PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Reporte Técnico RT 08-21

Splitting in the Simulation of the Network Creation Process

Leslie Murray **Héctor Cancela** **Gerardo Rubino**

2008

Splitting in the Simulation of the Network Creation Process.
Murray, Leslie; Cancela, Héctor; Rubino, Gerardo
ISSN 0797-6410
Reporte Técnico RT 08-21
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República

Montevideo, Uruguay, 2008

Splitting in the Simulation of the Network Creation Process

Leslie Murray
FCEIA
Universidad Nacional de Rosario
Rosario, Argentina

Héctor Cancela
Facultad de Ingeniería
Universidad de la Repblica
Montevideo, Uruguay

Gerardo Rubino
IRISA/INRIA
Campus de Beaulieu
Rennes, France

Abstract

Splitting is a variance reduction technique, widely used to improve the efficiency of markovian systems simulations. In this report Splitting is successfully adapted to the reliability network estimation problem by means of a well–known model based on the so called Creation Process. A network model based on the Creation Process is revisited, a brief review of Splitting in a general setting is introduced and afterwards Splitting is applied to the estimation of the source–terminal network unreliability by means of the Creation Process Model. Finally a set of experiments to assess the performance of Splitting in this task, are shown.

Key Words – network reliability, Monte Carlo simulation, splitting

1 Introduction

Network models are an important tool to represent a large variety of real life problems, covering from transportation infrastructure up to personal interaction in social organization. Among the huge variety of network models that appear in the literature, source-terminal and multi-terminal network reliability models have been widely studied and employed in computer, telecommunication and electrical networks, as well as in transportation, space and military applications [4, 6, 11, 19].

As the exact computation of the source-terminal reliability measure is known to be NP-hard [16], Monte Carlo techniques are a useful alternative to make estimations rather than exact calculation. If γ is the metric being estimated (e.g. network unreliability), and $\widehat{\gamma}$ the associated estimator, a typical measure of the relative error is σ/γ, where $\sigma^2 = \mathbb{V}(\widehat{\gamma})$. In general this error (accepted as a measure of accuracy) grows when the component's reliabilities grow and, hence, the overall network unreliability γ goes to zero. In order to make Monte Carlo estimations more accurate, many different proposals, like [2, 5, 12, 13, 15, 17], have attempted to reduce the variance of the estimators.

Splitting [7, 9, 10, 14, 18] is a variance reduction technique widely used to improve markovian systems simulations. In this report Splitting is successfully adapted to the source–terminal reliability network estimation problem. Once adapted, its performance is subject to empirical evaluation over many different benchmark network topologies. Splitting shows a very stable precision within a relatively small number of Monte Carlo iterations, particularly in the unreliability estimation of highly reliable networks.

The rest of the report is organized as follows. Section 2 gives a brief review of Splitting, Section 3 introduces network models and Section 4 presents a Splitting application to solve one of the network models introduced in Section 3. Implementation details and experimental results are given in Sections 5 and 6 respectively, while conclusions and future work are assessed in Section 7. At the report's end, appendixes A, B, C and D contain a large set of graphics to complement Section 6, showing the main experimental results.

2 Splitting – General Setting

Given a Markov process $X(t)$, $t \geq 0$, with state space $\mathcal{X}$, and given two disjoint regions $\mathcal{X}_A$ and $\mathcal{X}_B$ of $\mathcal{X}$, it may be of interest to find the probability that, evolving from $t = 0$, $X(t)$ enters $\mathcal{X}_B$ without

having entered $\mathcal{X}_A$ before. Being τ_A the time at which $X(t)$ enters $\mathcal{X}_A$ for the first time (or comes back to $\mathcal{X}_A$ if $X(0) \in \mathcal{X}_A$) and τ_B the time at which $X(t)$ enters $\mathcal{X}_B$ for the first time, the problem is to find the probability of the event $[\tau_B < \tau_A]$. Splitting [7, 9, 10, 14, 18] makes efficient estimations of this probability, particularly when $[\tau_B < \tau_A]$ is a rare event.

$\mathcal{X}_A$ and $\mathcal{X}_B$ can be expressed in terms of two different regions, in a different space obtained via an *importance function* $h : \mathcal{X} \to \mathbb{R}$. A possible definition for $\mathcal{X}_A$ and $\mathcal{X}_B$ can be $\mathcal{X}_A = \{x \in \mathcal{X} : h(x) \leq 0\}$ and $\mathcal{X}_B = \{x \in \mathcal{X} : h(x) \geq \ell\}$ where both, 0 and ℓ, define bounds or thresholds in $\mathbb{R}$.

Let's now consider a set of thresholds $\ell_0 = 0 < \ell_1 < \ell_2 < \cdots < \ell_m = \ell$ to partition the state space of $h(X(t))$ as shown in Figure 1(a).

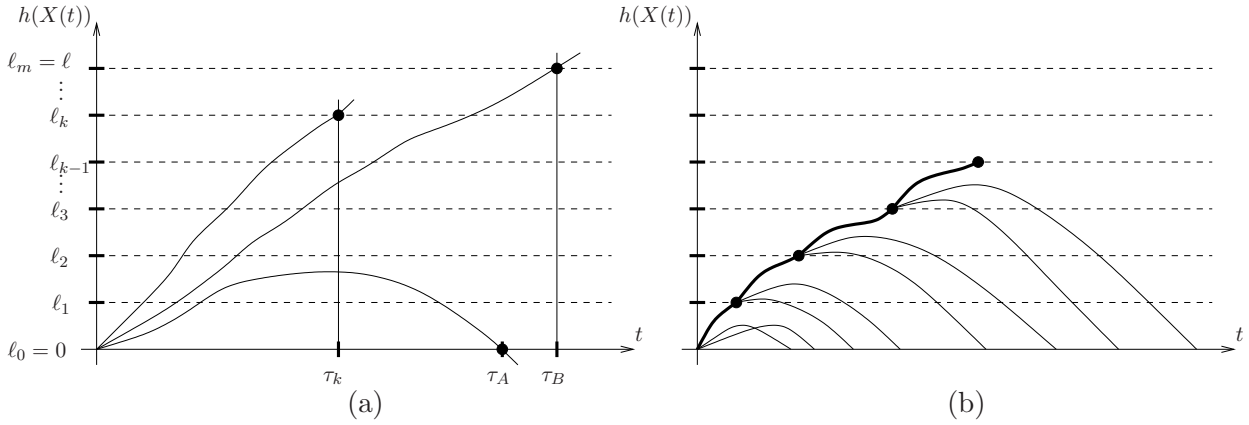

Figure 1: State Space of $h(X(t))$

Being τ_k the time at which process $h(X(t))$ up–crosses ℓ_k for the first time, it is possible to define the events $D_k = [\tau_k < \tau_A]$, $k = 1, 2, \cdots, m$, as an indication that process $h(X(t))$ has crossed threshold ℓ_k without having entered the region under threshold $\ell_0 = 0$. The following set of nested events can be defined then: $D_m \subset D_{m-1} \subset \cdots \subset D_2 \subset D_1$, being $D_m = [\tau_B < \tau_A]$ the event whose probability γ_m is to be estimated:

$$\begin{aligned} \gamma_m &= \mathbb{P}[D_m] \\ &= \underbrace{\mathbb{P}[D_m|D_{m-1}]}_{p_m} \underbrace{\mathbb{P}[D_{m-1}|D_{m-2}]}_{p_{m-1}} \cdots \underbrace{\mathbb{P}[D_2|D_1]}_{p_2} \underbrace{\mathbb{P}[D_1]}_{p_1} = \prod_{k=1}^{m} p_k \end{aligned}$$

This is the exact determination of γ_m. An estimation $\widehat{\gamma}_m$ can be done as $\widehat{\gamma}_m = \prod_{k=1}^{m} \widehat{p}_k$, where every $\widehat{p}_k$ is an estimation of $p_k,\ k = 1, 2, \cdots, m$. The product holds, and the estimation is unbiased, even though the values of $\widehat{p}_k$ are not necessarily independent [7]. Then, separate estimations of every $\widehat{p}_k$ must be done.

To determine $\widehat{p}_1$, N_0 independent realizations of $X(t)$ are started from $X(0)$. If R_1 of these realizations cross threshold ℓ_1 without falling under ℓ_0 before, the estimation results $\widehat{p}_1 = R_1/N_0$. At the cross of ℓ_1, the states of the R_1 realizations are saved (*saved states*). The remaining $m - 1$ estimations $\widehat{p}_k,\ k = 2, 3, \cdots, m$ are done similarly: N_{k-1} realizations of $X(t)$ are started from the R_{k-1} *saved states*, the number R_k is determined and $\widehat{p}_k$ is obtained as $\widehat{p}_k = R_k/N_{k-1}$. Every stage of the experiment is equivalent to observing N_k Bernoulli variables with parameter $p_k,\ k = 1, 2, \cdots, m$.

