# The Splitting Method for Decision Making

*Reuven Rubinstein, Andrey Dolgin and Radislav Vaisman*

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

December 6, 2009

**Abstract**

We show how a simple modification of the splitting method based on Gibbs sampler can be efficiently used for decision making in the sense that one can efficiently decide whether or not a given set of integer program constraints has at least one feasible solution. We also show how to incorporate the classic *capture-recapture* method into the splitting algorithm in order to obtain a low variance estimator for the counting quantity representing, say the number of feasible solutions on the set of the constraints of an integer program. We finally present numerical with with both, the decision making and the capture-recapture estimators and show their superiority as compared to the conventional one, while solving quite general decision making and counting ones, like the satisfiability problems.

**Keywords.** Decision Making, Gibbs Sampler, Cross-Entropy, Rare-Event, Combinatorial Optimization, Counting, Splitting.

[0†] This research was supported by the BSF (Binational Science Foundation, grant No 2008482)

# Contents

**1 Introduction: The Splitting Method** **3**

**2 Decision Making** **6**

**3 Counting with the Capture-Recapture Method** **7**
3.1 Application of the Classic Capture Recapture . . . . . 7
3.2 Application of the On-line Capture Recapture . . . . . 8

**4 Numerical Results** **8**
4.1 Counting . . . . . 9
4.2 Decision Making . . . . . 11

**5 Concluding Remarks** **12**

**6 Appendix: Splitting Algorithms** **13**
6.1 Basic Splitting Algorithm . . . . . 13
6.2 Enhanced Splitting Algorithm for Counting . . . . . 14
6.3 Direct Splitting Algorithm . . . . . 16

# 1 Introduction: The Splitting Method

In this work we show how a simple modification of the splitting method introduced in [14, 15] can be used for decision making. The goal of the decision making algorithm is to decide whether or not a discrete set, like the set defined by the integer programming constraints has a feasible solution.

Although there is a vast literature on the splitting method, (see [3], [4], [5], [6], [7], [9], [15]), we follow [15] and present some background on the splitting method, also called in [15] the *cloning* method. The main idea 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 $\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_m$ and such that $\mathcal{X}_m = \mathcal{X}^*$. Typically, 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 [17].

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 $\mathcal{X} = \mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_m$ such that $\mathcal{X}_0 \supset \mathcal{X}_1 \supset \cdots \supset \mathcal{X}_m = \mathcal{X}^*$, $|\mathcal{X}_m| = |\mathcal{X}^*|$ and $|\mathcal{X}| = |\mathcal{X}_0|$ is known.

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

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

   where $\ell = \prod_{t=1}^{m} \frac{|\mathcal{X}_t|}{|\mathcal{X}_{t-1}|}$. 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

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

   where $\widehat{\ell} = |\mathcal{X}_0| \prod_{t=1}^{m} \widehat{c}_t$.

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

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

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

To proceed note that $\ell$ can be also written as

$$\ell = \mathbb{E}_f \left[ I_{\{S(\boldsymbol{X}) \geq m\}} \right], \tag{4}$$

where $\boldsymbol{X} \sim f(\boldsymbol{x})$, $f(\boldsymbol{x})$ is a uniform distribution on the set of points of $\mathcal{X} = \mathcal{X}_0$, $m$ is a fixed parameter, like the total number of constraints in an integer program, and $S(\boldsymbol{X})$ is the sample performance, like the number of feasible solution generated by the constraints of the integer program. It can be also written (see(1)) as

$$\ell = \prod_{t=1}^{T} c_t, \tag{5}$$

where

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

Here

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

$\ell(m_{t-1})^{-1}$ is the normalization constant and similar to (1) the sequence $m_t,\ t = 0, 1, \ldots, T$ represents a fixed grid satisfying $-\infty < m_0 < m_1 < \cdots < m_T = m$. Note that in contrast to (1) we use in (5) a product of $T$ terms instead of a product of $m$ terms. Note that $T$ might be a random variable. The later case is associated with adaptive choice of the level sets $\{\widehat{m}_t\}_{t=0}^{T}$ resulting in $T \leq m$. Since for counting problems the pdf $f(\boldsymbol{x})$ should be *uniformly* distributed on $\mathcal{X}$, which we denote by $\mathcal{U}(\mathcal{X})$, it follows from (7) that the pdf $g^*(\boldsymbol{x}, m_{t-1})$ must be *uniformly* distributed on the set $\mathcal{X}_t = \{\boldsymbol{x} : S(\boldsymbol{x}) \geq m_{t-1}\}$, that is, $g^*(\boldsymbol{x}, m_{t-1})$ must be equal to $\mathcal{U}(\mathcal{X}_t)$. Although the pdf $g^*_{t-1} = \mathcal{U}(\mathcal{X}_t)$ is typically not available analytically, it is shown in [14, 15] that one can sample from it by using the MCMC method and in particular the Gibbs sampler, and as the result to update the parameters $c_t$ and $m_t$ adaptively. This is one of the most crucial issues of the cloning method.

Once sampling from $g^*_{t-1} = \mathcal{U}(\mathcal{X}_t)$ becomes available, the final estimator of $\ell$ (based on the estimators of $c_t = \mathbb{E}_{g^*_{t-1}}[I_{\{S(\boldsymbol{X})\geq m_{t-1}\}}],\ t = 0, \ldots, T$), can be written as

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

where

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

$N_t = \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i)\geq m_{t-1}\}}$, $\boldsymbol{X}_i \sim g^*_{t-1}$ and $g^*_{-1} = f$.

We next show how to cast the problem of counting the number of feasible solutions of the set of integer programming constraints into the framework (4)- (7).

**Example 1.1 Counting on the set of an integer programming constraints** Consider the set $\mathcal{X}^*$ containing both equality and inequality constraints of an integer program, that is,

$$\begin{aligned} &\textstyle\sum_{k=1}^{n} a_{ik} x_k = b_i,\ i = 1, \ldots, m_1, \\ &\textstyle\sum_{k=1}^{n} a_{jk} x_k \geq b_j,\ j = m_1 + 1, \ldots, m_1 + m_2, \\ &\boldsymbol{x} = (x_1, \ldots, x_n) \geq \boldsymbol{0},\quad x_k \text{ is integer } \forall k = 1, \ldots, n. \end{aligned} \tag{10}$$

Our goal is to count the number of feasible solutions (points) of the set (10). We assume that each component $x_k,\ k = 1, \ldots, n$ has $d$ different values, labeled $1, \ldots, d$. Note that the SAT problem represents a particular case of (10) with inequality constraints and where $x_1, \ldots, x_n$ are binary components. If not stated otherwise we

will bear in mind the counting problem on the set (10) and in particular counting the number of true (valid) assignments in a SAT problem.

