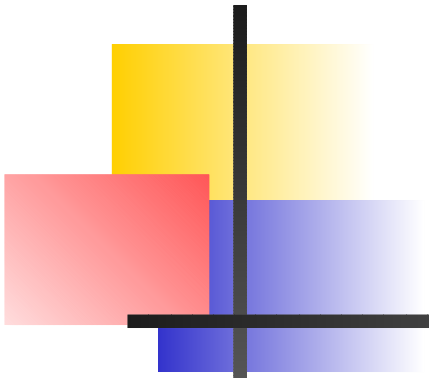
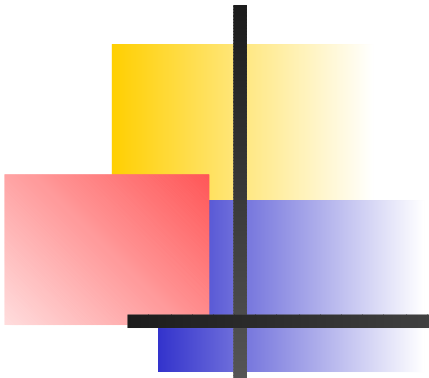


Monte-Carlo Algorithms with Splitting: How to Improve the Classic Randomized Algorithms

Radislav Vaisman

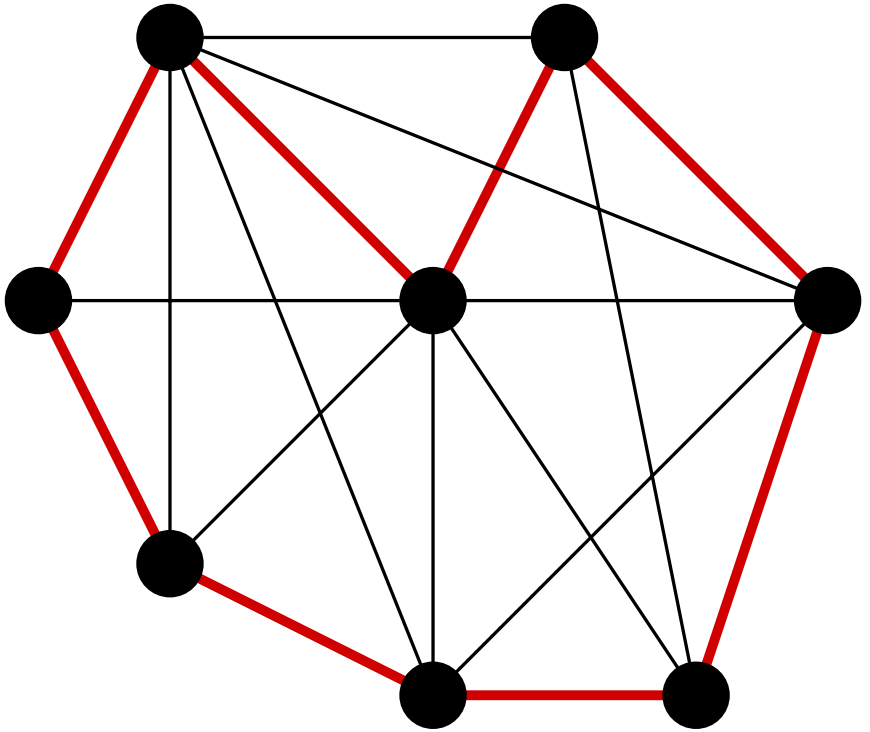
Faculty of Computer Science, The Open University of Israel



Contents

1. Introduction: Approximate Counting.
2. Motivating Examples and the Product Estimator.
3. The Classic Randomized Algorithms for Counting.
4. The Gibbs Sampler and Randomized Algorithms.
5. The Splitting Method as a Natural Extension of the Classic Randomized Algorithms.
6. Applications: Integer Programs, the Satisfiability Problem, Vertex Coloring, etc.
7. Convergence and Numerical Results.
8. Conclusions and futher research.

