random number of steps of the algorithm. From the preceding results, we have that almost surely for $N$ large enough

$$\prod_{k=1}^{T+1} F(\widehat{\varepsilon}_{k-1}, \widehat{\varepsilon}_k) < \ell < \prod_{k=1}^{T} F(\widehat{\varepsilon}_{k-1}, \widehat{\varepsilon}_k),$$

so that, almost surely for $N$ large enough, the algorithm stops after $\widehat{T} = T$ steps. Therefore, in the following, we can assume that $\widehat{T} = T$.

Using the same reasoning as previously, we have

$$|F(\widehat{\varepsilon}_T, 0) - F_N(\widehat{\varepsilon}_T, 0)| \xrightarrow[N\to\infty]{a.s.} 0.$$

By definition, $T$ satisfies

$$\prod_{k=1}^{T} F(\widehat{\varepsilon}_{k-1}, \widehat{\varepsilon}_k)\ F(\widehat{\varepsilon}_T, 0) = F(0) = \ell,$$

which implies

$$F(\widehat{\varepsilon}_T, 0) \xrightarrow[N\to\infty]{a.s.} \frac{\ell}{\rho^T},$$

and also

$$F_N(\widehat{\varepsilon}_T, 0) \xrightarrow[N\to\infty]{a.s.} \frac{\ell}{\rho^T}.$$

Putting all things together, we get

$$\widehat{\ell} = F_N(\widehat{\varepsilon}_T, 0) \times \rho^T \xrightarrow[N\to\infty]{a.s.} \frac{\ell}{\rho^T} \times \rho^T = \ell,$$

which concludes the proof of the consistency.

Let us prove now the exponential upper bound for the probability that $\widehat{T}$ differs from $T$. To this end, let us denote by $A = \{\widehat{T} = T\}$ the event for which the algorithm stops after the correct number of steps, and which can be written as follows

$$A = \{\widehat{\varepsilon}_{T+1} = 0 < \widehat{\varepsilon}_T\} = \left\{ \prod_{k=1}^{T+1} F(\widehat{\varepsilon}_{k-1}, \widehat{\varepsilon}_k) = \widehat{\ell} < \prod_{k=1}^{T} F(\widehat{\varepsilon}_{k-1}, \widehat{\varepsilon}_k) \right\}.$$

For all $k = 1, \ldots, T+1$, if we denote

$$A_k = \left\{ \ell^{\frac{1}{T}} - \rho < \rho - F(\widehat{\varepsilon}_{k-1}, \widehat{\varepsilon}_k) < \ell^{\frac{1}{T+1}} - \rho \right\},$$

we have

$$\mathbb{P}(A) \geq \mathbb{P}(A_1 \cap \cdots \cap A_{T+1}) \geq 1 - \sum_{k=1}^{T+1} (1 - \mathbb{P}(A_k)).$$

Denoting $\alpha = \min\left(\rho - \ell^{\frac{1}{T}}, \ell^{\frac{1}{T+1}} - \rho\right)$, the Dvoretsky-Kiefer-Wolfowitz inequality implies

$$1 - \mathbb{P}(A_k) \leq \mathbb{P}(|\rho - F(\widehat{\varepsilon}_{k-1}, \widehat{\varepsilon}_k)| > \alpha) \leq 2e^{-2N\alpha^2},$$

so that the result is proved

$$\mathbb{P}(A) = \mathbb{P}(\widehat{T} = T) \geq 1 - 2(T+1)e^{-2N\alpha^2}.$$

By the way, this is another method to see that $\widehat{T} \xrightarrow[N\to\infty]{a.s.} T$.

For the asymptotic normality and bias properties, we refer the reader to Theorem 1 and Proposition 4 of [3]: using the notations and tools of smoothed splitting, the proofs there can be adapted to yield the desired results.

## 4.2 Remarks and comments

**Number of steps** With an exponential probability, the number of steps of the algorithm is $T = \lfloor \log \ell / \log \rho \rfloor$.