It is desirable that $N_{k-1} > R_{k-1}$, however the relation between N_{k-1} and R_{k-1} is not yet analyzed but delayed for the next sections instead. But the consequence of $N_{k-1} > R_{k-1}$ is that from every *saved state* more than one realization may need to be started, being necessary to split or clone some of them, at every threshold cross, like shown in Figure 1(b). Finally, the estimation of interest results: $\widehat{\gamma}_m = R_1/N_0\ R_2/N_1\ R_3/N_2\ \cdots\ R_m/N_{m-1}$.

An analysis done in [8] reports a value of $\gamma_m^2 m^2 (1 - \gamma_m^{1/m})/\gamma_m^{1/m} N$ for the variance $\mathbb{V}\{\widehat{\gamma}_m\}$, being m the number of thresholds and N the *effort* (for the concept of *effort* see *Fixed Splitting* in Section 5). In [14] $\mathbb{V}\{\widehat{\gamma}_m\} = m\gamma_m^{2-(1/m)}/N$. Both calculations have been done for a very simple setting in which the values of $p_k,\ k = 1, 2, \cdots, n$ are all independent and equal.

3 Network Modeling

An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of n absolutely reliable nodes and $\mathcal{E}$ a set of m independent unreliable links, is a frequently used model for studying communication network reliability. Being $X_i = 1$ when the i^{th} link is *operative* and $X_i = 0$ when the i^{th} link is *failed*, vector $\mathbf{X} = (X_1, X_2, \cdots, X_m)$ depicts the state of the links and, somehow, the state of the whole network. $r_i = \mathbb{P}[X_i = 1]$ is the i^{th} single link reliability, whereas $q_i = 1 - r_i = \mathbb{P}[X_i = 0]$ the single link unreliability.

The structure function $\Phi(\mathbf{X})$ equals 1 when the network is *operative* and 0 when the network is *failed*. Using this function, we define the network reliability R and unreliability Q respectively as $\mathbb{P}[\Phi(\mathbf{X}) = 1]$ and $\mathbb{P}[\Phi(\mathbf{X}) = 0]$. From now on $\Phi(\mathbf{X})$ will be associated to the source–terminal network reliability model in which the network is considered *operative* if there is a path of *operative* links between two nodes s and t, and *failed* otherwise.

Straightforward (also called standard, direct or crude) Monte Carlo estimations of network reliability and unreliability can be computed, respectively, as $\widehat{R} = 1/N \sum_{i=1}^{N} \Phi(\mathbf{X}^{(i)})$ and $\widehat{Q} = 1/N \sum_{i=1}^{N} (1 - \Phi(\mathbf{X}^{(i)})$ where $\mathbf{X}^{(i)},\ i = 1, \cdots, N$ are *i.i.d.* samples taken from $f_{\mathbf{X}}(\mathbf{x})$ (the probability mass function of vector $\mathbf{X}$). For highly reliable networks most of the $X^{(i)}$ samples will equal 1 and, as a consequence, most of the replications $\Phi(\mathbf{X}^{(i)})$ will equal 1 also. For extremely reliable networks, with unreliabilities in the order of 10^{-10} or even less, it will take an average of 10^{10} replications to get $\Phi(\mathbf{X}^{(i)}) = 0$ at least once. For these networks $\Phi(\mathbf{X}) = 0$ is a rare event.

Another crude Monte Carlo approach consists of watching the links evolve in time, supposing that at time $t = 0$ all of them are *failed*, and that they become *operative* at times $\tau_i,\ i = 1, 2, \cdots, m$, exponentially distributed with parameter λ_i. Variable $\mathbf{X} = (X_1, X_2, \cdots, X_m)$ should then be replaced by $\mathbf{X}(t) = (X_1(t), X_2(t), \cdots, X_m(t))$, a Markov Process called *Creation Process* [5]. Then $X_i(t) = 1$ (i^{th} link *operative*) if $t \geq \tau_i$ and $X_i(t) = 0$ (i^{th} link *failed*) if $t < \tau_i$.

Since $\mathbb{P}[\tau_i \leq t] = 1 - e^{-\lambda_i t}$ is the probability that the i^{th} link becomes operative at time t or earlier, the choice of $\lambda_i = -\log(q_i)$ makes the probability that the i^{th} link is operative at $t = 1$ be exactly the single link reliability r_i. As a consequence the probability that the whole network is *operative* at $t = 1$ is R, and the probability that the network is *failed* at $t = 1$ is Q. Thereafter, $R = \mathbb{E}\{\Phi(\mathbf{X}(1))\}$ and $Q = \mathbb{E}\{1 - \Phi(\mathbf{X}(1))\}$.

The natural way to simulate this model (from now on referred to as *Creation Process Model*) is (i) to sample a set of exponentially distributed times $\tau_i,\ i = 1, 2, \cdots, m$, (ii) to check the value of $\Phi(\mathbf{X}(1)^{(i)})$ every sampled set, (iii) to repeat items (i) and (ii) N times, and finally to average the N values of either $\Phi(\mathbf{X}(1)^{(i)})$ to estimate R or $1 - \Phi(\mathbf{X}(1)^{(i)})$ to estimate Q. From now on the replications index will be omitted and $\mathbf{X}(1)^{(i)}$ will be referred to as $\mathbf{X}(1)$.

The main weakness of these crude estimations comes up when the networks are highly reliable. Particularly the relative precision of the unreliability estimations drops dramatically as Q tends to 0. For this estimations the exact variance is $\mathbb{V}\{\widehat{Q}\} = Q(1 - Q)/N$ and the relative error $\mathbb{V}\{\widehat{Q}\}^{1/2}/\mathbb{E}\{\widehat{Q}\} = ((1-Q)/NQ)^{1/2}$. Hence, for highly reliable networks (Q extremely small), an accurate estimation requires a very large sample size N.

4 Splitting Network Model

The core of this work is a Splitting adaptation to estimate the source–terminal network reliability measure. This estimation is performed on the network *Creation Process Model* model discussed in Section 3. The following paragraphs describe the fundamentals of this proposal.

Let's arrange the set $\mathcal{T} = \{\tau^1, \tau^2, \cdots, \tau^m\}$ of exponentially distributed (commutation) times[1], so that $\tau^1 \leq \tau^2 \leq \cdots \leq \tau^m$, and consider it as a random process trajectory. In correspondence with this trajectory, and under the same index considerations, let's define $\Lambda = \{\lambda^1, \lambda^2, \cdots, \lambda^m\}$, the arranged set of the corresponding rates.

As shown in Figure 2(a), for any given trajectory $\mathcal{T}$, event $E = [\Phi(\mathbf{X}(1)) = 0]$ occurs if the network becomes operative after $t = 1$, therefore $Q = \mathbb{P}[E]$. In a Crude Monte Carlo setting, where a considerable number of trajectories are sampled, the estimation $\widehat{Q}$ is the ratio between the number of successful events E and the total number of trajectories.

[1] A more explicit notation for every τ^i would be τ^i_j, meaning that link j goes from *failed* to *operative* in time τ^i. However, depending on the context, it might be enough with only one index.

Referring to the Splitting–General Setting of Section 2, the occurrence of event E, i.e. a trajectory $\mathcal{T}$ entering region $t > 1$ with the network still *failed*, is equivalent to process $X(t)$ entering region $\mathcal{X}_B$ without having entered region $\mathcal{X}_A$. On the other side, as shown in Figure 2(b), a trajectory $\mathcal{T}$ for which the network is *operative* in $t \leq 1$ is equivalent to process $X(t)$ entering region $\mathcal{X}_A$ without having reached region $\mathcal{X}_B$.