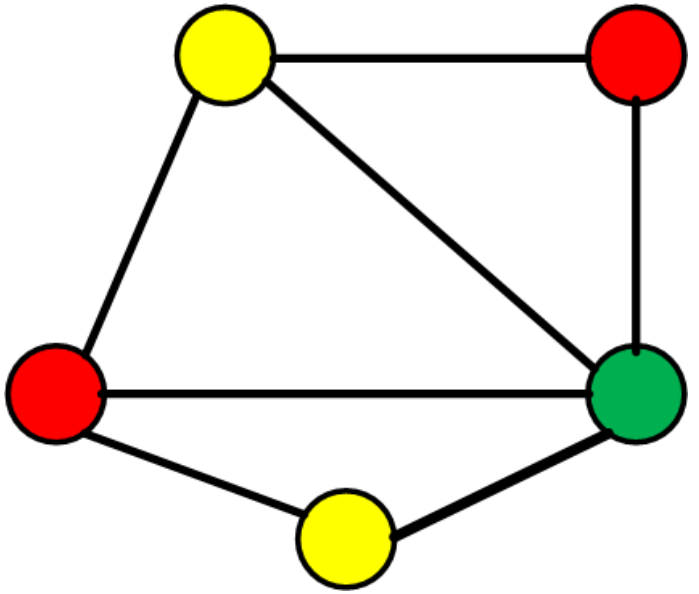
Counting Hamiltonian Cycles



How many Hamiltonian cycles does this graph have?

Vertex Coloring

Given a graph $G = (V, E)$ with m edges and n vertices, color the vertices of V with given q colors (say 2 colors), such that for each edge $(i, j) \in E$, vertices i and j have different colors.



Given q colors, how many different Vertex Coloring this graph has?

Counting Independent Sets

Consider a graph $G = (V, E)$ with m edges and n vertices. A node set is called *independent* if no two nodes are connected by an edge, that is, if no two nodes are adjacent; see Figure 1 for an illustration of this concept.

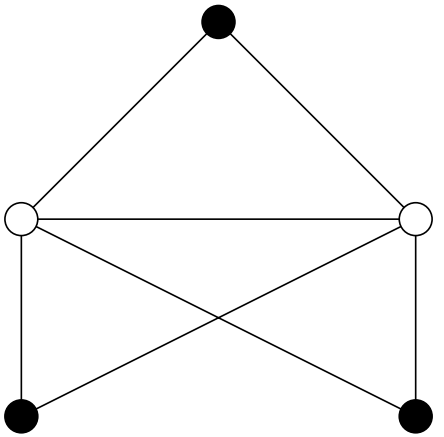


Figure 1: The black nodes form an independent set since they are not adjacent to each other.

General Case: Integer Constraints

Consider a set containing both equality and inequality constraints of an integer program, that is

$$\sum_{k=1}^{n} a_{ik} x_k = b_i, \ i = 1, \ldots, m_1,$$

$$\sum_{k=1}^{n} a_{jk} x_k \geq b_j, \ j = m_1 + 1, \ldots, m_1 + m_2,$$

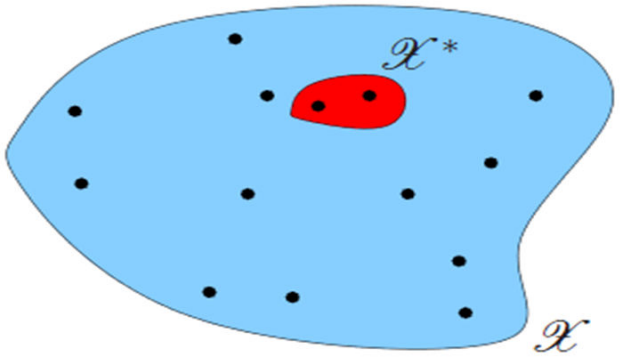
$$\boldsymbol{x} \geq \mathbf{0}, \quad x_k \text{ integer } \forall k = 1, \ldots, n.$$

Counting via Monte Carlo

We start with the following basic

Example.

Assume we want to calculate an area of some “irregular" region $\mathcal{X}^*$. The Monte-Carlo method suggests inserting the ”irregular" region $\mathcal{X}^*$ into a nice “regular" one $\mathcal{X}$ as per figure below



$\mathcal{X}$: Set of objects (paths in a graph, colorings of a graph, etc.)

$\mathcal{X}^*$: Subset of special objects (cycles in a graph, colorings of a certain type, etc).

Counting via Monte Carlo

To calculate $|\mathcal{X}^*|$ we apply the following sampling procedure:
(i) Generate a random sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$, *uniformly* distributed over the "regular" region $\mathcal{X}$.
(ii) Estimate the desired area $|\mathcal{X}^*|$ as

$$\widehat{|\mathcal{X}^*|} = \widehat{\ell}|\mathcal{X}|,$$

where

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

$I_{\{\boldsymbol{X}_k \in \mathcal{X}^*\}}$ denotes the indicator of the event $\{\boldsymbol{X}_k \in \mathcal{X}^*\}$ and $\{\boldsymbol{X}_k\}$ is a sample from $f(\boldsymbol{x})$ over $\mathcal{X}$, where $f(\boldsymbol{x}) = \frac{1}{|\mathcal{X}|}$.