**Bias** The fact that this estimator is biased stems from the adaptive character of the algorithm. This is not the case with a sequence of fixed levels $(\varepsilon_1, \ldots, \varepsilon_T)$. However, this bias is of order $1/N$, so that when $N$ is large enough, it is clearly negligible relative to the standard deviation. Moreover, the explicit formula for this bias allows us to derive confidence intervals for $\ell$ which take this bias into account.

**Estimate of the rare-event cardinality** The previous discussion focused on the estimation of the rare-event probability, which in turn provides an estimate of the actual number of solutions to the original SAT problem by taking $|\widehat{\mathcal{X}}^*| = 2^n \, \widehat{\ell}$. In fact, the number of solutions may be small and thus can be determined by actual counting the different instances in the last sample of the algorithm.This estimator will be denoted by $|\widehat{\mathcal{X}}^*_{dir}|$. Typically it underestimates the true number of solutions $|\mathcal{X}^*|$, but at the same time it has a smaller (empirical) variance as compared to the product estimator. Even if we do not know its mathematical properties, this estimate can be useful. Firstly, it may be interesting for practical purposes to know the set (and the number) of all the different solutions that have been found for the original SAT problem. Secondly, it is also convenient when

we compare our results with the ones given by the algorithm in [16], where a screening step (i.e. removal of the duplicates on the finite space) is involved.

**Mixing properties** Our purpose here is to explain why the Gibbs sampler used at each step of the algorithm is irreducible and globally reaching and hence has good mixing properties. For the sake of clarity, we will focus first on $g_\varepsilon$ as per (8). With this function, for a given $\varepsilon$, we can split the region explored by the Gibbs sampler in several small (sub) hypercubes or hyperrectangles, as shown schematically in Figure 1. To each vertex of the whole hypercube $(0,1)^n$ that represents a solution of the original SAT problem, corresponds a sub-hypercube of edge length $1/2+\varepsilon$, including the central point with coordinates $(1/2,\ldots,1/2)$. And around this point, we have a sub-hypercube of edge length $2\varepsilon$, which is common to all those elements.

For the other parts of the domain, which do not correspond to a solution, things become a bit more complicated. It is a union of $\varepsilon$-thin "fingers" extending outwards in several directions (a subspace). The corresponding sub-domain being explored depends on the minimum number of variables that need to be taken in $(1/2-\varepsilon, 1/2+\varepsilon)$ in order to satisfy all the $\varepsilon$-clauses. The domain is then a rectangle of length $1/2+\varepsilon$ on the "free" variables, and of length $2\varepsilon$ in the other directions, that is on the $(1/2-\varepsilon, 1/2+\varepsilon)$ constrained variables. Again, all those rectangles include the small central sub-hypercube.

The union of all these sub-hypercubes/rectangles is the domain currently explored by the Gibbs sampler. The geometry of the whole domain is then quite complex.

It is clear that starting with any one of these sub-hypercubes/rectangles we can reach any other point within it in one iteration of the Gibbs sampler. More-

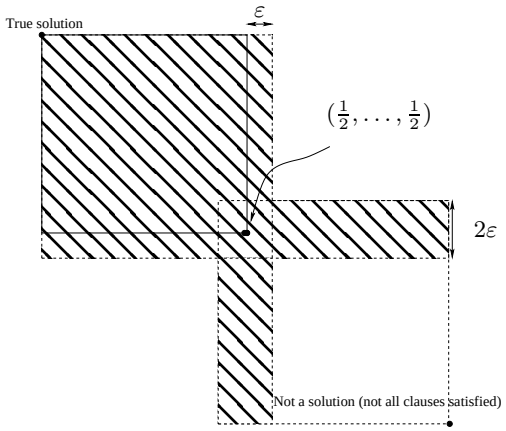

Figure 1: Partial mixing of the Gibbs sampler.

over, as long as the Markov chain stays within the same sub-hypercube/rectangle, any other point is accessed with uniform probability. This means that the mixing properties of our Gibbs sampler are the best possible as long as *we are restricted to one sub-hypercube*. Actually this suffices to make the algorithm work.