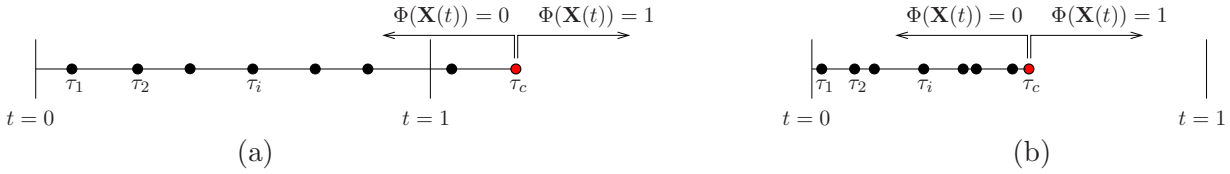

Figure 2: Two Different Trajectories in the Network *Dynamic Model*

In order to apply Splitting to this network *Creation Process Model*, the interval $[0, 1]$ needs to be partitioned by a set of thresholds $u_0 = 0 < u_1 < u_2 < \cdots < u_n = 1$. Given this threshold definition it is possible to define $E_k = [\Phi(\mathbf{X}(u_k)) = 0],\ k = 1, 2, \cdots, n$ as the indicator event that the network becomes operative after $t = u_k$. Then, $E = E_n$ and:

$$\begin{aligned} Q &= \mathbb{P}[E_n] \\ &= \underbrace{\mathbb{P}[E_n|E_{n-1}]}_{p_n}\ \underbrace{\mathbb{P}[E_{n-1}|E_{n-2}]}_{p_{n-1}}\ \cdots\ \underbrace{\mathbb{P}[E_2|E_1]}_{p_2}\ \underbrace{\mathbb{P}[E_1]}_{p_1} = \prod_{k=1}^{n} p_k \end{aligned}$$

Hence, $\widehat{Q} = \prod_{k=1}^{n} \widehat{p}_k$. It is necessary then to make separate estimations of the single values $\widehat{p}_k$. These estimations are done as part of the whole Splitting mechanism that for the current problem can be summarized as follows: start one or more trajectories $\mathcal{T}_1, \mathcal{T}_2, \cdots$ from $t = 0$ and then (i) cancel the ones for which event E_1 does not occur and (ii) split the ones for which event E_1 occurs; proceed the same way with all the new trajectories started from u_1, i.e. cancel or split at the cross of threshold u_2 and keep repeating the mechanism until threshold $u_n = 1$ is reached. In other words, cancel trajectories as soon as the network becomes *operative* and split them at every threshold cross. The target is to finally have some trajectories for which the network becomes operative beyond $t = 1$. If this happens, the following estimations are possible: $\widehat{p}_k = R_k/R_{k-1},\ k = 1, 2, \cdots, n$, where R_k is the total number of trajectories that have crossed threshold u_k and R_0 the number of trajectories $\mathcal{T}_1, \mathcal{T}_2, \cdots$ started from $t = 0$.

Two issues mentioned in the prior paragraphs deserve further explanation: how to generate trajectories and how to split them. Both are described in the rest of this section.

A straightforward way to sample $\mathcal{T} = \{\tau^1, \tau^2, \cdots, \tau^m\}$, $\tau^1 \leq \tau^2 \leq \cdots \leq \tau^m$, is to sample separate and independently every element as $\tau^i = -\log U/\lambda_i$ where $U \sim unif(0, 1]$, and once they are all sampled, to arrange them by increasing order of time. This procedure is however useless for the current Splitting application, because trajectories can only be sampled as a whole, being impossible to make them grow, adding new elements one by one.

Some properties of the *Creation Process* [5], due to the exponential distribution, make possible to add new elements to a partially sampled trajectory. Prior to introducing this sampling mechanism let's define sub–trajectories:

> $\mathcal{T}_-^i = \{\tau^1, \tau^2, \cdots, \tau^j\}$ is the subset of elements of $\mathcal{T}$ formed by the times that are earlier or equal to i: $\tau^j \leq i$ and $\tau^{j+1} > i$. In correspondence with $\mathcal{T}_-^i$ it is possible to define $\Lambda_-^i = \{\lambda^1, \lambda^2, \cdots, \lambda^j\}$.
>
> $\mathcal{T}_+^i = \{\tau^j, \tau^{j+1}, \cdots, \tau^m\}$ is the subset of elements of $\mathcal{T}$ formed by the times grater than i: $\tau^j > i$ and $\tau^{j-1} \leq i$. In correspondence with $\mathcal{T}_+^i$ it is possible to define $\Lambda_+^i = \{\lambda^j, \lambda^{j+1}, \cdots, \lambda^m\}$.

Based on these definitions, the procedure to sample trajectory elements, one by one, is as follows:

Sampling the next link Let the set of links whose commutation time belong to $\mathcal{T}_+^i$ in time i be $S_Z = \{e_r, e_s, \cdots, e_t\}$, then λ_r, $\lambda_s \cdots$ and λ_t belong to Λ_+^i. The first element of $\mathcal{T}_+^i$ can be sampled from the discrete random variable Z with sample space S_Z and $\mathbb{P}[e_w] = \lambda_w / \sum_{\lambda_v \in \Lambda_+^i} \lambda_v,\ w = r, s, \cdots, t$.

Sampling the time for the next link The times between consecutive commutation times of a trajectory: $\Delta^t = \tau^t - \tau^{t-1},\ t = 1, 2, \cdots, m$ (accepting $\tau^0 = 0$), are exponentially distributed with parameter $\sum_{\lambda_s \in \Lambda_t^+} \lambda_s$. Then for any time i, being $\mathcal{T}_i^-$ already sampled, it is possible to sample the time τ^{i+1} for the first position of $\mathcal{T}_+^i$, by adding Δ^{i+1} to τ^i, i.e. $\tau^{i+1} = \tau^i + \Delta^{i+1}$.

So, times $\tau_i,\ i = 1, 2, \cdots, m$ can be sampled *on demand*, being the resulting trajectory *under construction* all the time. Simulation conditions may be checked whenever necessary for all the existing trajectories and then, at any simulation stage, any one of them may grow by the addition of one or more elements.

The final issue to consider concerns the way to split trajectories every time a threshold is crossed. Suppose that trajectory $\mathcal{T} = \{\tau^1, \cdots, \tau^x, \tau^{x+1}\}$ is *under construction*, being $\tau^x < u_k < \tau^{x+1}$. Suppose also that the network is still *failed* at time τ^{x+1}, meaning that event E_k has actually occurred. Trajectory $\mathcal{T}$ must therefore be split at $t = u_k$. The value of τ^{x+1} has once been obtained as $\tau^{x+1} = \tau^x + \Delta^{x+1}$ where, as stated, it was $\tau^{x+1} > u_k$. The *memoryless* property let re–sample new statistically equivalent values for τ^{x+1} (as many as necessary), as $\tau^{x+1} = u_k + \Delta^{x+1}$, by just sampling new values of Δ^{x+1} every time. Then, if trajectory $\mathcal{T}$ has to be split n_k times, n_k new values can be obtained for τ^{x+1} and, consequently, n_k new trajectories may start at the cross of threshold u_k.

5 Implementation

There are two main options in a Splitting implementation: *Fixed Splitting* and *Fixed Effort*. If at every threshold cross the trajectories are split into a fixed number of new trajectories, the total number of trajectories started from every threshold is a random variable. This option is *Fixed Splitting*. Accepting that the whole simulation effort is proportional to the number of started trajectories, some sort of control on such a number would permit some control on the simulation times. To do this, the number of new trajectories should be controlled at every splitting point. *Fixed Effort* proposes to split as much as necessary to let the total number of trajectories that start from every threshold be a fixed number. There is no general agreement about the variance reduction benefits of every option, however *Fixed Effort* provides a closer control on the execution times.

A crucial issue in a Splitting implementation is how to select the number of thresholds. There is no recipe for such determination, only some guidelines derived from the analysis of very particular problems. These guidelines suggest to set this number to $-(\log \widehat{Q})/2$ where $\widehat{Q}$ is the probability being estimated. It is necessary then to dedicate some previous runs to set the number of thresholds. The value of $-(\log \widehat{Q})/2$ has been obtained separately and after different analysis by [8, 7] and [14] for the particular case in which the values of $p_k,\ k = 1, 2, \cdots, n$ are independent and equal. For these cases both analysis prove that $-(\log \widehat{Q})/2$ produces the maximum variance reduction. The use of this value is recommended anyway for any kind of splitting setting, at least as starting value for further iterations.

An important problem arising in typical Splitting implementations is the considerable computational effort that might be necessary to simulate the process since any threshold is crossed until the trajectory eventually "dies". In the most general case, after any threshold cross, the process may follow an *up* and *down* evolution before the final condition is reached (i.e. before it enters $\mathcal{X}_A$, in terms of Section 2). In the network *Creation Process Model*, this problem does not arise. The only wasted effort for any "dying" trajectory is the one devoted to take it closer to the next threshold but, if the next threshold is not reached, the trajectory "dies" suddenly with no additional effort, actually with the same effort that could have necessary to take it to the next threshold in the successful case.