It is shown in [15] that in order to count the number of points of the set (10) one can associate it with the following rare-event probability problem

$$\ell = \mathbb{E}_f \left[ I_{\{S(\boldsymbol{X})=m\}} \right] = \mathbb{E}_f \left[ I_{\{\sum_{i=1}^m C_i(\boldsymbol{X})=m\}} \right], \tag{11}$$

where the first $m_1$ terms $C_i(\boldsymbol{X})$'s in (11) are

$$C_i(\boldsymbol{X}) = I_{\{\sum_{k=1}^n a_{ik} X_k = b_i\}}, \; i = 1, \ldots, m_1, \tag{12}$$

while the remaining $m_2$ ones are

$$C_i(\boldsymbol{X}) = I_{\{\sum_{k=1}^n a_{ik} X_k \geq b_i\}}, \; i = m_1 + 1, \ldots, m_1 + m_2 \tag{13}$$

and $S(\boldsymbol{X}) = \sum_{i=1}^m C_i(\boldsymbol{X})$. Thus, in order to count the number of feasible solutions on the set (10) one can consider an associated rare-event probability estimation problem (11) involving a *sum of dependent Bernoulli random variables* $C_i$ $i = m_1 + 1, \ldots, m$, and then apply $\widehat{|\mathcal{X}^*|} = \widehat{\ell}|\mathcal{X}|$. In other words, in order to count on $\mathcal{X}^*$ one needs to estimate efficiently the rare event probability $\ell$ in (11). A rare-event probability estimation framework similar to (11) can be readily established for many NP-hard counting problems [15].

It follows from above that the proposed algorithm will generate an adaptive sequence of tuples

$$\{(m_0, g^*(\boldsymbol{x}, m_{-1})), \; (m_1, g^*(\boldsymbol{x}, m_0)), \; (m_2, g^*(\boldsymbol{x}, m_1)), \ldots, (m_T, g^*(\boldsymbol{x}, m_{T-1}))\} \tag{14}$$

Here as before $g^*(\boldsymbol{x}, m_{-1}) = f(\boldsymbol{x}) = \mathcal{U}(\mathcal{X})$, $g^*(\boldsymbol{x}, m_t) = \mathcal{U}(\mathcal{X}_t)$, and $m_t$ is obtained from the solution of the following non-linear equation

$$\mathbb{E}_{g^*_{t-1}} I_{\{S(\boldsymbol{X}) \geq m_t\}} = \rho, \tag{15}$$

where $\rho$ is called the *rarity* parameter [17]. Typically one sets $0.1 \leq \rho 0.01$. Note that in contrast to the classic cross-entropy (CE) method [13], [16], where one generates a sequence of tuples

$$\{(m_0, \boldsymbol{v}_0), \; (m_1, \boldsymbol{v}_1), \ldots, (m_T, \boldsymbol{v}_T)\}, \tag{16}$$

and, where $\{\boldsymbol{v}_t, \; t = 1, \ldots, T\}$ is a sequence of parameters in the parametric family of distributions $f(\boldsymbol{x}, \boldsymbol{v}_t)$, here in (14), $\{g^*(\boldsymbol{x}, m_{t-1}) = g^*_{t-1}, \; t = 0, 1, \ldots, T\}$ *is a sequence of non-parametric IS distributions.* Otherwise, the CE and the splitting algorithms are very similar.

In Appendix (see Section 6), following [15], we present two versions of the splitting algorithm: the so-called *basic* version and the *enhanced* version having in mind Example 1.1. Here we also present what is called the *direct* estimator and an associated Algorithm 6.3, which can be viewed as an alternative to the conventional product estimator $\widehat{|\mathcal{X}^*|}$ generated by Algorithm 6.2. This estimator is based on *direct counting* of the number of samples obtained immediately after crossing the level $m$, that is without involving the product of $\widehat{c}_t$. The drawback of the direct Algorithm 6.3 is that it is able to count only if $|\mathcal{X}^*|$ is up to the order of thousands.

Note that the splitting algorithm in [15] is also suitable for optimization. Here we shal use the same sequence of tuples (14), but *without involving the product of the estimators* $\widehat{c}_t, \; t = 1, \ldots, T$.

The rest of the paper is organized as follows. In Section 2 we present two heuristics for speeding up the direct splitting Algorithm 6.3. They are called (i)

*local* $m_t$ *updating* and (ii) *global* $m_t$ *updating*, respectively and will be used for decision making. Recall that decision making here is merely to decide whether or not the set $\mathcal{X}^*$ of the integer programming constraints (10) has a feasible solution. Section 3 shows how to combine the well known *capture-recapture* (CAP-RECAP) method with splitting in order to obtain a low variance alternative to both the product estimator $\widehat{|\mathcal{X}^*|}$ in (3) and the direct estimator $\widehat{|\mathcal{X}^*_{dir}|}$ in (22). Note that the CAP-RECAP estimator (see (17) below) can be viewed as a generalization of the direct one $\widehat{|\mathcal{X}^*_{dir}|}$ in the sense that once $m_t = m$ it involves two Gibbs samples instead of one sample. In Section 4 supportive numerical results are presented. In particular we show that the CAP-RECAP estimator outperforms the product one $\widehat{|\mathcal{X}^*|}$. Finally, in Section 5 some concluding remarks are given.

# 2 Decision Making

Here we present two heuristics for speeding up the direct Algorithm 6.3, which will be used for decision making. They are called (i) *local* $m_t$ *updating* and (ii) *global* $m_t$ *updating*, respectively. It is important to note that they are applicable only for the direct estimator (see (22) below), but not for the product one (8).

**Local $m_t$ updating** In this version, at each iteration $t$ we replace the fixed $m_t$ value in the Gibbs sampler with the elite sample values at that iteration. To clarify, consider for simplicity the sum of $n$ Bernoulli random variables. Let for concreteness $n = 5$, $N = 100$ and $\rho = 0,01$. Assume that while taking the sample of size $N = 100$ we obtained the following sequence of elite vectors $\boldsymbol{X}_{t1} = (1,1,0,1,0)$, $\boldsymbol{X}_{t2} = (1,0,0,1,0)$, $\boldsymbol{X}_{t3} = (1,0,1,1,0)$, $\boldsymbol{X}_{t4} = (1,0,0,1,0)$, $\boldsymbol{X}_{t5} = (1,0,0,1,1)$. The corresponding elite sample values are $S(\boldsymbol{X}_{t1}) = 3$, $S(\boldsymbol{X}_{t2}) = 2$, $S(\boldsymbol{X}_{t3}) = 3$, $S(\boldsymbol{X}_{t4}) = 2$, $S(\boldsymbol{X}_{t5}) = 3$, and clearly $m_t = 2$. Thus, in this version we simple replace $m_t = 2$ by the corresponding $S(\boldsymbol{X})$ elite values $3,\ 2,\ 3,2,\ = 3$, while all ather data in the direct Algorithm 6.3 remaining the same.