For $g_\varepsilon$ as per (6) or (7), the same picture mostly holds, but the mixing properties within each sub-hypercube is not that easy to analyze. This is somehow compensated by an ability to deal with the inter-variable relations: the geometry of the domain explored around the centre point reflects these constraints, and thus has a much more complicated shape. These $g_\varepsilon$ functions work in practice better than (8).

# 5 Numerical Results

Below we present numerical results with both SSM Algorithm 3.2 and its counterpart Enhanced Cloner Algorithm [16] for several SAT instances. In particular we present data for three different SAT models: one of small size, another of moderate size and the third of large size. To study the variability in the solutions we run each problem 10 times and report the statistic.

To compare the efficiencies of both algorithms we run them on the same set of parameters $\rho$ and $b$, where $b$ is the number of cycles in the systematic Gibbs sampler. If not stated otherwise we set $b = 1$ and $\rho = 0.2$. From our numerical results follows that although both algorithms perform similarly (in terms of the CPU time and variability) the SSM is more robust than its splitting counterpart. In particular we shall see that SSM Algorithm 3.2 produces quite reliable estimator for a large set of $b$ including $b = 1$, while its splitting counterpart Enhanced Cloner Algorithm is quite sensitive to $b$ and thus, requires tuning.

Below we use the following notations:

1. $N_t^{(e)}$ and $N_t^{(s)}$ denote the actual number of elites and the one after screening, respectively.

2. $\varepsilon_t$ denotes the adaptive $\varepsilon$ parameter at iteration $t$.

3. $\rho_t = N_t^{(e)}/N$ denotes the adaptive proposal rarity parameter at iteration $t$.

4. RE denotes the relative error. Note that for our first, second and third model we used $|\mathcal{X}^*| = 15$, $|\mathcal{X}^*| = 2258$ and $|\mathcal{X}^*| = 1$, respectively. They were obtained by using the direct estimator $|\widehat{\mathcal{X}}^*_{dir}|$ with a very large sample, namely $N = 100,000$.

## 5.1 Smoothed Splitting Algorithm

In all our numerical results we use $g_\varepsilon(y)$ in (7).

### 5.1.1 First Model: 3-SAT with matrix $A = (20 \times 80)$

Table 1 presents the performance of smoothed Algorithm 3.2 for the 3-SAT problem with an instance matrix $A = (20 \times 80)$ with $N = 1,000$, $\rho = 0.2$ and $b = 1$.

Since the true number of solution is $|\mathcal{X}^*| = 15$, following the notations of Section 4, we have that

$$\ell = \frac{15}{2^{20}} = r\ \rho^T, \text{with } T = \left\lfloor \frac{\log \ell}{\log \rho} \right\rfloor = \left\lfloor \frac{\log(15/2^{20})}{\log 0.2} \right\rfloor = 6$$

and

$$r = \ell \rho^{-T} = \frac{15}{2^{20}} 0.2^{-6} \approx 0.22.$$

Each run of the algorithm gives an estimator :

$$|\widehat{\mathcal{X}}^*| = 2^{20} \times \widehat{\ell} = 2^{20} \times (\widehat{r}\ \rho^{\widehat{T}}) = 2^{20} \times (\widehat{r}\ 0.2^{\widehat{T}}), \text{with } \widehat{r} \in (\rho, 1] = (0.2, 1].$$

In Table 1 , the column "Iterations" corresponds to $\widehat{T}+1$ for each of the 10 runs (the theoretical value is thus $T + 1 = 7$). It is indeed 7 most of the time, but sometimes jumps to 8, which is not a surprise since $r = 0.22 \approx 0.2$.

