A contribution review
In Memoriam of Professor Reuven Rubinstein

Slava Vaisman
Faculty of Industrial Engineering and Management
Technion, Israel Institute of Technology
Haifa, Israel

May 18, 2013
1 About Reuven

2 Score Function

3 Cross-Entropy Method

4 Splitting Method for Counting and Optimization

5 Stochastic Enumeration
About Reuven

- 1973 - joined the Faculty of Industrial Engineering and Management of the Technion, Israel Institute of Technology.
- 1992 - Full Professor position.
- More than a hundred published papers in scientific journals.
Books

In Russian


In English


To Appear

- Rubinstein, R.Y. and Kroese D. P. Simulation and the Monte Carlo Methods: 3rd Edition
Research interests

- Computer Simulation
- Sensitivity Analysis
- Stochastic systems in a real-world applications
- Stochastic Optimization
- Probability of Rare Events
- Evolutionary Heuristic
Main Results

- The Score Function Method
- Cross-Entropy Method
- The Splitting Method
- Stochastic Enumeration Algorithm
Main results - Score Function Method

- How the change of interatrial, service rates or a buffer size will affect the performance in a queue?
- Consider

\[ \ell(u) = \mathbb{E}_u[H(X)] \]  

(1)

where the distribution of the sample performance \( H(X) \) depends on the control or reference parameter \( u \in \mathcal{V} \).

- Sensitivity analysis is concerned with evaluating sensitivities (gradients, Hessians, etc.) of the response function \( \ell(u) \) with respect to parameter vector \( u \). Sensitivity analysis provides guidance for design and operational decisions and plays an important role in selecting system parameters that optimize certain performance measures.
The cross-entropy (CE) method is a powerful technique for solving difficult estimation and optimization problems, based on Kullback-Leibler (or cross-entropy) minimization. It was pioneered by Rubinstein in 1999 as an adaptive importance sampling procedure for the estimation of rare-event probabilities. It was also shown that many optimization problems can be translated into a rare-event estimation problem.
Applications

- Combinatorial Optimization (e.g., Travelling Salesman, Maximal Cut and Quadratic Assignment Problems)
- Noisy Optimization (e.g., Buffer Allocation, Financial Engineering)
- Multi-Extremal Continuous Optimization
- Pattern Recognition, Clustering and Image Analysis
- Production Lines and Project Management
- Network Reliability Estimation
- Vehicle Routing and Scheduling
- DNA Sequence Alignment
To date very little is known about how to construct efficient algorithms for hard counting problems (rare events)

First approach - Importance Sampling (CE)

Second approach - MCMC (Splitting)
The basic procedure for counting

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

2. Find a sequence of decreasing sets $\mathcal{X} = \mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_m$ such that

$$\mathcal{X}_0 \supset \mathcal{X}_1 \supset \cdots \supset \mathcal{X}_m = \mathcal{X}^*$$

and $|\mathcal{X}| = |\mathcal{X}_0|$ is known.

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

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

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

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

should be not too 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 $\hat{c}_t$ for each $c_t$ and estimate $|\mathcal{X}^*|$ by

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

where $\hat{\ell} = |\mathcal{X}_0| \prod_{t=1}^{m} \hat{c}_t$. 
In order to deliver a meaningful estimator of $|X^*|$, we have to solve the following two major problems:

1. Construct the sequence $X_0 \supset X_1 \supset \cdots \supset X_m = X^*$ such that each $c_t$ is not a rare-event probability.
2. Obtain a low variance unbiased estimator $\hat{c}_t$ of each $c_t = |X_t|/|X_{t-1}|$. 
Consider the counting problem with $T$ subsets, that is

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

where $\ell = \prod_{t=1}^{T} |\mathcal{X}_t|/|\mathcal{X}_{t-1}|$. We assume that the subsets $\mathcal{X}_t$ are associated with levels $m_t$ and can be written as

$$\mathcal{X}_t = \{ x \in \mathcal{X} : S(x) \geq m_t \} \quad t = 1, \ldots, T,$$

where $S : \mathcal{X} \to \mathbb{R}$ is the sample performance function and $m_1 \leq m_2 \leq \cdots \leq m_T = m$. 
Figure: Iteration 1 of the splitting algorithm.
Algorithm (Fixed Level Splitting Algorithm for Counting)

1. **Acceptance-Rejection.** Set a counter \( t = 1 \). Generate a sample \( [X]_0 = \{X_1, \ldots, X_{N_0}\} \) uniformly on \( \mathcal{X}_0 \). Let \( [X]_1^{(e)} \) be the subset consisting of the samples for which \( S(X_i) \geq m_1 \). Suppose that \( N_1^{(e)} \) is the size of this subset, and denote these samples by \( \tilde{X}_i \); thus \( [X]_1^{(e)} = \{\tilde{X}_1, \ldots, \tilde{X}_{N_1^{(e)}}\} \). We simply say that this subset is the elite sample of the points with \( S(X_i) \geq m_1 \). Take \( \hat{c}_1 = \frac{1}{N_0} \sum_{i=1}^{N_0} I\{S(X_i) \geq m_1\} = \frac{N_1^{(e)}}{N_0} \) as an unbiased estimator of \( c_1 \). Note that \( \tilde{X}_1, \ldots, \tilde{X}_{N_1^{(e)}} \sim U(\mathcal{X}_1) \).