6 Numerical Results

The experiments reported in the following sections have been conducted on a set of benchmark network topologies taken from [3]. These topologies are: Dodecahedron, Arpanet (1972 topology), K10, S2, S3, S4, S5 and S6, all of them with equi-reliable links. Their size, indicated by the number of nodes $|\mathcal{V}|$ and edges $|\mathcal{E}|$, are shown in Table 1.

Unless for special experiments, their single link reliabilities were set, alternatively, to `0.9`, `0.99`, `0.9999` and `0.999999`, resulting in a wide range of source-terminal unreliabilities (`9.53e-02` $< \widehat{Q} <$ `2.01e-54`). The Splitting implementation tested was *Fixed Effort*.

Table 1: Benchmark Network Topologies Size

Network	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$
Dodecahedron	20	30
Arpanet	21	26
K10	10	45
S2	4	5
S3	8	13
S4	14	25
S5	22	41
S6	32	61

Table 2: Source–Terminal Unreliability Estimation, Simulation Time and Relative Error

Network	$\widehat{\mathbf{Q}}_{\mathbf{max}}$	$\mathbf{t}[seg]$	$\mathbf{RE}[\%]$	$\widehat{\mathbf{Q}}_{\mathbf{min}}$	$\mathbf{t}[seg]$	$\mathbf{RE}[\%]$
Dodecahedron	2.87e-03	173.47	0.31	2.03e-18	1,278.89	0.57
Arpanet	9.53e-02	110.39	0.14	6.00e-12	708.48	0.44
K10	2.00e-09	802.65	0.49	2.01e-54	6,174.49	1.19
S2	2.16e-02	13.86	0.20	2.01e-12	134.55	0.38
S3	3.78e-03	56.06	0.27	2.99e-18	524.17	0.50
S4	6.02e-04	163.70	0.34	4.03e-24	1,379.40	0.71
S5	9.15e-05	372.12	0.38	4.98e-30	2,941.92	0.88
S6	4.31e-05	834.86	0.44	5.38e-36	6,193.18	2.61

In most experiments the Effort (i.e. the fixed number of trajectories started from every threshold) was set to 4,000 and N, the number of replications per experiment, was set to 200. Then the Total Effort (Total Effort = $N\times$ Effort) which is, at last, the total number of trajectories actually started in the experiment, was set to: $200 \times 4{,}000 = 8e{+}05$.

Out of every replication, a single estimation $\widehat{Q}_i,\ i = 1, 2, \cdots, N$ was obtained. Then, after the run of all replications, the unreliability, and its associated variance, were estimated respectively as $\widehat{Q} = 1/N \sum_{i=1}^{N} \widehat{Q}_i$ and $\widehat{\mathbb{V}}\{\widehat{Q}\} = \sum_{i=1}^{N} \widehat{Q}_i^2/(N(N-1)) - \widehat{Q}^2/(N-1)$.

6.1 Relative Error and Execution Time

As remarked in Section 3, the relative error (RE) is expected to grow together with the network reliability. Even though this point was observed (and proved) for the crude implementations, it is still valid for any implementation. Hence, in an experiment where the reliability grows (unreliability drops) it is desirable that the relative error grows as slow as possible (in the best case, not to grow at all).

The data shown in Table 2 is a summary for all the experiments. It shows the simulation time $\mathbf{t}$ and the relative error $\mathbf{RE}$ for two sets of the source–terminal network unreliability estimation of every network ($\widehat{\mathbf{Q}}_{\mathbf{max}}$ and $\widehat{\mathbf{Q}}_{\mathbf{min}}$, obtained by setting the link reliabilities respectively to 0.9 and 0.999999). The relative error, $\mathbf{RE} = \widehat{\mathbb{V}}\{\widehat{Q}\}^{1/2}/\widehat{Q}$, shows a very high independence with respect to the estimated value $\widehat{\mathbf{Q}}$.

Time $\mathbf{t}$ grows together with the network size and, for the same network, grows as the network becomes more reliable. The increase due to the network size is rather obvious and comes from the increasing number of calculation. The increase due to the reliability comes from growth of the number of thresholds and, therefore, the increasing number of steps necessary to make the splitting estimation.

6.2 Speedup vs. Number of Thresholds

Besides the relative error, some other parameters are useful to carry on performance analysis. The most important of them are: the execution time t and the variance of the estimation $\mathbb{V}\{\widehat{Q}\}$ (or alternatively $\widehat{\mathbb{V}}\{\widehat{Q}\}$). Both quantities can be considered together as the product $(t \times \mathbb{V}\{\widehat{Q}\})$, being this product particularly useful in comparative tests. Splitting was then compared to Crude Monte Carlo, for a variety of number of thresholds, assessing the quotient $(t_c \times \mathbb{V}\{\widehat{Q}_c\})/(t_s \times \mathbb{V}\{\widehat{Q}_s\})$ (index c holds for Crude Monte Carlo and s for Splitting), usually referred to as the Speedup or also Precision Gain, as it shows the precision improvement of the incumbent method s over method c, given a fixed computational time[1].

The Speedup values obtained are shown in Appendix A (Figures 3, 4, 5, 6, 7 and 8). The number of $-(\log \widehat{Q})/2$ thresholds is always close to the best performance one. However, as the best performance choice seems to be always slightly higher than $-(\log \widehat{Q})/2$, the recommendation derived from this experiment is to set the number of thresholds in $\lceil(-(\log \widehat{Q})/2)\rceil$. These recommended values are indicated by a filled dot in the figure. It is to notice how Splitting outperforms Crude Monte Carlo as the network reliability grows, being the improvements remarkable for highly reliable networks.

6.3 Relative Error vs. Effort

In the next set of experiments, the evolution of the relative error is observed for different values of Effort. It is expected that the error falls according to the square root of the Effort. This tendency can be seen in the graphics of Appendix B (Figures 9, 10, 11 and 12). This behaviour confirms that the Total Effort is high enough to let the variance estimations be steady around the expected value.

6.4 Relative Error vs. Number of Replications

In every experiment, N replications are averaged, each one of them consisting in a number Effort of trajectories started. A "small" value of N will cause the final average to lose accuracy and a "small" value of Effort may cause some of the thresholds to never be crossed and to consequently obtain an undefined result for the experiment. In order to find some bounds for these two values, an interesting experiment is to let N and Effort vary so as to keep constant their product, i.e. to let the Total Effort (Total Effort = $N\times$ Effort) be fixed at a certain value. In the proposed experiment Effort assumes values in a range from `40,000` down to `2,000` while, in correspondence, the number of replications N goes from `10` to `200`. Results are shown in Appendix C: Figure 13 for $N <$ `200` and Figure 14 for the cases of $N <$ `20`, in more detail. In both figures $N\times$ Effort = `400,000`.

The value of the relative error seems to be quite steady (and even more steady as higher are the values of N). This suggests that the relative error only depends on the value of the Total Effort. As N tends to the lowest values, the relative error is still around a certain fixed value, but the results are more scattered. Hence, depending on the expected accuracy, the recommendation is as follows: given an Effort that guarantees that threshold $t = 1$ is always reached, set the value of N so as to let the estimations be "not so" scattered (trying that a change on the random numbers seeds does not affect considerably the estimations). The values of $N =$ `200` and Effort = `4,000` initially used in prior experiments are acceptable for the ongoing problems.

6.5 Relative Error vs. Network Unreliability

As an extension of Table 2, Appendix D (Figure 15) shows many values of Relative Error, for unreliabilities between $\widehat{Q}_{max}$ and $\widehat{Q}_{min}$. It is also interesting to see that the errors grow as the networks become more reliable and also as the size of the networks grow higher. This feature is analyzed in the next section.

6.6 Splitting vs. Permutation Monte Carlo

Splitting is finally compared to a well–known algorithm that makes efficient estimations of highly reliable networks unreliability. The algorithm to compare to is Permutation Monte Carlo [12], a technique that also applies to the network *Creation Process Model* model discussed in Section 3. The basis of

Permutation Monte Carlo consists in sampling one permutation[2] per replication, assessing (by means of an exact formula) the probability that the sampled permutation keeps the network still failed at $t = 1$, and then averaging all the assessed probabilities.