Concerning the relative error of $|\widehat{\mathcal{X}}^*|$ (RE of $|\widehat{\mathcal{X}}^*|$), Theorem 4.1 states that it should be approximately equal to

$$\frac{1}{\sqrt{N}} \sqrt{T \frac{1-\rho}{\rho} + \frac{1-r}{r}} \approx 0.17,$$

while we find experimentally (see Table 1) a relative error of 0.228. There are two main reasons for this: first we performed only 10 runs, and second we set $b = 1$, while the analysis of the i.i.d. SSM suggests $b$ to be large. Altogether, it gives the correct order of magnitude.

Concerning the relative bias of $|\widehat{\mathcal{X}}^*|$, Theorem 4.1 states that it should be

approximately equal to

$$\frac{1}{N} \times \left( T \frac{1-\rho}{\rho} \right) \approx 0.024,$$

while experimentally (see Table 1) we find a relative bias of 0.018. The comments on the bias are the same as for the relative error above.

Table 1: Performance of smoothed Algorithm 3.2 for SAT $20 \times 80$ model.

| Run $N_0$ | Iterations | $\lvert\widehat{\mathcal{X}}^*\rvert$ | RE of $\lvert\widehat{\mathcal{X}}^*\rvert$ | $\lvert\widehat{\mathcal{X}}^*_{dir}\rvert$ | RE of $\lvert\widehat{\mathcal{X}}^*_{dir}\rvert$ | CPU |
|---|---|---|---|---|---|---|
| 1 | 7 | 13.682 | 0.088 | 15 | 0 | 1.207 |
| 2 | 7 | 16.725 | 0.115 | 15 | 0 | 1.192 |
| 3 | 7 | 24.852 | 0.657 | 15 | 0 | 1.189 |
| 4 | 8 | 12.233 | 0.184 | 15 | 0 | 1.383 |
| 5 | 7 | 14.217 | 0.052 | 15 | 0 | 1.248 |
| 6 | 8 | 12.564 | 0.162 | 15 | 0 | 1.341 |
| 7 | 7 | 19.770 | 0.318 | 15 | 0 | 1.174 |
| 8 | 7 | 17.073 | 0.138 | 15 | 0 | 1.192 |
| 9 | 8 | 12.448 | 0.170 | 15 | 0 | 1.338 |
| 10 | 8 | 9.089 | 0.394 | 15 | 0 | 1.399 |
| Average | 7.4 | 15.265 | 0.228 | 15 | 0 | 1.266 |

In Figure 2, we give an illustration of the asymptotic normality, as given by theorem 4.1. The Figure compares the cdf of the limit Gaussian distribution, and the empirical distribution on 100 runs. Here $\rho = 1/2$.

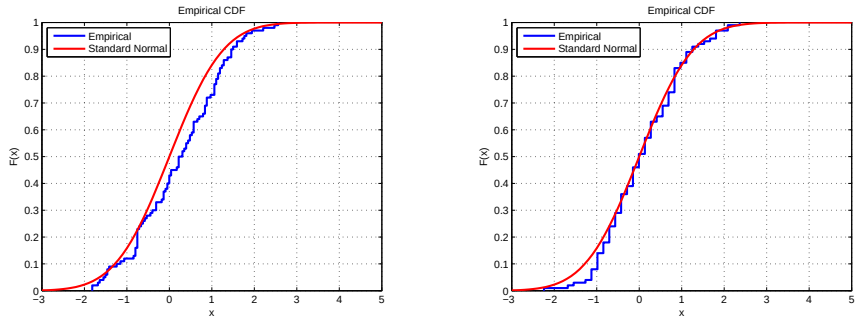


Figure 2: Asymptotic normality: empirical (100 runs) and limiting Gaussian cdf's, 1000 replicas (left) and $10,000$ (right).

### 5.1.2 Second model: Random 3-SAT with matrix $A = (75 \times 325)$.

This example is taken from `www.satlib.org`. Table 2 presents the performance of smoothed Algorithm 3.2. We set $N = 10,000$, $\rho = 0.2$ and $b = 1$ for all iterations.