The Randomized Algorithm

Estimating $|\mathcal{X}^*|$ with a known $|\mathcal{X}_0| = |\mathcal{X}|$.

1. Define a sequence of sets $\mathcal{X}_1, \ldots, \mathcal{X}_m$ and write $|\mathcal{X}^*|$ as

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

   Note that the ratio $\frac{|\mathcal{X}^*|}{|\mathcal{X}_0|}$ is very small, like $= 10^{-100}$, while each ratio $c_t = \frac{|\mathcal{X}_t|}{|\mathcal{X}_{t-1}|}$ is not, like $c_t = 10^{-2}$ or greater and $\mathcal{X}_0 \supset \mathcal{X}_1 \supset \cdots \supset \mathcal{X}_m = \mathcal{X}^*$.

2. Develop an efficient estimator for each c_t and deliver

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

Vertex Coloring

Given a graph $G = (V, E)$ with m edges and n vertices, color the vertices of V with given q colors, such that for each edge $(i, j) \in E$, vertices i and j have different colors. Here, as before, we consider an arbitrary ordering of the edges. Let E_j be the set of the first j edges and let $G_j = (V, E_j)$ be the associated sub-graph. Note that $G_m = G$, and that G_{j+1} is obtained from G_j by adding the edge e_{j+1}. Here $|\mathcal{X}_0| = q^n$, since G_0 has no edges.

The Rare-Event Approach

It is often more convenient to cast the problem of estimating $|\mathscr{X}^*|$ into the problem of estimating the rare event probability

$$\ell(m) = \frac{|\mathscr{X}^*|}{|\mathscr{X}|},$$

which can be also written as

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

Here $S(\boldsymbol{X})$ is the sample performance, like the length of a randomly selected Hamiltonian cycle, $\boldsymbol{X} \sim f(\boldsymbol{x})$, $f(\boldsymbol{x})$ is typically being uniformly distributed on the set of points of $\mathscr{X}$ and m is as before, a fixed parameter.

The Rare-Event Approach

We estimate $\ell(m)$ as

$$\ell(m) = c_0 \prod_{t=1}^{T} c_t, = \mathbb{E}_f \left[I_{\{S(\boldsymbol{X}) \geq m\}} \right]$$

where, as before $c_0 = \mathbb{E}_f[I_{\{S(\boldsymbol{X}) \geq m_0\}}]$,

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

$\{m_t,\ t = 0, 1, \ldots, T\}$ is a fixed grid satisfying $-\infty < m_0 < m_1 < \ldots m_T = m$ (typically $m_t = t$). Here $g^*_{t-1} = \mathcal{U}(\mathscr{X}_{t-1})$ and $\mathscr{X}_t = \{\boldsymbol{x} : S(\boldsymbol{x}) \geq m_t\}$.

The Rare-Event Approach

The final estimator of $\ell(m)$, based on the product of $c_t = \mathbb{E}_{g^*_{t-1}}[I_{\{S(\boldsymbol{X})\geq m_t\}}],\ t = 0, \ldots, T$ can be written as

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

where

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

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

The main trick is to show how to sample uniformly from the IS pdf $g^*(\boldsymbol{x}, m_{t-1})$ in the reduced space $\mathscr{X}_t$.