The comparative test is done by means of the Speedup $(t_P \times \mathbb{V}\{\widehat{Q}_P\})/(t_S \times \mathbb{V}\{\widehat{Q}_S\})$ (index p holds for Permutation Monte Carlo and s for Splitting) for a variety of networks and for different single link reliabilities (S.L.R.). The Speedup values are recorded in Table 3. N was set to 200 and the number of thresholds to $\lceil(-(\log \widehat{Q})/2)\rceil$ in all the Splitting runs. The Effort and the number of Permutation replications were selected to let the factors $(t \times \mathbb{V}\{\widehat{Q}\})$ be as steady as possible. For convenience the unreliability estimation $\widehat{Q}$ is also reported in the table.

Table 3: Permutation Monte Carlo–Splitting Speedup

Network	**S.L.R.**	$\widehat{\mathbf{Q}}$	**Speedup**
Dodecahedron	0.9	2.87E-03	1,31
Arpanet	0.9	9.54E-02	0.43
S2	0.9	2.16E-02	0.06
S3	0.9	3.78E-03	0.27
S4	0.9	6.02E-04	1.18
S5	0.9	9.15E-05	6.67
S6	0.9	4.31E-05	2.86
Dodecahedron	0.99	2.08E-06	4.69
Arpanet	0.99	6.50E-04	0.61
S2	0.99	2.03E-04	0.03
S3	0.99	3.07E-06	0.25
S4	0.99	4.22E-08	2.71
S5	0.99	5.35E-10	84.80
S6	0.99	3.87E-11	119.64
Dodecahedron	0.9999	2.00E-12	2.56
Arpanet	0.9999	5.98E-08	0.26
S2	0.9999	1.99E-08	0.01
S3	0.9999	3.00E-12	0.08
S4	0.9999	4.04E-16	1.45
S5	0.9999	5.00E-20	52.47
S6	0.9999	2.12E-23	195.43
Dodecahedron	0.999999	2.01E-18	1.89
Arpanet	0.999999	6.00E-12	0.15
S2	0.999999	2.01E-12	0.00
S3	0.999999	3.02E-18	0.04
S4	0.999999	4.03E-24	0.91
S5	0.999999	5.07E-30	28.19
S6	0.999999	5.06E-36	473.68

The efficiency of Splitting over Permutation Monte Carlo seems to increase as the network reliability increases and also when the size of the network grows (accepting as a size measure, the number of links).

[2]A permutation is the order in which the links become operative in the network *Creation Process Model*, i.e. the order of the elements in sets like $\mathcal{T}$ in Section 4.

7 Conclusions

A well known Monte Carlo technique called Splitting, that was successfully employed to solve a variety of problems, most of them over Markovian models, was customized in this report to solve the source-terminal network reliability estimation by means of the Creation Process Model. The performance over a set of numerical experiments reveal a very high independence with respect to the estimated values. For the tested networks the relative error is almost invariant. Compared to the pure Crude Monte Carlo the speedup (precision gain) is huge, and compared to the Permutation Monte Carlo it is considerably larger in many cases.

The ideal number of thresholds seems to be slightly higher than the value of $-(\log \widehat{Q})/2$ proposed in the literature. However the method is robust in this feature and does not change the performance considerably at a variation on the number of thresholds around the proposed value.

An open problem derived from this report proposal is to try to find a relation between the best performance number of thresholds and the network parameters. It will be interesting also to implement and evaluate a *Fixed Splitting* variant.

Finally, as the results obtained in this work are very promising, specially by the precision robustness in the case of highly reliable networks, an in-depth comparative study against some other variance reduction schemes should be made.

References

[1] H. Cancela and M. El Khadiri. A recursive variance-reduction algorithm for estimating communication-network reliability. *IEEE Transactions on Reliability*, 44(4):595–602, December 1995.

[2] H. Cancela and M. El Khadiri. On the rvr simulation algorithm for network reliability evaluation. *IEEE Trans. Rel.*, 52(2):207–212, June 2003.

[3] Department of Operational Research Computing Science Institute (School of Engineering) of the Universidad de la República. Network Reliability – Repository. Website, 2008. `http://www.fing.edu.uy/inco/grupos/invop/graphtopologies/`.

[4] Jason L. Cook and Jose Emmanuel Ramirez-Marquez. Two-terminal reliability analyses for a mobile ad hoc wireless network. *Reliability engineering & systems safety*, 92(6):821–829, 2007.

[5] T. Elperin, I. B. Gertsbakh, and M. Lomonosov. Estimation of network reliability using graph evolution models. *IEEE Transactions on Reliability*, 40(5):572–581, Dec 1991.

[6] Susana Duarte Flores, Benjamín Barán Cegla, and Diana Benítez Cáceres. Telecommunication network design with parallel multi-objective evolutionary algorithms. In *Proceedings of the 2003 IFIP/ACM Latin America conference*, pages 1 – 11. ACM, 2003.

[7] M. J. J. Garvels. *The Splitting Method in Rare Event Simulation*. PhD thesis, Faculty of mathematical Science, University of Twente, The Netherlands, 2000.

[8] M. J. J. Garvels and D. P. Kroese. A comparison of RESTART implementations. In *Proceedings of the 1998 Winter Simulation Conference*, pages 601–609. IEEE Press, 1998.

[9] M. J. J. Garvels, D. P. Kroese, and J.-K. C. W. Van Ommeren. On the importance function in splitting simulation. *European Transactions on Telecommunications*, 13(4):363–371, 2002.

[10] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. Splitting for rare event simulation: Analysis of simple cases. In *Proceedings of the 1996 Winter Simulation Conference*. IEEE Press, 1996.

[11] Dilek Günnec and F. Sibel Salman. Assessing the reliability and the expected performance of a network under disaster risk. In *Proceedings of the International Network Optimization Conference, INOC 2007*, Spa, Belgium, 2007.

[12] K-P. Hui, N. Bean, M. Kraetzl, and Dirk Kroese. The cross-entropy method for network reliability estimation. *Annals of Operations Research*, 134:101–118(18), February 2005.

[13] C.H. Jun and S.M. Ross. System reliability by simulation: Random hazards versus importance sampling. *Probability in the Engineering and Informational Sciences*, 6, 1992.

[14] P. L'Ecuyer, V. Demers, and B. Tuffin. Rare-events, splitting, and quasi-Monte Carlo. *ACM Transactions on Modeling and Computer Simulation*, 17(2):Article 9, 2007.

[15] M. Lomonosov. On Monte Carlo estimates in network reliability. *Probability in the Engineering and Informational Sciences*, 8:245–264, 1994.

[16] J.S. Provan and M.O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12:777–788, 1983.

[17] S.M. Ross. A new simulation estimator of system reliability. *Journal of Applied Mathematics and Stochastic Analysis*, 7(3), 1994.

[18] M. Villén-Altamirano and J. Villén-Altamirano. Restart: A method for accelerating rare events simulations. In *Proceedings of the 13th International Teletraffic Congress*, pages 71–76. North-Holland, 1991.

[19] Hiroshi Wakabayashi. Network reliability improvement: Probability importance and criticality importance. In *Proceedings of the 2nd Symposium on Transportation Network Reliability (INSTR 2004)*, pages vol.3, pp. 204–210, Christchurch, New Zealand, 2004.