2. **Splitting.** Reproduce (clone) \( \eta_t \) times each sample point \( \tilde{X}_i \), \( i = 1, \ldots, N_t^{(e)} \) of the elite sample, that is take \( \eta_t \) identical copies of each point. Set \( N_t = \eta_t N_t^{(e)} \).

3. **Mutation** To each of the cloned points apply the uniform mutation mapping \( \Phi_t \). Denote the new sample by \( [X]_t = \{X_1, \ldots, X_{N_t}\} \). Note that each point in the sample is distributed uniformly on \( \mathcal{X}_t \).

4. **Selecting elites.** Determine the elite sample, i.e., the \( N_{t+1}^{(e)} \) points for which \( S(X_i) \geq m_{t+1} \). Denote the elite points by \( \{\tilde{X}_1, \ldots, \tilde{X}_{N_{t+1}^{(e)}}\} \).

5. **Estimating** \( c_{t+1} \). Take \( \hat{c}_{t+1} = \frac{1}{N_t} \sum_{i=1}^{N_t} I\{S(X_i) \geq m_{t+1}\} = \frac{N_{t+1}^{(e)}}{N_t} \) as an estimator of \( c_{t+1} \).

6. **Stopping rule and estimation.** If \( m_{t+1} = m \) stop and deliver the estimators otherwise, increase the counter to \( t = t + 1 \), and repeat from step 2.
Consider a 3-SAT problem consisting of $n = 20$ literals and $m = 80$ clauses $A = (20 \times 80)$ and $|\mathcal{X}^*| = 15$. We applied the Chi-square test for uniformity of $N = 5000$ for the following cases.

- $\varrho = 0.05$ and no additional iterations ($k = 0$);
- $\varrho = 0.5$ and no additional iterations ($k = 0$);
- $\varrho = 0.5$ and one additional iteration ($k = 1$).

All three experiments passed the Chi-square test at level $\alpha = 0.05$: the observed test values were $13.709$, $9.016$, and $8.434$, respectively, against critical value $\chi^2_{14,0.95} = 23.685$. The corresponding histograms are shown in the following Figure.

Figure: Histograms of 5000 samples for the 3-SAT problem.
Sequential importance sampling (SIS) is simply importance sampling carried out in a sequential manner. To explain, consider the expected performance

\[ \ell = \mathbb{E}_f[S(\mathbf{X})] = \int S(\mathbf{x}) f(\mathbf{x}) \, d\mathbf{x}, \quad (7) \]

where \( H \) is the sample performance and \( f \) is the probability density of \( \mathbf{X} \). Let \( g \) be another probability density such that \( H f \) is dominated by \( g \). That is, \( g(\mathbf{x}) = 0 \Rightarrow S(\mathbf{x}) f(\mathbf{x}) = 0 \). Using the density \( g \) we can represent \( \ell \) as

\[ \ell = \int S(\mathbf{x}) \frac{f(\mathbf{x})}{g(\mathbf{x})} \, g(\mathbf{x}) \, d\mathbf{x} = \mathbb{E}_g \left[ S(\mathbf{X}) \frac{f(\mathbf{X})}{g(\mathbf{X})} \right], \quad (8) \]

Such a density is called the importance sampling density.
Suppose that (a) \( \mathbf{X} \) is decomposable, that is, it can be written as a vector \( \mathbf{X} = (X_1, \ldots, X_n) \), where each of the \( X_i \) may be multi-dimensional, and (b) it is easy to sample from \( g(\mathbf{x}) \) sequentially. Specifically, suppose that \( g(\mathbf{x}) \) is of the form

\[
g(\mathbf{x}) = g_1(x_1) g_2(x_2 | x_1) \cdots g_n(x_n | x_1, \ldots, x_{n-1}).
\]  

(9)

To further simplify the notation, we abbreviate \( (x_1, \ldots, x_t) \) to \( \mathbf{x}_{1:t} \) for all \( t \). In particular, \( \mathbf{x}_{1:n} = \mathbf{x} \).
Self-Avoiding Walk of Length $n = 130$
One-Step-Look-Ahead (OSLA) Procedure

OSLA is the state of the art procedure due to Rosenbluth and Rosenbluth (1959).