Table 2: Performance of the smoothed Algorithm 3.2 for SAT $75 \times 325$ model.

| Run $N_0$ | Iterations | $\lvert\widehat{\mathcal{X}}^*\rvert$ | RE of $\lvert\widehat{\mathcal{X}}^*\rvert$ | $\lvert\widehat{\mathcal{X}}^*_{dir}\rvert$ | RE of $\lvert\widehat{\mathcal{X}}^*_{dir}\rvert$ | CPU |
|---|---|---|---|---|---|---|
| 1 | 28 | 2210.2 | 0.021 | 2254 | 0.0018 | 519.7 |
| 2 | 28 | 2750.5 | 0.218 | 2232 | 0.0115 | 518.0 |
| 3 | 28 | 1826.1 | 0.191 | 2248 | 0.0044 | 523.6 |
| 4 | 28 | 2403.3 | 0.064 | 2254 | 0.0018 | 524.3 |
| 5 | 28 | 2189.6 | 0.030 | 2250 | 0.0035 | 519.3 |
| 6 | 28 | 1353.6 | 0.401 | 2254 | 0.0018 | 524.5 |
| 7 | 28 | 2572.8 | 0.139 | 2214 | 0.0195 | 528.6 |
| 8 | 28 | 2520.0 | 0.116 | 2246 | 0.0053 | 525.2 |
| 9 | 28 | 2049.2 | 0.092 | 2208 | 0.0221 | 521.8 |
| 10 | 28 | 2827.3 | 0.252 | 2244 | 0.0062 | 528.8 |
| Average | 28 | 2270.3 | 0.153 | 2240.4 | 0.0078 | 523.4 |

It follows from Table 2 that the average relative error of the product estimator $|\widehat{\mathcal{X}}^*|$ is RE $= 0.163$ and of the direct estimator $|\widehat{\mathcal{X}}^*_{dir}|$ is only RE $= 0.038$.

### 5.1.3 Third Model: Random $3-4$-SAT with matrix $A = (122 \times 663)$.

Our last model is the random 3-SAT with the instance matrix $A = (122 \times 663)$ and a single valid assignment, that is $|\mathcal{X}^*| = 1$, taken from `http://www.is.titech.ac.jp/~watanabe/gensat`. We set $N = 50,000$ and $\rho = 0.4$ for all iterations.

We found that the the average CPU time is about 3 hours for each run, the average relative error for the product estimator $|\widehat{\mathcal{X}}^*|$ is RE $= 0.15$, while for the direct estimator $|\widehat{\mathcal{X}}^*_{dir}|$ it is RE $= 0.1$. This means that in 9 out of 10 runs SSM finds the unique SAT assignment.

## 5.2 Splitting Algorithm

### 5.2.1 First Model: 3-SAT with matrix $A = (20 \times 80)$

Table 3 presents the performance of the improved splitting Enhanced Cloner Algorithm [16] for the 3-SAT problem with an instance matrix $A = (20 \times 80)$ with $N = 1,000$, $\rho = 0.2$ and $b = 1$.

Table 3: Performance of Enhanced Cloner Algorithm [16] for SAT $20 \times 80$ model.

| Run $N_0$ | Iterations | $\lvert\widehat{\mathcal{X}}^*\rvert$ | RE of $\lvert\widehat{\mathcal{X}}^*\rvert$ | $\lvert\widehat{\mathcal{X}}^*_{dir}\rvert$ | RE of $\lvert\widehat{\mathcal{X}}^*_{dir}\rvert$ | CPU |
|---|---|---|---|---|---|---|
| 1 | 10 | 17.316 | 0.154 | 15 | 0 | 0.641 |
| 2 | 10 | 15.143 | 0.010 | 15 | 0 | 0.640 |
| 3 | 10 | 12.709 | 0.153 | 15 | 0 | 0.645 |
| 4 | 9 | 16.931 | 0.129 | 15 | 0 | 0.566 |
| 5 | 10 | 13.678 | 0.088 | 15 | 0 | 0.644 |
| 6 | 9 | 15.090 | 0.006 | 15 | 0 | 0.565 |
| 7 | 9 | 10.681 | 0.288 | 15 | 0 | 0.558 |
| 8 | 10 | 13.753 | 0.083 | 15 | 0 | 0.661 |
| 9 | 10 | 14.022 | 0.065 | 15 | 0 | 0.646 |
| 10 | 10 | 13.445 | 0.104 | 15 | 0 | 0.651 |
| Average | 9.7 | 14.277 | 0.108 | 15 | 0 | 0.622 |