**Global $m_t$ updating** In this version we want the sample performance $S(\boldsymbol{X})$ to be a non-decreasing function as Gibbs proceeds, like the one in Figure **??** corresponding to the non-decreasing random work. To clarify, assume that at iteration $t$ we have the same elite sample as before, that is $\boldsymbol{X}_{t1} = (1,1,0,1,0)$, $\boldsymbol{X}_{t2} = (1,0,0,1,0)$, $\boldsymbol{X}_{t3} = (1,0,1,1,0)$, $\boldsymbol{X}_{t4} = (1,0,0,1,0)$, $\boldsymbol{X}_{t5} = (1,0,0,1,1)$ with $m_t = 2$. Let us pick up one of the elites, say the first one corresponding to $\boldsymbol{X}_{t1} = (1,1,0,1,0)$ and let us apply to it the systematic Gibbs sampler. Noticing that $\boldsymbol{X}_{t1}$ has 3 unities, we simply replace the original level $m_t = 2$ by the current value $S(\boldsymbol{X}_{t1}) = 3$ and set a new sub-level $m_{t1} = S(\boldsymbol{X}_{t1}) = 3$. We then proceed from $\boldsymbol{X}_{t1} = (1,1,0,1,0)$ with $m_{t1} = 3$ to a new value denoted as $\boldsymbol{X}^{(1)}_{t1}$. Assume that while applying systematic Gibbs sampler to the first components we obtained $\boldsymbol{X}^{(1)}_{t1} = (1,1,0,1,0)$, that is $\boldsymbol{X}^{(1)}_{t1} = \boldsymbol{X}_{t1} = (1,1,0,1,0)$. We next proceed from $\boldsymbol{X}^{(1)}_{t1} = (1,1,0,1,0)$ (with $m_{t2} = 3$) to a new vector $\boldsymbol{X}^{(2)}_{t1}$ by applying systematic Gibbs sampler to the second components. Let $\boldsymbol{X}^{(2)}_{t1} = (1,1,1,1,0)$. Since $\boldsymbol{X}^{(2)}_{t1} = (1,1,1,1,0)$ contains 4 unities, we set the new sub-level $m_{t3} = 4$. Assume that proceeding further we obtain $m_{t5} = m_{t4} = m_{t3} = 4$ and let also $\boldsymbol{X}^{(4)}_{t1} = \boldsymbol{X}^{(3)}_{t1} = \boldsymbol{X}^{(2)}_{t1} = (1,1,1,1,0)$. The resulting sequence of sub-levels $m_{ti}$ in the systematic Gibbs sampler starting at $\boldsymbol{X}_{t1} = (1,1,0,1,0)$ is therefore $(m_{t1}, m_{t2}, m_{t3}, m_{t4}, m_{t5}) = (3,3,4,4,4)$ and similar for the remaining four elite values $\boldsymbol{X}_{t2} = (1,0,0,1,0)$, $\boldsymbol{X}_{t3} = (1,0,1,1,0)$, $\boldsymbol{X}_{t4} = (1,0,0,1,0)$, $\boldsymbol{X}_{t5} = (1,0,0,1,1)$. After that we define a new common level $m_{t+1}$ for iteration $t+1$. In Section 4 we present some numerical results with the above heuristics.

# 3 Counting with the Capture-Recapture Method

Here we show how the well known *capture-recapture* (CAP-RECAP) method can be used as alternative to the product estimator $\widehat{|\mathcal{X}^*|}$ in (3).

We consider two versions of (CAP-RECAP) (i) the classic one (ii) the proposed on-line one.

## 3.1 Application of the Classic Capture Recapture

Originally the capture-recapture method was used to estimate the size, say $M$, of unknown population, under the assumption that *two* independent samples are taken from that population.

To see how the CAP-RECAP method works consider an urn model model with a total of $M$ identical balls. Denote by $N_1$ and $N_2$, the sample sizes taken at the first and the second draw. Assume in addition that

1. The second draw take place only after all $N_1$ balls are returned back to the urn.
2. Before returning the $N_1$ balls back we *mark* each of them, say we paint them in a different color.

Denote by $R$ the number of balls from the first draw that also appear at the second one. Clearly that the estimate of $M$, denoted by $\widetilde{M}$ is

$$\widetilde{M} = \frac{N_1 N_2}{R}.$$

This is so since

$$\frac{N_2}{M} \approx \frac{R}{N_1}.$$

Note that the name *capture-recapture* comes from the name of model where one is interested to estimate the animal population size in a particular area, provided two visits are available to the area. In this case $R$ denotes the number of animals captured on the first visit that were then recaptured on the second one.

It is well know that a slightly better unbiased estimate of $M$ is

$$\widehat{M} = \frac{(N_1+1)(N_2+1)}{(R+1)} - 1. \tag{17}$$

The corresponding variance is

$$\boldsymbol{Var}\ (\widehat{M}) = \frac{(N_1+1)(N_2+1)(N_1-R)(N_2-R)}{(R+1)(R+2)(R+3)}. \tag{18}$$

Application of The CAP-RECAP to counting problems is trivial. We set $|\mathcal{X}^*| = M$ and note that $N_1$ and $N_2$ correspond to the screened out Gibbs samples at the first and second draws, which are performed after Algorithm 6.2 reaches the desired level $m$.

As an example, assume that in both experiments (draws) we set originally $N = 10,000$ and then we obtained $N_1 = 5,000$, $N_2 = 5,010$ and $R = 10$. The capture-recapture (CAP-RECAP) estimator of $|\mathcal{X}^*|$, denoted by $\widehat{|\mathcal{X}^*|_{cap}}$ is therefore

$$\widehat{|\mathcal{X}^*|_{cap}} = 2,505,000.$$

Clearly, the direct estimator $\widehat{|\mathcal{X}^*|_{dir}}$ can not handle such big number.

Our numerical results below clearly indicate that the CAP-RECAP estimator $\widehat{|\mathcal{X}^*|_{cap}}$ is typically more accurate than the product one $\widehat{|\mathcal{X}^*|}$, that is

$$\textbf{\textit{Var}}\ \widehat{|\mathcal{X}^*|} > \textbf{\textit{Var}}\ \widehat{|\mathcal{X}^*|_{cap}},$$

provided that the the sample $N$ is limited, say by 10,000 and if $|\mathcal{X}^*|$ is large but limited, say by $10^7$. If, however, $|\mathcal{X}^*|$ is very large, say $|\mathcal{X}^*| > 10^7$, then $\widehat{|\mathcal{X}^*|_{cap}}$ might become meaningless, since with the budget of $N = 10,000$ we will obtain often $R = 0$, provided $|\mathcal{X}^*| > 10^7$. However, the latter case has limited application, since if $|\mathcal{X}^*|$ is very large we can estimate it with the crude Monte Carlo.

### 3.2 Application of the On-line Capture Recapture

To see how the CAP-RECAP method works on-line consider again the urn model with a total of $M$ unknown identical balls. In this case we take only *one draw* of size $N$ instead of two ones (of sizes $N_1$ and $N_2$, respectively as before) and proceed as follows:

1. We draw the $N$ balls *one-by-one with replacement*.
2. Before returning each of the $N$ balls back to the urn we *mark* it.
3. As in classic method we count the number of marked balls.