A Speedup vs. Number of Thresholds

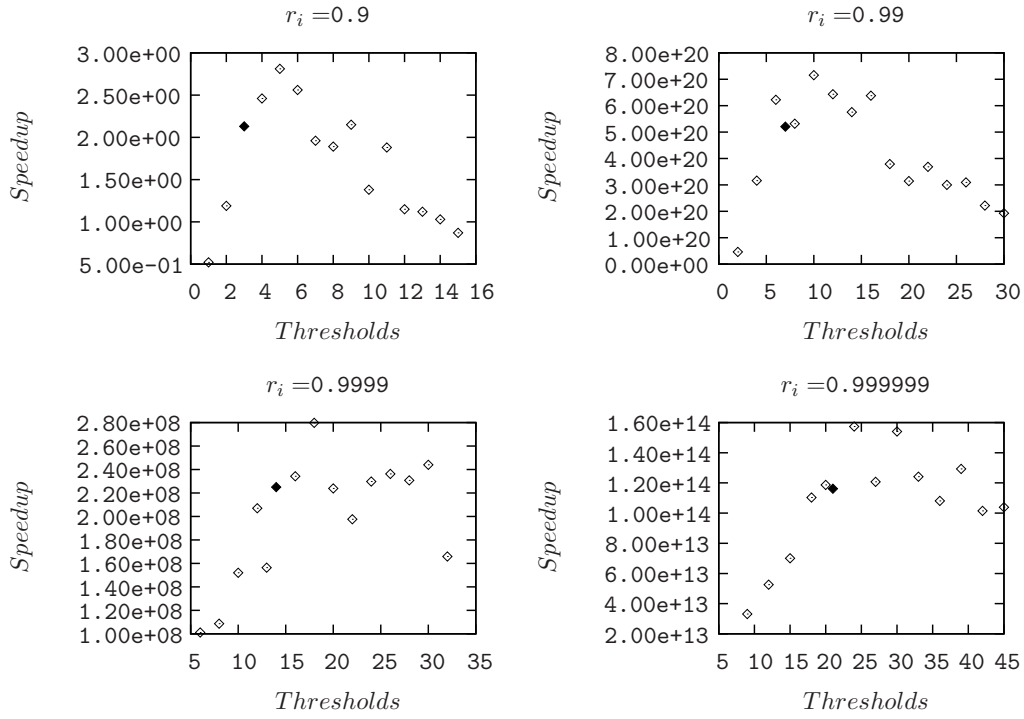

Figure 3: Dodecahedron Splitting–Crude Monte Carlo Speedup

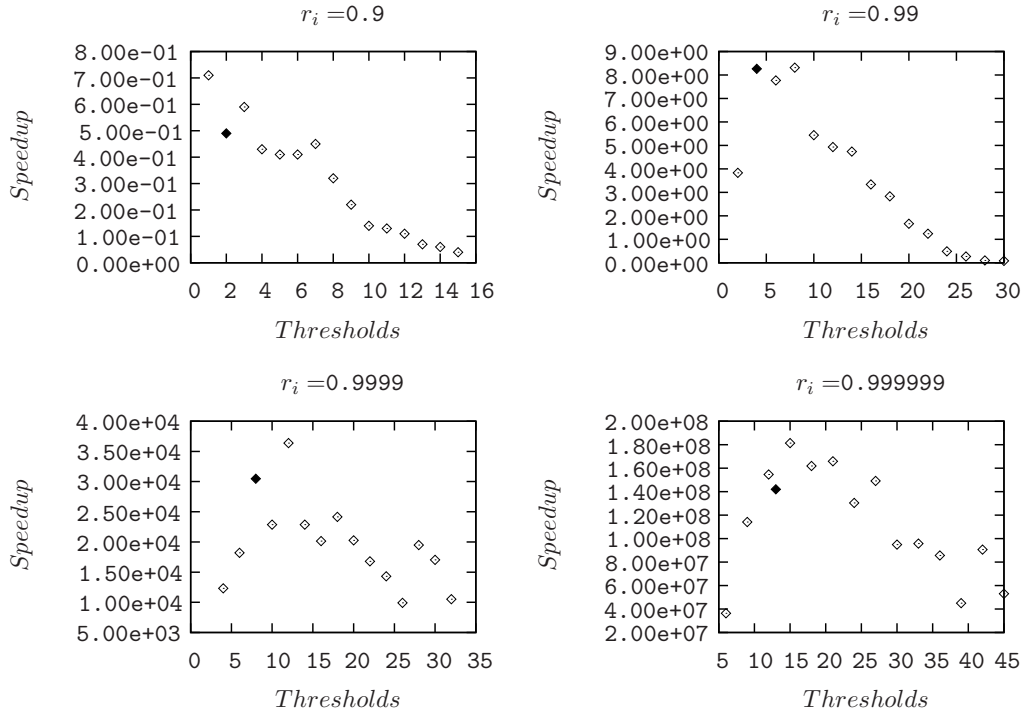

Figure 4: Arpanet Splitting–Crude Monte Carlo Speedup

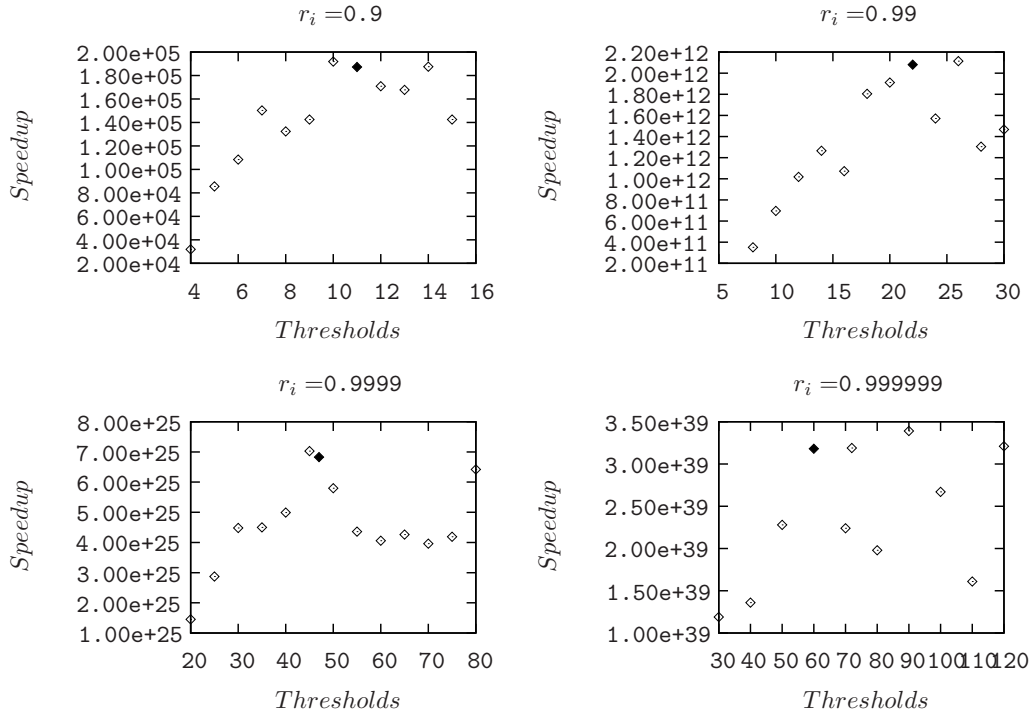

Figure 5: K10 Splitting–Crude Monte Carlo Speedup

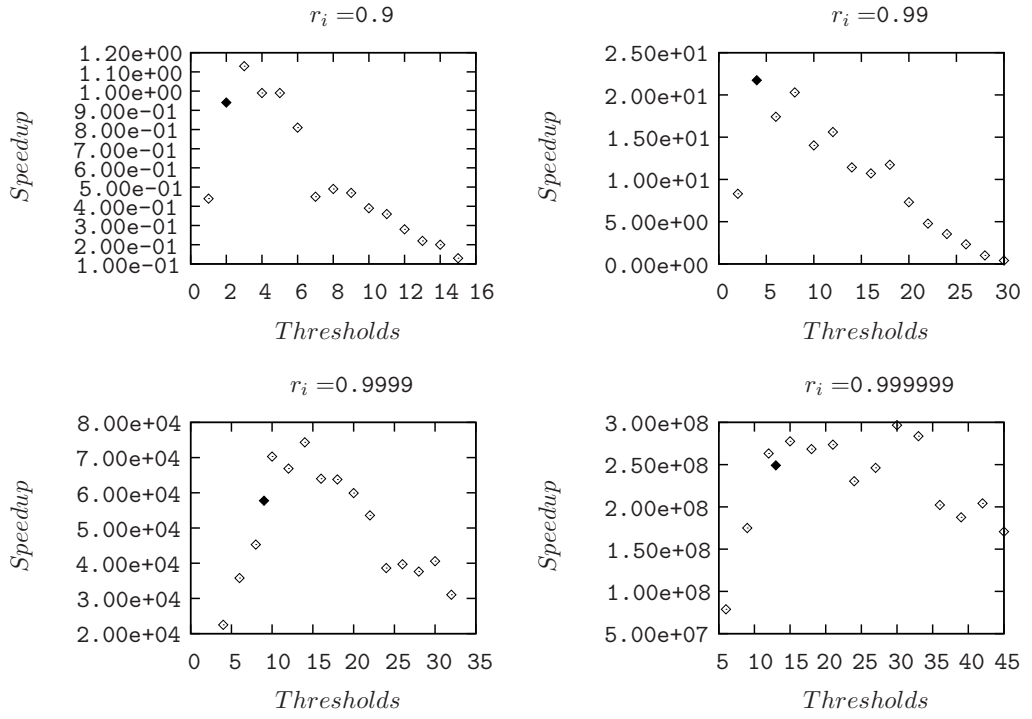