The Splitting Approach

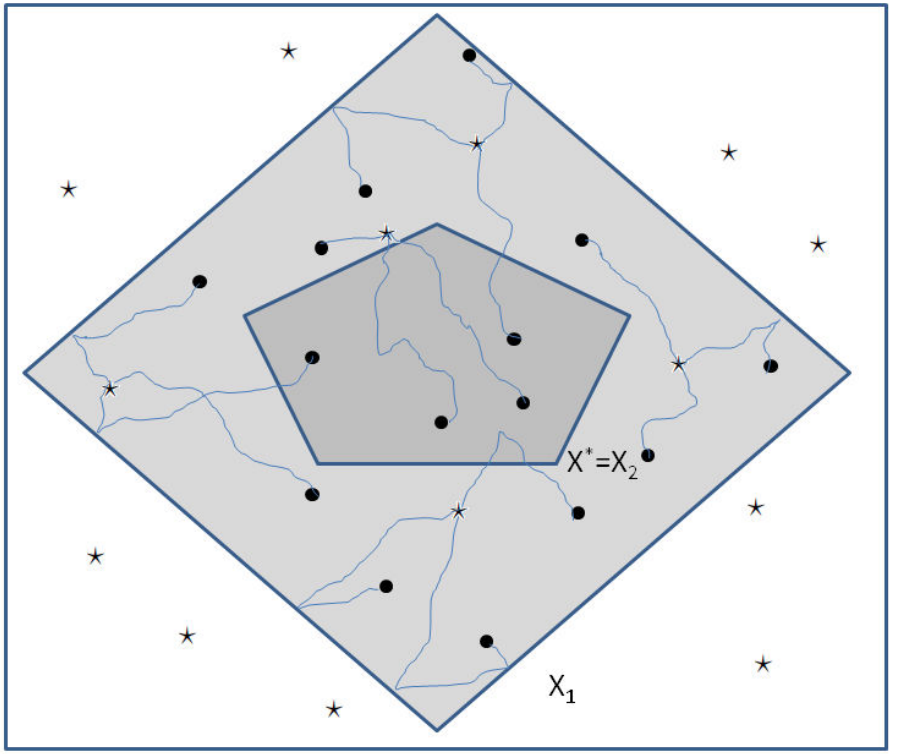
To sample uniformly in the reduced space $\mathscr{X}_t$ we shall use a combination of the Gibbs sampler with classic splitting method. It is based on runing multiple trajectories in parallel. The algorithm will be called the *splitting* or *cloning* algorithm.

Our *splitting* algorithm generates an adaptive sequence of *non-parametric* tuples

$$\{(m_0, f(\boldsymbol{x}, \boldsymbol{v}_0)),\ (m_1, g^*(\boldsymbol{x}, m_0)), \ldots, (m_T, g^*(\boldsymbol{x}, m_{T-1}))\}$$

Typical Dynamics with Two Iterations

Points denoted as $\star$ are uniformly distributed on the sets $\mathcal{X}_0$ and $\mathcal{X}_1$.
Points denoted as $\bullet$ are approximately uniformly distributed on the sets $\mathcal{X}_1$ and $\mathcal{X}_2$.



General Case: Integer Constraints

Consider a set containing both equality and inequality constraints of an integer program, that is

$$\sum_{k=1}^{n} a_{ik} x_k = b_i, \;\; i = 1, \ldots, m_1,$$

$$\sum_{k=1}^{n} a_{jk} x_k \geq b_j, \;\; j = m_1 + 1, \ldots, m_1 + m_2,$$

$$\boldsymbol{x} \geq \mathbf{0}, \quad x_k \text{ integer } \forall k = 1, \ldots, n.$$

General Case: Integer Constraints

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

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

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

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

while the remaining m_2 ones are

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

General Case: Integer Constraints

Thus, in order to count the the number of feasible solution on the above set we shall consider an associated rare event probability estimation problem involving a *sum of dependent Bernoulli random variables*. Such representation is crucial for a large set of counting problems.

General Procedure

As mentioned we cast the original counting problem into an associated rare-events probability estimation problem. The estimation of

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

involves the following iterative steps:

A General Randomized Algorithm

1 **Starting:** Start with the proposal pdf $f(x)$, which is uniformly distributed on the sample space $\mathscr{X}$. Set $t := 1$.

A General Randomized Algorithm

1 **Starting:** Start with the proposal pdf $f(x)$, which is uniformly distributed on the sample space $\mathscr{X}$. Set $t := 1$.

2 **Update $\hat{m}_t$:** Draw $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from the uniform pdf $g_t = g(\boldsymbol{x}, \widehat{m}_t) = \mathcal{U}(\mathscr{X}_t)$. Find the elite sampling based on $\hat{m}_t$, which is the worst performance of the $\rho \times 100\%$ best performances. Estimate c_t as $\widehat{c}_t = N_t/N$.

A General Randomized Algorithm

1 **Starting:** Start with the proposal pdf $f(x)$, which is uniformly distributed on the sample space $\mathscr{X}$. Set $t := 1$.

2 **Update $\hat{m}_t$:** Draw $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from the uniform pdf $g_t = g(\boldsymbol{x}, \widehat{m}_t) = \mathcal{U}(\mathscr{X}_t)$. Find the elite sampling based on $\hat{m}_t$, which is the worst performance of the $\rho \times 100\%$ best performances. Estimate c_t as $\widehat{c}_t = N_t/N$.

3 **Split the elite sample and update $g_t = \mathcal{U}(\mathscr{X}_t)$ and $\mathscr{X}_t$:** Deliver $g_{t+1} = \mathcal{U}(\mathscr{X}_{t+1})$ and $\mathscr{X}_{t+1}$ and increase t by 1.