Note that the marking procedure here is different from the classic one. Also note that by drawing the $N$ balls on-line (sequentially), each ball can be drawn with positive probability up to $N$ times, while in the classic one each ball has a positive probability to be drawn only up to *two* times. Now having $R$ marked balls at hand how can we find the on-line estimator of $M$, denoted by $\widehat{|\mathcal{X}^*_{on}|}$ (recall that in our case $M = |\mathcal{X}^*|$), its expected value and the associated variance. We proceed argue as follows.

The probability that the same ball will appear exactly $N$ times is $(1/M)^N$; exactly 2 times is $N(1-1/M)(1/M)^{N-1}$, etc.

Proceeding we can readily obtain that the on-line estimator of $M$ and the associated variance. Although this will be done some where else, it is intuitively clear that the on-line CAP-RECAP estimator $\widehat{|\mathcal{X}^*_{on}|}$ is more exact than the classic one $\widehat{|\mathcal{X}^*_{cap}|}$, provided $N = N_1 + N_2$. The reason is that the on-line one is based on conditioning and conditioning always reduces variance.

**Remark 3.1** As a third alternative to both the classic and the on-line CAP-RECAP estimators we can use the direct estimator $\widehat{|\mathcal{X}^*_{dir}|}$ in (22) to estimate $M$ by taking into account the number of screened out elements at the level $m$, which is equal to $N - \widehat{|\mathcal{X}^*_{dir}|}$. As an estimator of $\mathcal{X}^*$, denoted by $\widehat{|\mathcal{X}^*_{scr}|}$ we can take the following one.

$\widehat{|\mathcal{X}^*_{scr}|} = \widehat{|\mathcal{X}^*_{dir}|}$ if $\widehat{|\mathcal{X}^*_{dir}|} \leq N/2$ and $\widehat{|\mathcal{X}^*_{scr}|} = \frac{N^2}{\widehat{|\mathcal{X}^*_{dir}|}}$, otherwise.

## 4 Numerical Results

Below we present numerical results with CAP-RECAP method for counting and with the global $m_t$ heuristics for decision making.

## 4.1 Counting

Consider the random 3-SAT with the instance matrix $A = (75 \times 325)$ taken from www.satlib.org and its truncated version $A = (75 \times 305)$.

Table 1 presents comparison of the performance of the product estimator $\widehat{|\mathcal{X}^*|}$ and its counterpart $\widehat{|\mathcal{X}^*_{cap}|}$ using the enhanced splitting Algorithm 6.2 for $A = (75 \times 305)$. Table 2 presents similar data for $A = (75 \times 325)$. We set $N = 10,000$, $\rho = 0.1$ and $b = \eta$. Table 3 presents the dynamic of one of the runs of Algorithm 6.2 for $A = (75 \times 305)$. We found that the the average CPU time is about 6 minutes for each run.

Note that the sample $N_1$ was obtained as soon as soon as Algorithm 6.2 reaches the final level $m$, and $N_2$ was obtained while runing Algorithm 6.2 for one more iteration at the same level $m$. The actual sample sizes $N_1$ and $N_2$ were chosen according to the following rule: *sample until Algorithm 6.2 screens out 50% of the samples and then stop*. It follows from Tables 1 that for model $A = (75 \times 305)$ this corresponds to $N_1 \approx N_2 \approx 26,000$ and $R \approx 22,000$, while for model $A = (75 \times 325)$ it follows from Table 2 that $N_1 \approx N_2 \approx R \approx 2,200$. It also follows that for $A = (75 \times 305)$ and $A = (75 \times 325)$ the relative error of $\widehat{|\mathcal{X}^*_{cap}|}$ is about 10 and 100 times smaller as compared to $\widehat{|\mathcal{X}^*|}$. It is readily seen that by enlarging the samples $N_1$ and $N_2$ only at the last two iterations of Algorithm 6.2 the relative error of $\widehat{|\mathcal{X}^*_{cap}|}$ will further decrease.

Table 1: Comparison of the performance of the product estimator $\widehat{|\mathcal{X}^*|}$ with its counterpart $\widehat{|\mathcal{X}^*_{cap}|}$ for SAT $(75 \times 305)$ model.

| Run $N_0$ | Iterations | $\widehat{\lvert\mathcal{X}^*\rvert}$ | RE of $\widehat{\lvert\mathcal{X}^*\rvert}$ | $\widehat{\lvert\mathcal{X}^*_{cap}\rvert}$ | RE of $\widehat{\lvert\mathcal{X}^*_{cap}\rvert}$ | $N_1$ | $N_2$ | $R$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 21 | 2.67E+04 | 1.39E-01 | 3.07E+04 | 4.49E-02 | 23993 | 23908 | 18681 |
| 2 | 21 | 4.10E+04 | 3.22E-01 | 3.27E+04 | 1.57E-02 | 27064 | 26945 | 22333 |
| 3 | 21 | 2.85E+04 | 8.08E-02 | 3.19E+04 | 7.33E-03 | 26638 | 26567 | 22176 |
| 4 | 21 | 2.96E+04 | 4.36E-02 | 3.09E+04 | 3.83E-02 | 23907 | 23993 | 18552 |
| 5 | 21 | 2.87E+04 | 7.29E-02 | 3.29E+04 | 2.41E-02 | 26967 | 27120 | 22214 |
| 6 | 21 | 3.63E+04 | 1.71E-01 | 3.23E+04 | 4.25E-03 | 26838 | 26762 | 22247 |
| 7 | 21 | 2.39E+04 | 2.28E-01 | 3.30E+04 | 2.64E-02 | 26719 | 26697 | 21618 |
| 8 | 21 | 4.10E+04 | 3.22E-01 | 3.29E+04 | 2.32E-02 | 26842 | 26878 | 21933 |
| 9 | 21 | 2.72E+04 | 1.23E-01 | 3.21E+04 | 1.44E-03 | 26645 | 26578 | 22060 |
| 10 | 21 | 2.70E+04 | 1.29E-01 | 3.21E+04 | 1.75E-03 | 26512 | 26588 | 21965 |
| Average | 21 | 3.10E+04 | 1.63E-01 | 3.21E+04 | 1.87E-02 | | | |
| Variance | 0 | 3.77E+07 | 9.71E-03 | 6.45E+05 | 2.34E-04 | | | |

Table 2: Comparison of the performance of the product estimator $\widehat{|\mathcal{X}^*|}$ and its counterpart $\widehat{|\mathcal{X}^*_{cap}|}$ for SAT $(75 \times 325)$ model.