### 5.2.2 Second model: Random 3-SAT with matrix $A = (75 \times 325)$.

This example is taken from `www.satlib.org`. Table 4 presents the performance of the Enhanced Cloner Algorithm [16]. We set $N = 10,000$ and $\rho = 0.1$ and $b = \eta$ for all iterations until the Enhanced Cloner Algorithm reached the desired level 325, (recall that $b$ is the number of Gibbs cycles and $\eta$ is the number of splitting of each trajectory). After that, at the last iteration, we switched to $N = 100,000$.

Table 4: Performance of the Enhanced Cloner Algorithm [16] for SAT $75 \times 325$ model.

| Run $N_0$ | Iterations | $\lvert\widehat{\mathcal{X}}^*\rvert$ | RE of $\lvert\widehat{\mathcal{X}}^*\rvert$ | $\lvert\widehat{\mathcal{X}}^*_{dir}\rvert$ | RE of $\lvert\widehat{\mathcal{X}}^*_{dir}\rvert$ | CPU |
|---|---|---|---|---|---|---|
| 1 | 24 | 2458.8 | 0.089 | 2220 | 0.017 | 640.8 |
| 2 | 24 | 1927.8 | 0.146 | 2224 | 0.015 | 673.8 |
| 3 | 24 | 1964.6 | 0.130 | 2185 | 0.032 | 664.5 |
| 4 | 24 | 2218.9 | 0.017 | 2216 | 0.019 | 661.3 |
| 5 | 24 | 2396.9 | 0.062 | 2191 | 0.030 | 678.1 |
| 6 | 24 | 2271.8 | 0.006 | 2230 | 0.012 | 661 |
| 7 | 24 | 2446.1 | 0.083 | 2202 | 0.025 | 695 |
| 8 | 24 | 2090.5 | 0.074 | 2200 | 0.026 | 711.7 |
| 9 | 24 | 2147.7 | 0.049 | 2213 | 0.020 | 696.8 |
| 10 | 24 | 2395 | 0.061 | 2223 | 0.016 | 803.3 |
| Average | 24 | 2231.8 | 0.072 | 2210.4 | 0.021 | 688.6 |

It is interesting to note that if we set $b = 1$ instead of $b = \eta$, the average relative error of both the product and the direct estimators of Enhanced Cloner Algorithm [16] substantially increases. They become 0.27 and 0.16 instead of 0.072 and 0.021, respectively (see Table 4). This is in turn worse than 0.153 and 0.0078, the average relative errors of the product estimator of SSM Algorithm 3.2 (see Table 2). It is also important to note that by setting $b \neq 1$ in the SSM Algorithm 3.2, in particular setting $b = \eta$ we found that both relative errors remain basically close to these for $b = 1$. This means that one full cycle of the Gibbs sampler suffices for Algorithm 3.2, while the Basic Splitting Algorithm of [16] requires tuning of $b$. In other words, the SSM Algorithm 3.2 is robust with respect to $b$, while its counterpart Basic Splitting Algorithm is not.

### 5.2.3 Third Model: Random 3-4-SAT with matrix $A = (122 \times 663)$

Similar to SSM Algorithm 3.2 we set $N = 10,000$ and $\rho = 0.1$ for all iterations until Enhanced Cloner Algorithm [16] has reached the desired level 663. After that we switched to $N = 100,000$ for the last iteration. Again, as for the second model, we set here $b = \eta$ instead of $b = 1$ as for SSM Algorithm 3.2.