A General Randomized Algorithm

1. **Starting:** Start with the proposal pdf $f(x)$, which is uniformly distributed on the sample space $\mathscr{X}$. Set $t := 1$.
2. **Update $\hat{m}_t$:** Draw $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from the uniform pdf $g_t = g(\boldsymbol{x}, \widehat{m}_t) = \mathcal{U}(\mathscr{X}_t)$. Find the elite sampling based on $\hat{m}_t$, which is the worst performance of the $\rho \times 100\%$ best performances. Estimate c_t as $\widehat{c}_t = N_t/N$.
3. **Split the elite sample and update $g_t = \mathcal{U}(\mathscr{X}_t)$ and $\mathscr{X}_t$:** Deliver $g_{t+1} = \mathcal{U}(\mathscr{X}_{t+1})$ and $\mathscr{X}_{t+1}$ and increase t by 1.
4. **Stopping:** If the stopping criterion is met, then stop; otherwise set $t := t + 1$ and reiterate from step 2.

The Gibbs Sampler

We need to sample from any pdf $g^*(\boldsymbol{x})$ or any other pdf $g(\boldsymbol{x})$. It is assumed that generating from the conditional pdfs $g(X_i|X_1,\ldots,X_{i-1},X_{i+1},\ldots,X_n),\ i=1,\ldots,n$ is simple.

In Gibbs sampler for any given vector $\boldsymbol{X}=(X_1,\ldots,X_n)\in\mathcal{X}$ one generates a *new* vector $\widetilde{\boldsymbol{X}}=(\widetilde{X}_1,\ldots,\widetilde{X}_n)$ as:

Algorithm: The Systematic Gibbs Sampler

1. *Draw* $\widetilde{X}_1$ *from the conditional pdf* $g(X_1|X_2,\ldots,X_n)$.
2. *Draw* $\widetilde{X}_i$ *from the conditional pdf* $g(X_i|\widetilde{X}_1,\ldots,\widetilde{X}_{i-1},X_{i+1},\ldots,X_n),\ i=2,\ldots,n-1.$
3. *Draw* $\widetilde{X}_n$ *from the conditional pdf* $g(X_n|\widetilde{X}_1,\ldots,\widetilde{X}_{n-1})$.

The Gibbs Sampler: Bernoulli Example

Consider estimation

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

The Gibbs sampler for generating variables $X_i,\ i = 1, \ldots, N$ is

$$g^*(x_i, m | \boldsymbol{x}_{-i}) = c_i(m) f_i(x_i) I_{\{x_i \geq m - \sum_{j \neq i} x_j\}},$$

where $|\boldsymbol{x}_{-i}$ denotes conditioning on all random variables but *excluding* the remaining ones and $c_i(m)$ is the normalization constant. Sampling a random variable $\widetilde{X}_i$ can be performed as follows. Generate $Y \sim$ Ber $(1/2)$. If $I_{\{\widetilde{X} \geq m - \sum_{j \neq i} x_j\}}$, then set $\widetilde{X}_i = Y$, otherwise set set

Algorithm versions

(i) Splitting of the sub-graphs $G_j,\ j = 1, \ldots, m$ with strictly fixed topology configurations and with levels $m_j,\ j = 1, \ldots, m$ strictly fixed in advance.

(ii) Splitting of the sub-graphs $G_j,\ j = 1, \ldots, m$ with strictly fixed topology configurations, but with adaptive levels $m_j,\ j = 1, \ldots, m$.

(iii) Splitting of the sub-graphs $G_j,\ j = 1, \ldots, m$ with adaptive choice of the topology configurations and with levels $m_j,\ j = 1, \ldots, m$ strictly fixed in advance.

(iv) Splitting of the sub-graphs $G_j,\ j = 1, \ldots, m$ with both adaptive choice of the topology configurations and of levels $m_j,\ j = 1, \ldots, m$.

We shall call the above four versions *Split1, Split2, Split3* and *cloning*, respectively.

Convergence of Split1

Assume for a moment that similar to RAN we use a long warm-up periods for each of the N parallel independent Markov chains in Split1. In this case the convergence theorems established for RAN (Hayes, Vera and Vigoda, 2007; Mitzenmacher and Upfal, 2005; Motwani and Raghavan, 1997) automatically hold for Split1. The reason is that Split1 extends RAN in the sense that in the former case we independently run in N parallel Markov chains instead of a single one. Clearly, by taking the best performance from N parallel simulations one can not do worse than when using a single one.