| Run $N_0$ | Iterations | $\widehat{\lvert\mathcal{X}^*\rvert}$ | RE of $\widehat{\lvert\mathcal{X}^*\rvert}$ | $\widehat{\lvert\mathcal{X}^*_{cap}\rvert}$ | RE of $\widehat{\lvert\mathcal{X}^*_{cap}\rvert}$ | $N_1$ | $N_2$ | $R$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 24 | 2.02E+03 | 1.03E-02 | 2.21E+03 | 6.55E-03 | 2201 | 2195 | 2191 |
| 2 | 24 | 1.94E+03 | 2.95E-02 | 2.20E+03 | 7.01E-03 | 2200 | 2202 | 2198 |
| 3 | 24 | 1.59E+03 | 2.03E-01 | 2.24E+03 | 7.86E-03 | 2234 | 2235 | 2232 |
| 4 | 24 | 2.34E+03 | 1.70E-01 | 2.23E+03 | 2.45E-03 | 2221 | 2223 | 2219 |
| 5 | 24 | 1.69E+03 | 1.54E-01 | 2.20E+03 | 1.11E-02 | 2194 | 2191 | 2190 |
| 6 | 24 | 2.38E+03 | 1.89E-01 | 2.24E+03 | 6.96E-03 | 2230 | 2230 | 2225 |
| 7 | 24 | 1.63E+03 | 1.86E-01 | 2.22E+03 | 6.53E-04 | 2215 | 2216 | 2210 |
| 8 | 24 | 2.38E+03 | 1.89E-01 | 2.23E+03 | 5.15E-03 | 2225 | 2229 | 2223 |
| 9 | 24 | 1.97E+03 | 1.66E-02 | 2.22E+03 | 1.55E-03 | 2217 | 2219 | 2213 |
| 10 | 24 | 2.12E+03 | 6.03E-02 | 2.21E+03 | 3.86E-03 | 2206 | 2208 | 2203 |
| Average | 24 | 2.01E+03 | 1.21E-01 | 2.22E+03 | 5.31E-03 | | | |
| Variance | 0 | 9.10E+04 | 6.55E-03 | 2.04E+02 | 1.03E-05 | | | |

Table 3: Dynamics of of one of the runs of the enhanced Algorithm for the random 3-SAT with matrix $A = (75 \times 305)$.

| $t$ | $\widehat{\lvert\mathcal{X}^*\rvert}$ | $\widehat{\lvert\mathcal{X}^*_{cap}\rvert}$ | $N_t$ | $N_t^{(s)}$ | $m_t^*$ | $m_{*t}$ | $\rho_t$ |
|---|---|---|---|---|---|---|---|
| 1 | 4.62E+21 | – | 1223 | 1223 | 285 | 274 | 0.122 |
| 2 | 6.88E+20 | – | 1490 | 1490 | 288 | 279 | 0.149 |
| 3 | 7.52E+19 | – | 1093 | 1093 | 291 | 283 | 0.109 |
| 4 | 8.62E+18 | – | 1146 | 1146 | 292 | 286 | 0.115 |
| 5 | 1.57E+18 | – | 1817 | 1817 | 293 | 288 | 0.182 |
| 6 | 2.33E+17 | – | 1489 | 1489 | 296 | 290 | 0.149 |
| 7 | 2.46E+16 | – | 1053 | 1053 | 296 | 292 | 0.105 |
| 8 | 7.34E+15 | – | 2987 | 2987 | 297 | 293 | 0.299 |
| 9 | 1.93E+15 | – | 2635 | 2635 | 298 | 294 | 0.264 |
| 10 | 4.74E+14 | – | 2454 | 2454 | 300 | 295 | 0.245 |
| 11 | 1.07E+14 | – | 2251 | 2251 | 299 | 296 | 0.225 |
| 12 | 2.09E+13 | – | 1960 | 1960 | 300 | 297 | 0.196 |
| 13 | 3.65E+12 | – | 1742 | 1742 | 302 | 298 | 0.174 |
| 14 | 5.66E+11 | – | 1551 | 1551 | 302 | 299 | 0.155 |
| 15 | 7.22E+10 | – | 1276 | 1276 | 303 | 300 | 0.128 |
| 16 | 8.34E+09 | – | 1155 | 1155 | 304 | 301 | 0.116 |
| 17 | 7.64E+08 | – | 917 | 917 | 304 | 302 | 0.092 |
| 18 | 5.10E+07 | – | 667 | 667 | 304 | 303 | 0.067 |
| 19 | 2.10E+06 | – | 412 | 412 | 305 | 304 | 0.041 |
| 20 | 3.28E+04 | – | 156 | 156 | 305 | 305 | 0.016 |
| 21 | 3.38E+04 | 3.21e+004 | 10000 | 8484 | 305 | 305 | 1.000 |

Here we used the following notations

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

2. $m_t^*$ and $m_{*t}$ denote the upper and the lower elite levels reached, respectively.

3. $\rho_t = N_t/N$ denote the adaptive rarity parameter.

## 4.2 Decision Making

Recall that the goal of the decision making algorithm is to decide whether or not the set $\mathcal{X}$ of the integer program constraints (10) has a feasible solution. The decision making algorithm presents a simple modification the direct Algorithm 6.3. In particular:

1. Instead of counting according to (22), that is

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

   we make decision in the sense that we need to decide whether $|\widehat{\mathcal{X}}^*_{dir}| > 0$ or $|\widehat{\mathcal{X}}^*_{dir}| = 0$.

2. Apply *global* $m_t$ policy from instead of the standard splitting step (see Step 3. in Algorithm 6.3.

We call such modified Algorithm 6.3 as the *decision making* Algorithm.

We run the decision making Algorithm 6.3 for different SAT problems taken from www.satlib.org using the *global* $m_t$ policy. In particular we took 40 instances out of more than 200 instances available, each presenting a random 3-SAT with the instance matrix $\boldsymbol{A}$ of size $(250 \times 1065)$ and with $|\widehat{\mathcal{X}}^*_{dir}| > 1$. We set the burn in parameter $b = 10$, $N = 10,000$ and $\rho = 0.5$. The CPU time was about 10 minutes. We always obtained that $|\widehat{\mathcal{X}}^*_{dir}| > 1$, that is we found that our algorithms works nicely. For $N = 1,000$ we found, however that our algorithm failed for some instances. The same was true for $N = 10,000$ and $\rho < 0.5$.

Table 4 presents the dynamics of one of such runs.

Table 4: Performance of the decision making Algorithm 6.3 with *global* $m_t$ policy for the random 3-SAT with the clause matrix $A = (250 \times 1065)$, $N = 10,000$, $\rho = 0.5$ and $b = 10$

| $t$ | $\widehat{\lvert\mathcal{X}^*\rvert}$ | $\lvert\widehat{\mathcal{X}}^*_{dir}\rvert$ | $N_t$ | $N_t^{(s)}$ | $m_t^*$ | $m_{*t}$ | $\rho_t$ |
|---|---|---|---|---|---|---|---|
| 1 | 8.76e+074 | 0 | 5161 | 4841 | 972 | 933 | 0.48 |
| 2 | 4.27e+074 | 0 | 7436 | 7079 | 1050 | 1023 | 0.49 |
| 3 | 1.84e+074 | 0 | 7288 | 6106 | 1058 | 1043 | 0.43 |
| 4 | 7.49e+073 | 0 | 6601 | 4970 | 1062 | 1051 | 0.41 |
| 5 | 2.85e+073 | 0 | 8318 | 5665 | 1062 | 1056 | 0.38 |
| 6 | 8.43e+072 | 0 | 6383 | 3353 | 1064 | 1059 | 0.30 |
| 7 | 2.48e+072 | 0 | 6745 | 2965 | 1065 | 1061 | 0.29 |
| 8 | 3.65e+071 | 0 | 6090 | 1745 | 1065 | 1063 | 0.15 |
| 9 | 5.90e+070 | 0 | 10470 | 1690 | 1065 | 1064 | 0.16 |
| 10 | 1.09e+070 | 1871 | 10140 | 1873 | 1065 | 1065 | 0.18 |
| 11 | 1.09e+070 | 11238 | 11238 | 11238 | 1065 | 1065 | 1.00 |