Figure 6: S2 Splitting–Crude Monte Carlo Speedup

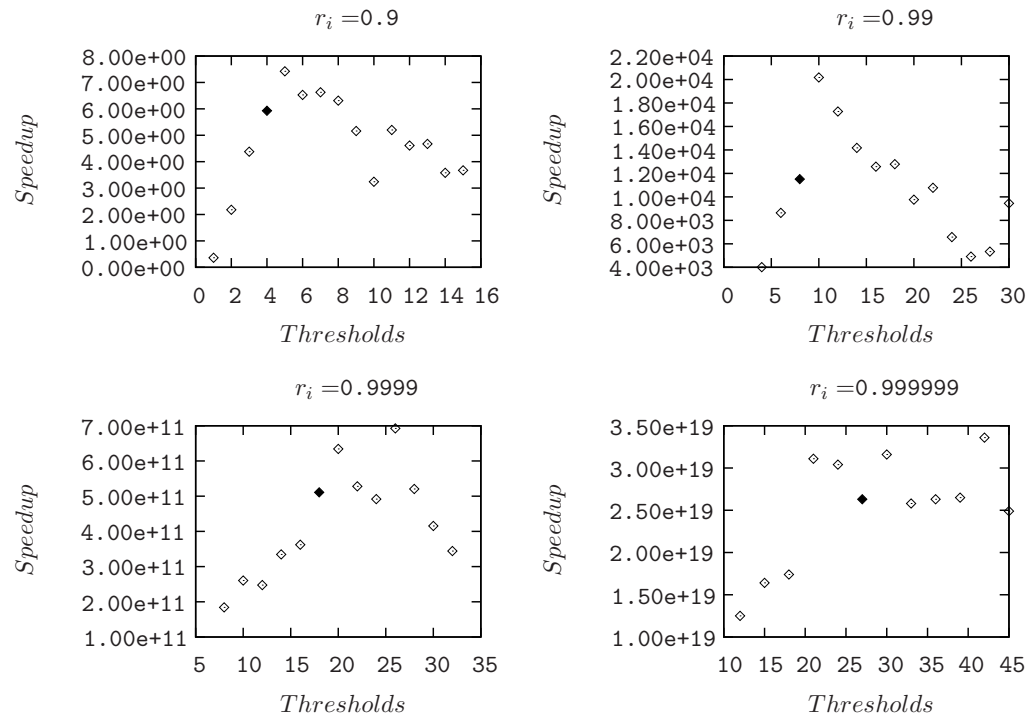

Figure 7: S4 Splitting–Crude Monte Carlo Speedup

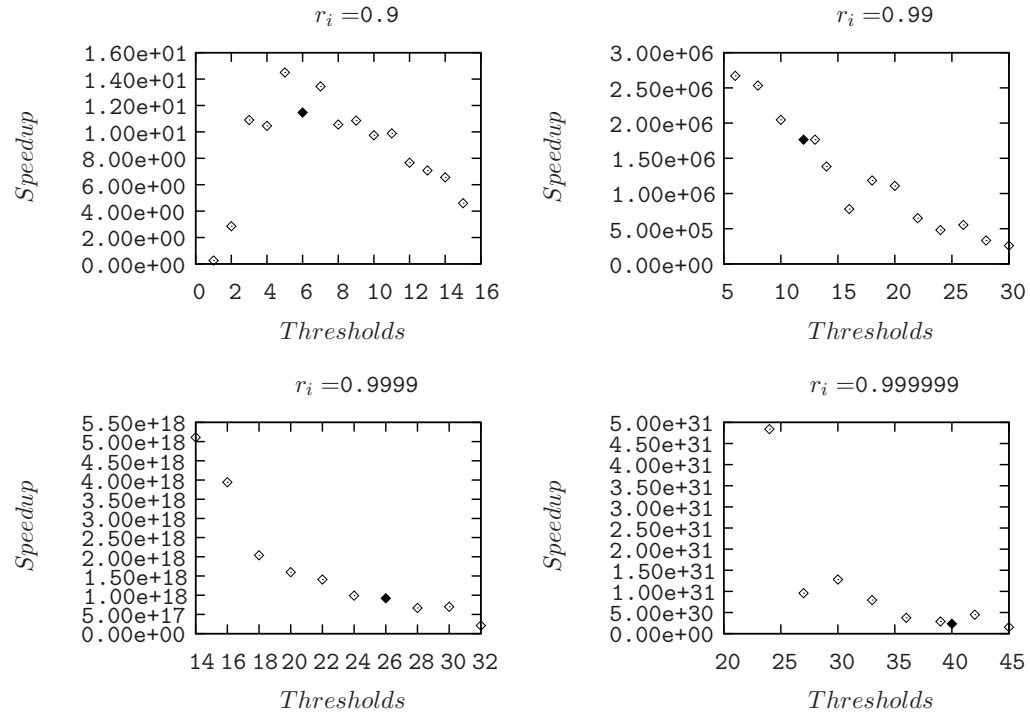

Figure 8: S6 Splitting–Crude Monte Carlo Speedup

B Relative Error vs. Effort

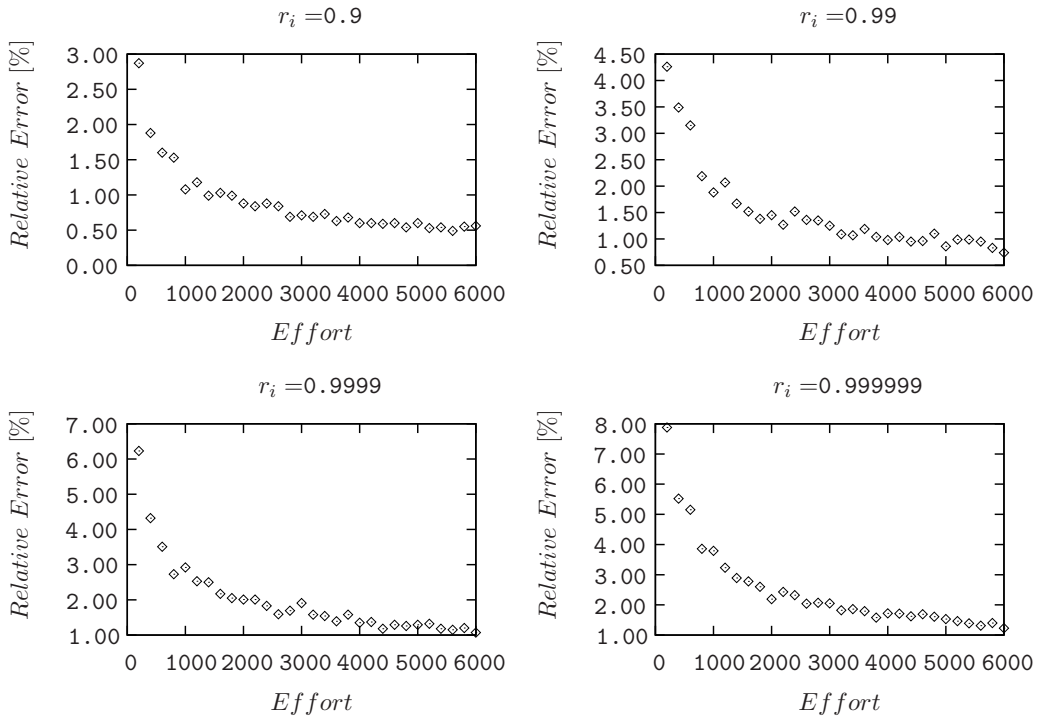

Figure 9: S2 Relative Error