Numerical Results - SAT

Performance of splitting Algorithm (clone) for SAT 20×80 model.

Run N_0	Iterations	$\lvert\widetilde{\mathscr{X}}^*\rvert$	RE of $\lvert\widetilde{\mathscr{X}}^*\rvert$	$\lvert\hat{\mathscr{X}}^*_{dir}\rvert$	RE of $\lvert\hat{\mathscr{X}}^*_{dir}\rvert$	CPU
1	10	14.612	0.026	15	0.000	5.143
2	10	14.376	0.042	15	0.000	5.168
3	10	16.304	0.087	15	0.000	5.154
4	10	19.589	0.306	15	0.000	5.178
5	10	13.253	0.116	15	0.000	5.140
6	10	17.104	0.140	15	0.000	5.137
7	10	14.908	0.006	15	0.000	5.173
8	10	13.853	0.076	15	0.000	5.149
9	10	18.376	0.225	15	0.000	5.135
10	10	12.668	0.155	15	0.000	5.156
Average	10	15.504	[illegible]	[illegible]	[illegible]	[illegible]

Numerical Results - SAT

Comparative studies of Algorithms RAN, Split1, Split 2, Split 3 and cloning Algorithm for the product estimator $|\bar{\mathscr{X}}^*|$ and the direct estimator $|\bar{\mathscr{X}}^*|_{dir}$ on SAT 20×80 model.

Algorithm	$|\bar{\mathcal{X}}^*|$	$\bar{RE}$	$|\bar{\mathscr{X}}^*|_{dir}$	$\bar{RE}_{dir}$
RAN	95.606	5.470	8.4	0.440
Split1	13.608	0.159	15	0.000
Split2	16.155	0.130	15	0.000
Split3	14.025	0.127	15	0.000
Cloning	15.504	0.118	15	0.000

Numerical Results - more SAT models

- SAT 75×325 model was sucessefuly solved by Split3 and Clone. RAN is always stuck at some intermediate level. Split1 and Split2 are never stuck, but both are much slower than splitting Algorithm 3.1. In addition, both are trapped at some local extremum.
- We have solved a very hard instances too. Example: 3-SAT 122×663 having a single solution althought the running time was long.

Numerical Results - Coloring

We ran the vertex coloring models in the following two settings:
(i) Condition $q \geq 2\Delta + 1$ holds.
(ii) The condition is violated, that is, q is any integer number satisfying $q \geq 2$. We found that:

1. The RAN algorithm and its associates Split1 and Split2 still perform satisfactorily, as predicted by Theorem (Mitcenmacher and Upfal) for case (i), but fail for case (ii), in particular when q is small.
2. The splitting algorithms Split3 and the Cloning Algorithm perform nicely irrespective of the value of q. Note that case (ii), in particular when q is small, is the most interesting and most difficult, since $|\mathscr{X}^*|$ is very small relative to $|\mathscr{X}|$, thus $\ell = \frac{|\mathscr{X}^*|}{|\mathscr{X}|}$ is a very low probability.

Numerical Results - Coloring

Performance of cloning Algorithm for 4-coloring problem with $n = 40$ nodes.

Run N_0	Iterations	$\lvert\widetilde{\mathscr{X}}^*\rvert$	RE of $\lvert\widetilde{\mathscr{X}}^*\rvert$	$\lvert\hat{\mathscr{X}}^*_{dir}\rvert$	RE of $\lvert\hat{\mathscr{X}}^*_{dir}\rvert$	CPU
1	24	1415.62	0.055	1318	0.015	257.168
2	24	1194.38	0.110	1322	0.012	257.098
3	24	1356.09	0.010	1316	0.016	256.942
4	24	1596.61	0.190	1264	0.055	258.255
5	24	1348.93	0.005	1316	0.016	256.568
6	24	1627.90	0.213	1328	0.007	258.743
7	24	1353.55	0.009	1304	0.025	257.663
8	24	1293.32	0.036	1330	0.006	260.002
9	24	1229.90	0.084	1312	0.019	259.695
10	24	1590.67	0.185	1334	0.003	259.309

Numerical Results - Permanent

To apply the Gibbs sampler for permanent we adopt the concept of "neighboring" elements, (see, chapter 10 of Ross, (2002) and chapter 6 of Rubinstein and Kroese, (2007)). In the latter reference it is called *2-opt* heuristics. Given a point (tour) $\boldsymbol{x}$ of length m_t generated by the pdf $g_t = g(\boldsymbol{x}, m_t)$, the conditional Gibbs sampling updates the existing tour $\boldsymbol{x}$ to $\widetilde{\boldsymbol{x}}$, where $\widetilde{\boldsymbol{x}}$ is generated from $g(\widetilde{\boldsymbol{x}}, m_t)$ and where $\widetilde{\boldsymbol{x}}$ is the same as $\boldsymbol{x}$ with one exception that the points x_i and x_j in $\widetilde{\boldsymbol{x}}$ are reversed.