Table 5 presents the dynamics of a run with the same *global* $m_t$ policy Algorithm 6.3 for the random SAT with the instance matrix $A = (122 \times 663)$ and a single valid assignment, $(|\mathcal{X}^*| = 1)$, taken from http://www.is.titech.ac.jp/ watanabe/gensat. We set $N = 50,000$, $\rho = 0.95$ and $b = 10$. The results are self- explanatory. Note that

1. For $\rho < 0.95$ Algorithm 6.3 is stacked some where before 663.
2. The CPU time is about 5 hours.
3. For this difficult model with a single valid assignment we found that the *global* $m_t$ policy Algorithm 6.3 has approximately the same running time as the enhanced cloning Algorithm 6.2, for which we used the same $N = 50,000$, but $\rho = 0.1$ instead of $\rho = 0.95$.

Table 5: Performance of the *global* $m_t$ policy Algorithm 6.3 for the random $3-4$-SAT with the instance matrix $A = (122 \times 663)$, $N = 50,000$, $\rho = 0.95$ and $b = 10$

| $t$ | $\widehat{\lvert\mathcal{X}^*\rvert}$ | $\lvert\widehat{\mathcal{X}}^*_{dir}\rvert$ | $N_t$ | $N_t^{(s)}$ | $m_t^*$ | $m_{*t}$ | $\rho_t$ |
|---|---|---|---|---|---|---|---|
| 1 | 5.03e+036 | 0 | 28777 | 28412 | 627 | 585 | 0.96 |
| 2 | 4.77e+036 | 0 | 54792 | 53926 | 650 | 622 | 0.96 |
| 3 | 4.43e+036 | 0 | 52042 | 50652 | 653 | 634 | 0.97 |
| 4 | 4.18e+036 | 0 | 49586 | 48082 | 655 | 639 | 0.98 |
| 5 | 3.90e+036 | 0 | 47238 | 45392 | 656 | 642 | 0.98 |
| 6 | 3.63e+036 | 0 | 44824 | 42724 | 657 | 644 | 0.99 |
| 7 | 3.15e+036 | 0 | 41357 | 37781 | 658 | 646 | 0.97 |
| 8 | 2.85e+036 | 0 | 37781 | 34972 | 658 | 647 | 1.00 |
| 9 | 2.62e+036 | 0 | 34972 | 31710 | 658 | 648 | 1.00 |
| 10 | 2.20e+036 | 0 | 31710 | 27547 | 658 | 649 | 1.00 |
| 11 | 2.03e+036 | 0 | 55094 | 45921 | 660 | 650 | 1.00 |
| 12 | 1.44e+036 | 0 | 45921 | 34996 | 660 | 651 | 1.00 |
| 13 | 5.36e+035 | 0 | 34996 | 23854 | 660 | 652 | 1.00 |
| 14 | 4.20e+035 | 0 | 47708 | 28573 | 660 | 653 | 1.00 |
| 15 | 4.19e+035 | 0 | 57146 | 28378 | 660 | 654 | 1.00 |
| 16 | 3.43e+035 | 0 | 56756 | 22473 | 661 | 655 | 1.00 |
| 17 | 1.07e+035 | 0 | 44946 | 13879 | 661 | 656 | 1.00 |
| 18 | 1.66e+032 | 0 | 41637 | 7867 | 661 | 657 | 1.00 |
| 19 | 1.08e+031 | 0 | 31468 | 3390 | 661 | 658 | 1.00 |
| 20 | 9.72e+030 | 0 | 30510 | 1126 | 663 | 659 | 1.00 |
| 21 | 6.98e+024 | 0 | 30402 | 204 | 663 | 660 | 1.00 |
| 22 | 1.99e+017 | 0 | 30600 | 3 | 663 | 661 | 1.00 |
| 23 | 1.10e+010 | 0 | 30600 | 1 | 663 | 662 | 1.00 |
| 24 | 1.10e+010 | 1 | 30600 | 1 | 663 | 663 | 1.00 |

# 5 Concluding Remarks

We showed how a simple modification of the splitting method based on Gibbs sampler can be efficiently used for decision making in the sense that one can efficiently decide whether or not a given set of integer program constraints has at least one feasible solution. Our decision making is based on what is called the *local* $m_t$ and *global* $m_t$ modification of the direct splitting Algorithm 6.3. We also show how to incorporate the classic *capture-recapture* method into the direct Algorithm 6.3 in order to obtain a low variance estimator for the counting quantity representing, say the number feasible solution on the set defined as by the constraints of an integer program. We finally present numerical with with both, the decision making and the capture-recapture estimators and show their superiority as compared to the conventional product one $\widehat{|\mathcal{X}^*|}$, while solving quite general decision making and counting ones, like the satisfiability problems.

# 6 Appendix: Splitting Algorithms

Below, following [15], we present two versions of the splitting algorithm: the so-called *basic* version and the *enhanced* version having in mind Example 1.1.

## 6.1 Basic Splitting Algorithm

Let $N$, $\rho_t$ and $N_t$ be the fixed sample size, the adaptive rarity parameter and the number of elite samples at iteration $t$, respectively (see [15] details). Recall [15] that the elite sample $\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_t}$ corresponds to the largest subset of the population $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{N_t}\}$, for which $S(\boldsymbol{X}_i) \geq \widehat{m}_t$, that is $\widehat{m}_t$ is the $(1-\rho_t)$ sample quantile of of the ordered statistics values of $S(\boldsymbol{X}_1), \ldots, S(\boldsymbol{X}_N)$. It follows that the number of elites $N_t = \lceil N\rho_t \rceil$, where $\lceil \cdot \rceil$ denotes rounding to the largest integer.

In the basic version at iteration $t$ we *split* each elite sample $\eta_t = \left\lceil \rho_t^{-1} \right\rceil$ times. By doing so we generate $\left\lceil \rho_t^{-1} N_t \right\rceil \approx N$ new samples for the next iteration $t+1$. The rationale is based on the fact that if all $\rho_t$ are not small, say $\rho_t \geq 0.01$, we have at each iteration $t$ enough *stationary* elite samples, and all what the Gibbs sampler has to do for the next iteration is to generate $N \approx \left\lceil \rho_t^{-1} N_t \right\rceil$ *new* samples uniformly distributed on $\mathcal{X}_{t+1}$.