The the average CPU time is about 2 hours for each run, the average relative error for the product estimator $\lvert\widehat{\mathcal{X}}^*\rvert$ is RE $= 0.23$, while for the direct estimator

$|\widehat{\mathcal{X}}^*_{dir}|$ it is RE = 0.4. This means that only in 6 out of 10 runs Enhanced Cloner Algorithm [16] finds the unique SAT assignment (compare this with 9 out of 10 runs for SSM Algorithm 3.2).

The above numerical results can be summarized as follows:

The proposed smoothed splitting method performs similarly to the standard splitting one (in terms of CPU time and variability).

The proposed method is robust, while the standard splitting is not, especially for the more difficult models, such as the Second and the Third Models. This means that parameters $\rho$ and $N$ in the former method can be chosen from a wide range, while in the latter they require careful tuning.

# References


[1] Botev, Z.I.; Kroese, D.P. Efficient Monte Carlo Simulation via The Generalized Splitting Method. Statistics and Computing. **2010**.

[2] Botev, Z.I.; Kroese, D.P. An Efficient Algorithm for Rare-event Probability Estimation, Combinatorial Optimization, and Counting, Methodology and Computing in Applied Probability. **2008**, *10*, 471-505.

[3] Cerou, F.; Del Moral, P.; Furon, T.; Guyader, A.. Sequential Monte Carlo for Rare Event Estimation. Statistics and Computing. (to appear), **2011**.

[4] Cerou, F.; Del Moral, P.; Le Gland, F.; Lezaud, P. Genetic genealogical models in rare event analysis. Latin American Journal of Probability and Mathematical Statistics (Alea). **2006**, *1*, 181-203.

[5] Cerou, F.; Guyader, A. Adaptive multilevel splitting for rare event analysis. Stoch. Anal. Appl. **2007**,*25*, 417-443.

[6] Del Moral, P. *Feynman-Kac Formulae Genealogical and Interacting Particle Systems*, Springer: New York, 2004.

[7] Garvels, M.J.J. *The splitting method in rare-event simulation*, Ph.D. thesis, University of Twente, 2000.

[8] Garvels, M.J.J.; Rubinstein, R.Y. A Combined Splitting - Cross Entropy Method for Rare Event Probability Estimation of Single Queues and ATM Networks. Unpublished Manuscript.

[9] Lagnoux-Renaudie, A. A two-steps branching splitting model under cost constraint. Journal of Applied Probability. **2009**, *46*, 429-452.

[10] Melas, V..B. On the efficiency of the splitting and roulette approach for sensitivity analysis. Winter Simulation Conference, Atlanta, Georgia. **1997**, 269-274.

[11] L'Ecuyer, P.; Demers, V.; Tuffin, B. Rare-Events, Cloning, and Quasi-Monte Carlo, ACM Transactions on Modeling and Computer Simulation. **2007**, *17*.

[12] Mitzenmacher, M. and Upfal, E.. *Probability and Computing : Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press: New York, 2005.

[13] Motwani, R.; Raghavan, R. *Randomized Algorithms*, Cambridge University Press, 1997.

[14] Ross, S.M. *Simulation*, Wiley, 2006.

[15] Rubinstein, R.Y. The Gibbs Cloner for Combinatorial Optimization, Counting and Sampling. Methodology and Computing in Applied Probability. **2009**, *11*, 491-549.

[16] Rubinstein, R.Y. Randomized Algorithms with Splitting: Why the Classic Randomized Algorithms do not Work and how to Make them Work. Methodology and Computing in Applied Probability. **2010**, *12*, 1-41.

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

[18] Rubinstein, R.Y.; Kroese, D.P. *Simulation and the Monte Carlo Method; Second Edition*, Wiley, 2007.

[19] van der Vaart, A.W. *Asymptotic Statistics*, Cambridge University Press, 1998.