Numerical Results - Permanent

Performance of cloning Algorithm for permanent with $A = 30 \times 30$ matrix.

Run N_0	Iterations	$\lvert\widetilde{\mathscr{X}}^*\rvert$	RE of $\lvert\widetilde{\mathscr{X}}^*\rvert$	$\lvert\hat{\mathscr{X}}^*_{dir}\rvert$	RE of $\lvert\hat{\mathscr{X}}^*_{dir}\rvert$	CPU
1	21	261.14	0.018	266	0	115.68
2	21	254.45	0.043	266	0	115.98
3	21	268.04	0.008	266	0	115.65
4	21	272.20	0.023	266	0	117.68
5	21	261.50	0.017	266	0	118.38
6	21	255.03	0.041	266	0	117.10
7	21	261.36	0.017	266	0	116.58
8	21	266.82	0.003	266	0	115.82
9	21	264.76	0.005	266	0	115.84
10	21	254.13	0.045	266	0	116.13
Average	21	261.04	0.020	266	0	116.49

Numerical Results - Hamiltonian Cycles

We solve the Hamiltonian cycles problem by applying again the 2-opt heuristic used for the permanent.
Observations:

- The set of Hamiltonian cycles of length n presents a subset of the associated permanent trajectories set.
- The latter set is formed from cycles of length $\leq n$.

Conclusion: The following simple procedure can be used to calculate the number of Hamiltonian Cycles.

Numerical Results - Hamiltonian Cycles

1. Run the cloning Algorithm and calculate the estimator of $|\mathcal{X}^*|$ of the associated permanent using the product formula. Denote such permanent estimator by $|\widehat{\mathcal{X}^*}_p|$.
2. Proceed with one more iteration of cloning Algorithm and calculate the ratio of the number of screened Hamiltonian elite cycles (cycles of length n) to the number of the screened elite samples (samples of length $\leq n$) in the permanent. Denote the ratio as ζ.
3. Deliver $|\widehat{\mathcal{X}^*}_H| = \zeta|\widehat{\mathcal{X}^*}_p|$ as the estimator of the number $|\mathcal{X}^*|$ of Hamiltonian cycles.

Numerical Results - Hamiltonian Cycles

Performance of cloning Algorithm for Hamiltonian cycle problem with $A = 30 \times 30$ matrix.

Run N_0	Iterations	$\lvert\widetilde{\mathscr{X}}^*\rvert$	RE of $\lvert\widetilde{\mathscr{X}}^*\rvert$	CPU
1	14	2.38E+20	0.046	63.295
2	14	2.36E+20	0.033	62.809
3	14	2.28E+20	0.000	62.791
4	14	2.17E+20	0.049	62.714
5	14	2.35E+20	0.029	62.806
6	14	2.18E+20	0.046	62.951
7	14	2.10E+20	0.078	64.009
8	14	2.35E+20	0.032	62.790
9	14	2.23E+20	0.024	62.816
10	14	2.41E+20	0.057	62.614

Numerical Results - Summary

Test case	RAN	Split1	Split2	Split3	Cloning	CMC
3-SAT 20x80	-	+	+	+	+	NaN
3-SAT 75x325	-	-	-	+	+	NaN
3-SAT 122x663	-	-	-	+	+	NaN
3-SAT 20x100	NaN	NaN	NaN	NaN	+	NaN
Coloring 20x20	+	NaN	NaN	NaN	+	+
Coloring 40x40	-	-	-	+	+	-
Permanent 30x30	-	-	-	+	+	NaN
Hamiltonian 30x30 (1)	-	-	-	+	+	NaN
Hamiltonian 30x30 (2)	-	-	-	+	+	NaN
Hamiltonian 30x30 (3)	-	-	-	+	+	NaN
Polytope 11x10	+	+	+	+	+	+
Polytope 41x40	+	+	+	+	+	NaN
Polytope 61x30	+	+	+	+	+	NaN

Discrete Hit And Run

(1) Initialize $\boldsymbol{X}_1 \in \mathscr{X}^*$ and set $t = 1$.