**Algorithm 6.1 (Basic Splitting Algorithm for Counting)** Given the initial parameter $\rho_0$, say $\rho_0 \in (0.01, 0.25)$ and the sample size $N$, say $N = nm$, execute the following steps:

1. **Acceptance-Rejection** Set a counter $t = 1$. Generate a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ uniformly on $\mathcal{X}_0$. Let $\widehat{\mathcal{X}}_0 = \{\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_0}\}$ be the elite samples. Take

$$\widehat{c}_0 = \widehat{\ell}(\widehat{m}_0) = \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \widehat{m}_0\}} = \frac{N_0}{N} \tag{19}$$

   as an *unbiased* estimator of $c_0$. Note that $\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_0} \sim g^*(\boldsymbol{x}, \widehat{m}_0)$, where $g^*(\boldsymbol{x}, \widehat{m}_0)$ is a *uniform distribution* on the set $\mathcal{X}_1 = \{\boldsymbol{x} : S(\boldsymbol{x}) \geq \widehat{m}_0\}$.

2. **Splitting** Let $\widehat{\mathcal{X}}_{t-1} = \{\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$ be the elite sample at iteration $(t-1)$, that is the subset of the population $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ for which $S(\boldsymbol{X}_i) \geq \widehat{m}_{t-1}$. Reproduce $\eta_{t-1} = \left\lceil \rho_{t-1}^{-1} \right\rceil$ times each vector $\widehat{\boldsymbol{X}}_k = (\widehat{X}_{1k}, \ldots, \widehat{X}_{nk})$ of the elite sample $\{\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$, that is take $\eta_{t-1}$ identical copies of each vector $\widehat{\boldsymbol{X}}_k$. Denote the entire new population ($\eta_{t-1} N_{t-1}$ cloned vectors plus the original 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 random Gibbs sampler) for a single period (single burn-in). Denote the *new entire* population by $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$. Note that each vector in the sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ is distributed $g^*(\boldsymbol{x}, \widehat{m}_{t-1})$, where $g^*(\boldsymbol{x}, \widehat{m}_{t-1})$ has *approximately* a uniform distribution on the set $\mathcal{X}_t = \{\boldsymbol{x} : S(\boldsymbol{x}) \geq \widehat{m}_{t-1}\}$.

3. **Estimating** $c_t$ Take $\widehat{c}_t = \frac{N_t}{N}$ (see (9)) as an estimator of $c_t$ in (7). Note again that each vector of $\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_t}$ of the elite sample is distributed $g^*(\boldsymbol{x}, \widehat{m}_t)$, where $g^*(\boldsymbol{x}, \widehat{m}_t)$ has approximately a uniform distribution on the set $\mathcal{X}_{t+1} = \{\boldsymbol{x} : S(\boldsymbol{x}) \geq \widehat{m}_t\}$.

4. **Stopping rule** If $m_t = m$ go to step 5, otherwise set $t = t+1$ and repeat from step 2.

5. **Final Estimator** Deliver $\widehat{\ell}$ given in (8) as an estimator of $\ell$ and $|\widehat{\mathcal{X}}^*| = \widehat{\ell}|\mathcal{X}|$ as an estimator of $|\mathcal{X}^*|$.

Note that at iteration $t$ Algorithm 6.1 *splits* each elite sample $\eta_t = \lceil \rho_t^{-1} \rceil$ times. By doing it generates $\lceil \rho_t^{-1} N_t \rceil \approx N$ new samples for the next iteration $t+1$. The rationale is based on the fact that if all $\rho_t$ are not small, say $\rho_t \geq 0.01$, we have at each iteration $t$ enough *stationary* elite samples, and all what the Gibbs sampler has to do for the next iteration is to generate $N \approx \lceil \rho_t^{-1} N_t \rceil$ *new* samples uniformly distributed on $\mathcal{X}_{t+1}$.

Figure 1 presents a typical dynamics of the splitting algorithm, which terminates after two iterations. The set of points denoted $\star$ and $\bullet$ is associated with these two iterations. In particular the points marked by $\star$ are uniformly distributed on the sets $\mathcal{X}_0$ and $\mathcal{X}_1$. (Those, which are in $\mathcal{X}_1$ correspond to the elite samples). The points marked by $\bullet$ are approximately uniformly distributed on the sets $\mathcal{X}_1$ and $\mathcal{X}_2$. (Those, which are in $\mathcal{X}_2 = \mathcal{X}^*$ likewise correspond to the elite samples).

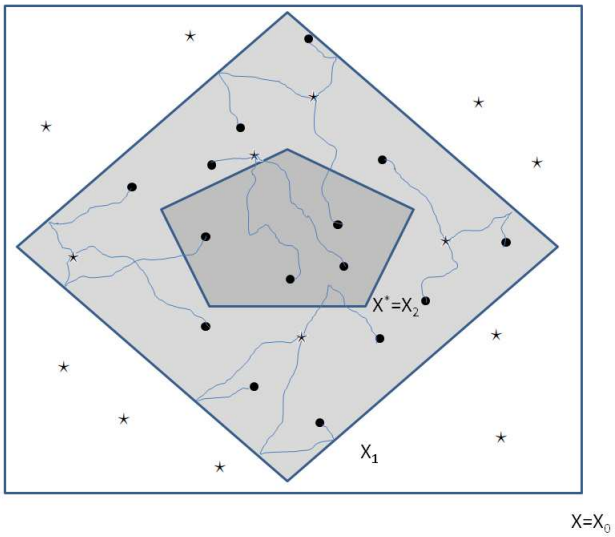


Figure 1: Dynamics of Algorithm 6.1

## 6.2 Enhanced Splitting Algorithm for Counting

Here we introduce an enhanced version of the basic splitting Algorithm 6.1, which contains (i) an enhanced splitting (splitting) step instead of the original one as in Algorithms 6.1 and a (ii) new screening step.

(i) **Enhanced cloning step** Denote be $\eta_t$ the number of times each of the $N_t$ elite samples is reproduced at iteration $t$, and call it the *cloning (splitting) parameter*. Denote by $b_t$ the *burn-in parameter*, that is the number of times each elite sample has to follow through the MCMC (Gibbs) sampler. The purpose of enhanced cloning step is to find a good balance, in terms of bias-variance of the estimator of $|\mathcal{X}^*|$, between $\eta_t$ and $b_t$, provided the number of samples $N$ is given.

Let us assume for a moment that $b_t = b$ is fixed. Then for fixed $N$, we can define the adaptive *cloning* parameter $\eta_{t-1}$ at iteration $t-1$ as follows

$$\eta_{t-1} = \left\lceil \frac{N}{b N_{t-1}} \right\rceil - 1 = \left\lceil \frac{N_{cl}}{N_{t-1}} \right\rceil - 1. \tag{20}$$

Here $N_{cl} = N/b$ is called the *cloned sample size*, and as before $N_{t-1} = \rho_{t-1}N$ denotes the number of elites and $\rho_{t-1}$ is the adaptive rarety parameter at iteration $t-1$ [see [17] for details).

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 ), respectively.