1. Start from $X_0 = (0, 0)$. Let $d_t$ be the number of neighbors of $X_{t-1}$ that have not yet been visited. If $d_t > 0$, choose $X_t$ with probability $1/d_t$ from its neighbors. If $d_t = 0$ stop generating the path.

2. Stop if $t = n$. Otherwise increase $t$ by 1 and go to step 2.

Note that the procedure either generates a SAW $\mathbf{x}$ of fixed length $n$ or the path gets value zero. The product rule pdf $g(\mathbf{x})$ is

$$g(\mathbf{x}) = \frac{1}{d_1} \frac{1}{d_2} \cdots \frac{1}{d_n} = \frac{1}{w(\mathbf{x})}, \; (w(\mathbf{x}) = d_1 \ldots d_n). \quad (10)$$
OSLA Algorithm for SAW

1. Generate independently $M$ paths $X_1, \ldots, X_M$ via the OSLA procedure.

2. For each SAW $X_k$ compute the corresponding $w(X_k)$ as above. For the other parts (which do not reach the value $n$) set $w(X_k) = 0$.

3. Return

$$|\hat{X}^*| = \frac{1}{M} \sum_{i=k}^{M} w(X_k).$$  \hspace{1cm} (11)
The drawback of OSLA is that it looses most of its trajectories even for moderate $n$, say $n = 100$. Figure below present a SAW trapped after 15 iterations.
We next extend OSLA to \( k \)-step-look ahead and in particular to \( n \)-step-look ahead, called \( n \)SLA. Here \( n \) denotes the size of the problem, such as the number of variables (literals) in SAT and the number of edges in a network. We assume that all \( n \) variables \( x_1, \ldots, x_n \) are binary, that is \( x \in \{0, 1\} \).

The \( n \)-SLA (based an oracle) is very similar to OSLA. Its major advantage versus OSLA: it never looses a trajectory.
Our main strategy (slogan) is as follows:
Use fast polynomial decision making oracles to solve \#P-sharp problems.
In particular we use
- BFS algorithm for solving a shortest path problem.
- Hungarian method for solving an assignment problem in polynomial time.
- DPLL decision making algorithm for counting the number of valid assignments in 2-SAT.
As mentioned the major advantage of $n$SLA versus OSLA - it never looses trajectories. Its main drawback is that the generated trajectories are not uniformly distributed. As results its estimators are heavily biased. To overcome this difficulty we modify $n$SLA as:
Instead of a single trajectory we ran in parallel multiple ones. This will improve dramatically the non-uniformity issue.
To see how SE improves \( n \)SLA consider a 2-SAT model with clauses \( C_1 \land C_2 \land \ldots \land C_n \), where \( C_i = x_i \lor \overline{x}_{i+1} \geq 1 \). Figure below presents a graph with \( n = 4 \) variables and \( |X^*| = 5 \).

It can be shown that \( E[|X^*|^2] = 2|X^*| + 2|X^*|^{-1} - 2 \).
The table below corresponds to the above figure for \( n = 99 \) and \( | \mathcal{X}^* | = 100 \). It shows how bad SE works for \( N^{(e)} = 1 \), (which is \( nSLA \)) and how SE improves for \( N^{(e)} > 1 \). Here \( N^{(e)} \) denotes the number of multiple trajectories and RE-relative error.

| \((N^{(e)}, M)\) | \( | \tilde{\mathcal{X}}^* | \) | RE |
|------------------|-----------------|-----|
| \((N^{(e)} = 1, M = 500)\) | 11.110 | 0.296 |
| \((N^{(e)} = 10, M = 50)\) | 69.854 | 0.175 |
| \((N^{(e)} = 50, M = 10)\) | 100.11 | 0.032 |
Comparison of the efficiencies of SE and standard splitting. It follows that SE is about 50 times faster than splitting.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Time</th>
<th>SE</th>
<th>SE RE</th>
<th>Split</th>
<th>Split RE</th>
</tr>
</thead>
<tbody>
<tr>
<td>20x80</td>
<td>1 sec</td>
<td>15.0158</td>
<td>5.51E-03</td>
<td>14.97</td>
<td>3.96E-02</td>
</tr>
<tr>
<td>75x325</td>
<td>137 sec</td>
<td>2248.8</td>
<td>9.31E-03</td>
<td>2264.3</td>
<td>6.55E-02</td>
</tr>
<tr>
<td>75x270</td>
<td>122 sec</td>
<td>1.34E+06</td>
<td>1.49E-02</td>
<td>1.37E+06</td>
<td>3.68E-02</td>
</tr>
<tr>
<td>300x1080</td>
<td>1600 sec</td>
<td>3.32E+24</td>
<td>3.17E-02</td>
<td>3.27E+24</td>
<td>2.39E-01</td>
</tr>
</tbody>
</table>
Additional problems

- Permanent
- Shortest paths
- Vertex covers
Thank You!
Questions please