(2) Generate a bidirectional walk by generating two independent nearest neighbor random walks in $\mathcal{R}$ that start at $\boldsymbol{X}_t$ and end when they step out of $\mathcal{R}$. One random walk is called the *forward walk* and the other is called the *backward walk*. The bidirectional walk may have loops but has finite length with probability 1. The sequence of points visited by the bidirectional walk is stored in an ordered list, denoted $\mathcal{L}_t$.

Discrete Hit And Run

(3) Generate a candidate point $\boldsymbol{Y}$ uniformly distributed on the intersection $\mathcal{M}_t = \mathscr{X}^* \cap \mathcal{L}_t$.

(4) Set

$$\boldsymbol{X}_{t+1} = \begin{cases} \boldsymbol{Y} & \text{with probability } \alpha(\boldsymbol{X}_t, \boldsymbol{Y}) \\ \boldsymbol{X}_t & \text{otherwise.} \end{cases}$$

(4) If a stopping criterion is met, stop. Otherwise, increment t and return to Step 2.

Discrete Hit And Run

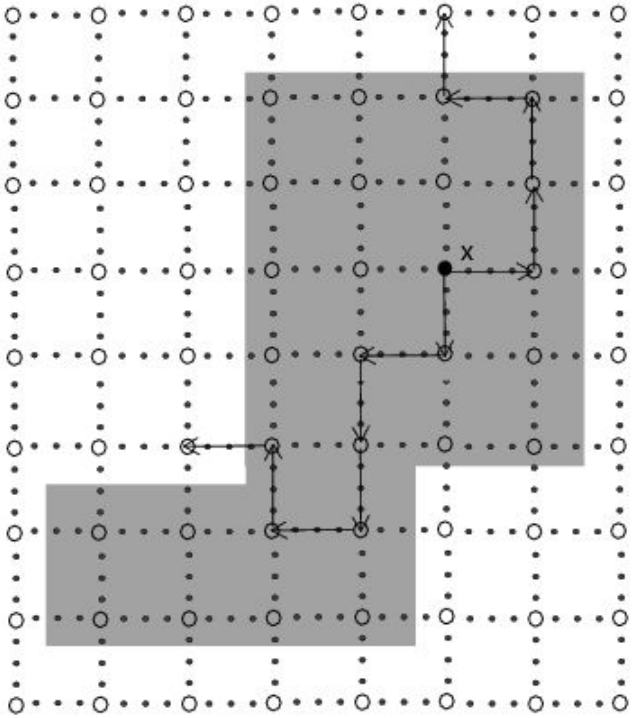


Figure 2: 2 typical random walks (forward and backward) from point x .

Performance of Splitting DHR Algorithm for the SAT 75×325 model.

Run N_0	Iterations	$\lvert\widetilde{\mathscr{X}}^*\rvert$	RE of $\lvert\widetilde{\mathscr{X}}^*\rvert$	$\lvert\hat{\mathscr{X}}^*_{dir}\rvert$	RE of $\lvert\hat{\mathscr{X}}^*_{dir}\rvert$	CPU
1	26	2.01E+04	8.038	1269	0.430	346.8
2	27	8.81E+01	0.960	972	0.563	318.8
3	26	1.67E+05	74.045	1669	0.250	310.4
4	29	1.27E+01	0.994	560	0.748	324.3
5	26	2.51E+04	10.268	1745	0.216	317.1
6	25	3.34E+06	1500.172	1976	0.112	162.7
7	26	3.25E+05	145.289	1936	0.130	312.8
8	26	5.26E+02	0.764	1177	0.471	315.6
9	26	1.85E+05	81.964	588	0.736	310.5
10	25	3.27E+06	1468.711	1468	0.340	303.1
Average	26.2	7.33E+05	329.121	1336	0.400	302.2

Conclusions

- In the sequence RAN $\rightarrow$ Split1 $\rightarrow$ Split2 $\rightarrow$ Split3 $\rightarrow$ cloning the statistical properties of the counting estimators of $|\mathscr{X}^*|$ substantially improve.
- For most of our case studies the original RAN algorithm typically fails, it either does not converge at all [is stuck at some intermediate level $m_t,\ (m_t < m)$], or is heavily biased (converges to a local extremum). Exceptions are convex counting problems, like estimating the volume of a convex plytope.
- As compared to the randomized algorithms, the proposed splitting algorithms require very little warm-up time when running the Gibbs sampler from iteration to iteration, since the underlying Markov chains are already in steady-state from the beginning.

Further Research

- Establish solid mathematical grounding based on the splitting method.
- Find the total sample size and the size of the elites at each iteration of the splitting Algorithm while estimating the rare-event probability

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