As an alternative to (20) one can use the following heuristic strategy in defining $b$ and $\eta$: find $b_{t-1}$ and $\eta_{t-1}$ 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

$$b_{t-1} \approx \eta_{t-1} \approx \left(\frac{N}{N_{t-1}}\right)^{1/2}. \tag{21}$$

Consider again the same two cases for $N_{t-1}$ and $N$ We have $b_{t-1} \approx \eta_{t-1} = 7$ and $b_{t-1} \approx \eta_{t-1} = 3$, respectively. We found numerically that both versions work well, but unless stated otherwise we shall use (21).

(ii) **Screening step**. Since the IS pdf $g^*(\boldsymbol{x}, m_t)$ must be *uniformly distributed* for each fixed $m_t$, the splitting algorithm checks at each iteration whether or not *all elite vectors* $\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_t}$ *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*. Note that this procedure prevents (at least partially) the empirical pdf associated with $\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_t}$ from deviating from the uniform.

**Algorithm 6.2 (Enhanced Splitting Algorithm for Counting)** Given the parameter $\rho$, say $\rho \in (0.01, 0.25)$ and the sample size $N$, say $N = nm$, execute the following steps:

1. **Acceptance-Rejection** - the same as in Algorithm 6.1.

2. **Screening** Denote the elite sample obtained at iteration $(t-1)$ by $\{\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$. Screen out the redundant elements from the subset $\{\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$, and denote the resulting (reduced) one as $\{\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$.

3. **Splitting (Cloning)** Given the size $N_{t-1}$ of the screened elites $\{\widehat{\boldsymbol{X}}_1, \ldots, \widehat{\boldsymbol{X}}_{N_{t-1}}\}$ at iteration $(t-1)$, find the splitting and the burn-in parameters $\eta_{t-1}$ and $b_{t-1}$ according to (21). Reproduce $\eta_{t-1}$ 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 $\eta_{t-1}$ 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 $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$. Note that each vector in the sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ is distributed approximately $g^*(\boldsymbol{x}, \widehat{m}_{t-1})$, where $g^*(\boldsymbol{x}, \widehat{m}_{t-1})$ is a uniform distribution on the set $\mathcal{X}_t = \{\boldsymbol{x} : S(\boldsymbol{x}) \geq \widehat{m}_{t-1}\}$.

4. **Estimating** $c_t$ - the same as in Algorithm 6.1.

5. **Stopping rule** -the same as in Algorithm 6.1.

6. **Final estimator** - the same as in Algorithm 6.1.

Note that the basic Algorithm 6.1 (with $b = 1$ and without screening) presents a particular case of the enhanced Algorithm 6.2.

### 6.3 Direct Splitting Algorithm

The direct estimator below can be viewed as an alternative to the estimator $|\widehat{\mathcal{X}}^*|$ obtained by Algorithm 6.1. This estimator is based on *direct counting* of the number of screened samples obtained immediately after crossing the level $m$. Such a counting estimator, denoted by $|\widehat{\mathcal{X}}^*_{dir}|$, is associated with the *empirical* distribution of the uniform distribution $g^*(\boldsymbol{x}, m)$. We found numerically that $|\widehat{\mathcal{X}}^*_{dir}|$ is extremely useful and very accurate. Note that it is applicable only for counting problems with $|\mathcal{X}^*|$ not too large. Specifically $|\mathcal{X}^*|$ 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 the most difficult ones and in many counting problems one is interested in the cases where $|\mathcal{X}^*|$ does not exceed some fixed quantity, say $\mathcal{N}$. Clearly, this is possible only if $N \geq \mathcal{N}$. It is important to note that $|\widehat{\mathcal{X}}^*_{dir}|$ is typically much more accurate than its counterpart, the standard estimator $|\widehat{\mathcal{X}}^*| = \widehat{\ell}|\mathcal{X}|$. The reason is that $|\widehat{\mathcal{X}}^*_{dir}|$ is obtained *directly* by counting all distinct values of $\boldsymbol{X}_i,\ i = 1, \ldots, N$ satisfying $S(\boldsymbol{X}_i) \geq m$, that is it can be written as

$$\widehat{|\mathcal{X}^*_{dir}|} = \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i^{(d)}) \geq m\}}, \tag{22}$$

where $\boldsymbol{X}_i^{(d)} = \boldsymbol{X}_i$, if $\boldsymbol{X}_i \neq \boldsymbol{X}_j,\ \forall j = 1, \ldots, i-1$ and $\boldsymbol{X}_i^{(d)} = 0$, otherwise. Note that we set in advance $\boldsymbol{X}_1^{(d)} = \boldsymbol{X}_1$. Note also that there is no need here to calculate $\widehat{c}_t$ at any step.

**Algorithm 6.3 ( Direct Algorithm for Counting)** Given the rarity parameter $\rho$, 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 $N$, execute the following steps:

1. **Acceptance-Rejection** - same as in Algorithm 6.2.
2. **Screening** - same as in Algorithm 6.2.
3. **Splitting** - same as in Algorithm 6.2.
4. **Stopping rule** - same as in Algorithm 6.2.
5. **Final Estimator** For $m_T = m$, take a sample of size $N$, and deliver $|\widehat{\mathcal{X}}^*_{dir}|$ in (22) as an estimator of $|\mathcal{X}^*|$.

Note that the counting Algorithm 6.3 can be readily modified for combinatorial optimization, since an optimization problem can be can be viewed as a particular case of counting, where the counting quantity $|\mathcal{X}^*| = 1$.

# References


[1] Asmussen S. and P.W. Glynn, *Stochastic Simulation: Algorithms and Analyses*, Springer, 2007.

[2] S. Baumert, A. Ghate, S. Kiatsupaibul, Y. Shen, R. L. Smith and Z. B. Zabinsky, "Discrete Hit-and-Run for Sampling Points from Arbitrary Distributions over Subsets of Integer Hyper-rectangles," Operations Research, 2009.

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

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

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

[6] A. Lagnoux-Renaudie *A two-steps branching splitting model under cost constraint*. to be published in Journal of Applied Probability.

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

[8] Gryazina E. and B. Polyak "Randomized Methods Based on new Monte Carlo Schemes for Control and Optimizations" Annals of Operations Research, 2009.

[9] P. L'Ecuyer, V. Demers, and B. Tuffin, "Rare-Events, Cloning, and Quasi-Monte Carlo", ACM Transactions on Modeling and Computer Simulation, 17, 2, 2007.

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

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

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

[13] Rubinstein, R. Y. " The Cross-Entropy Method for Combinatorial and Continuous Optimization " *Methodology and Computing in Applied Probability*, 1, 127-190, 1999.

[14] R. Y. Rubinstein. "The Gibbs Cloner for Combinatorial Optimization, Counting and Sampling", To be published in *Methodology and Computing in Applied Probability*, 2008.

[15] R. Y. Rubinstein. "Randomized Algorithms with Splitting: Why the Classic Randomized Algorithms do not Work and how to Make them Work" *Methodology and Computing in Applied Probability*, 2009.

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

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

[18] R. L. Smith, "Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions", Operations Research, vol. 32, pp. 1296–1308, 1984.