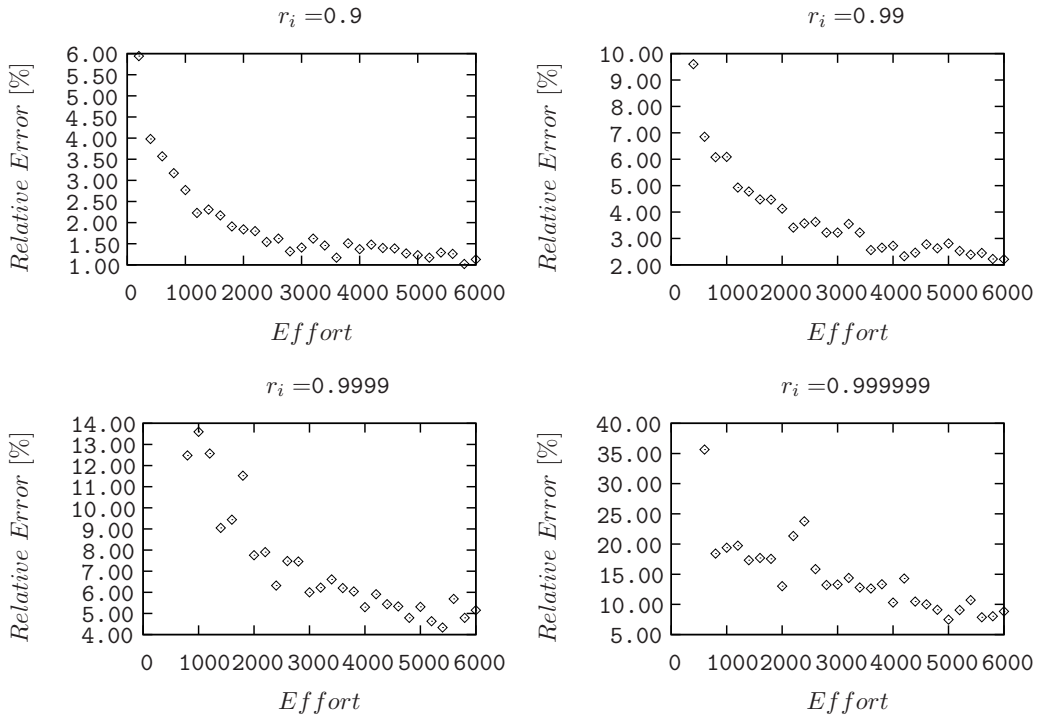

Figure 10: S6 Relative Error

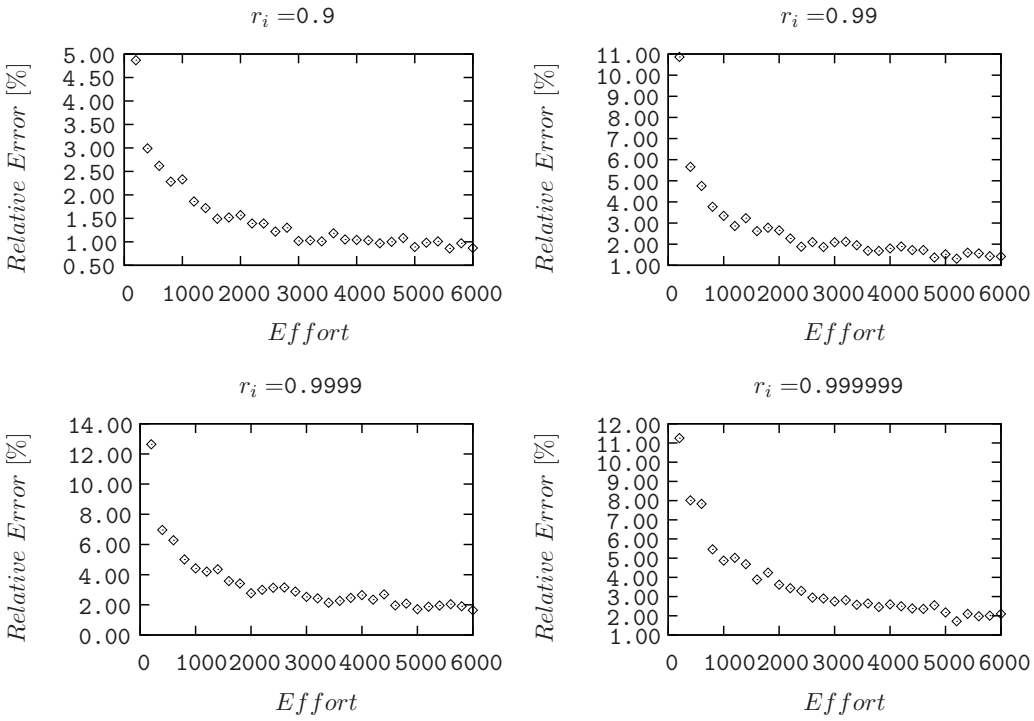

Figure 11: Dodecahedron Relative Error

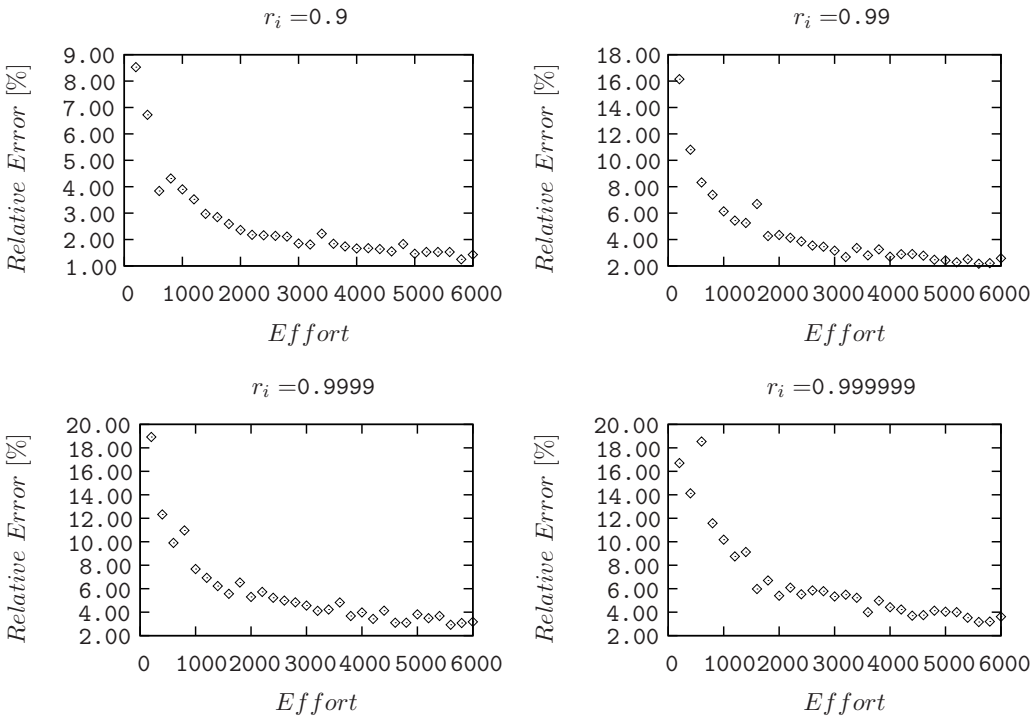

Figure 12: K10 Relative Error

C Relative Error vs. Number of Replications

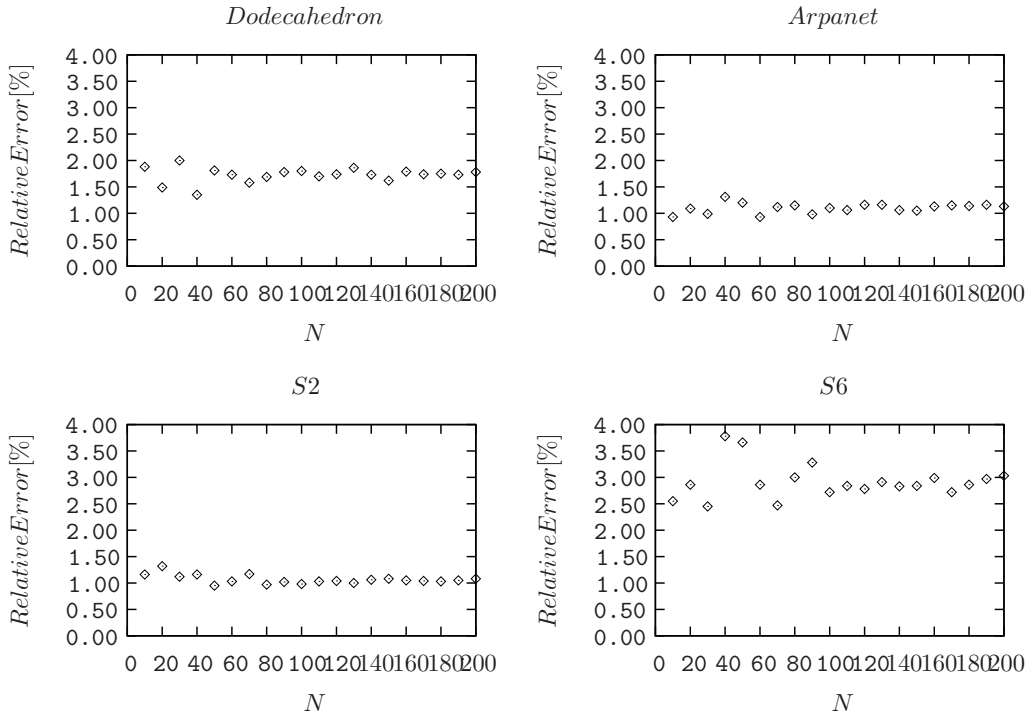

Figure 13: $N \times$ Effort = 400,000 $N < 200$

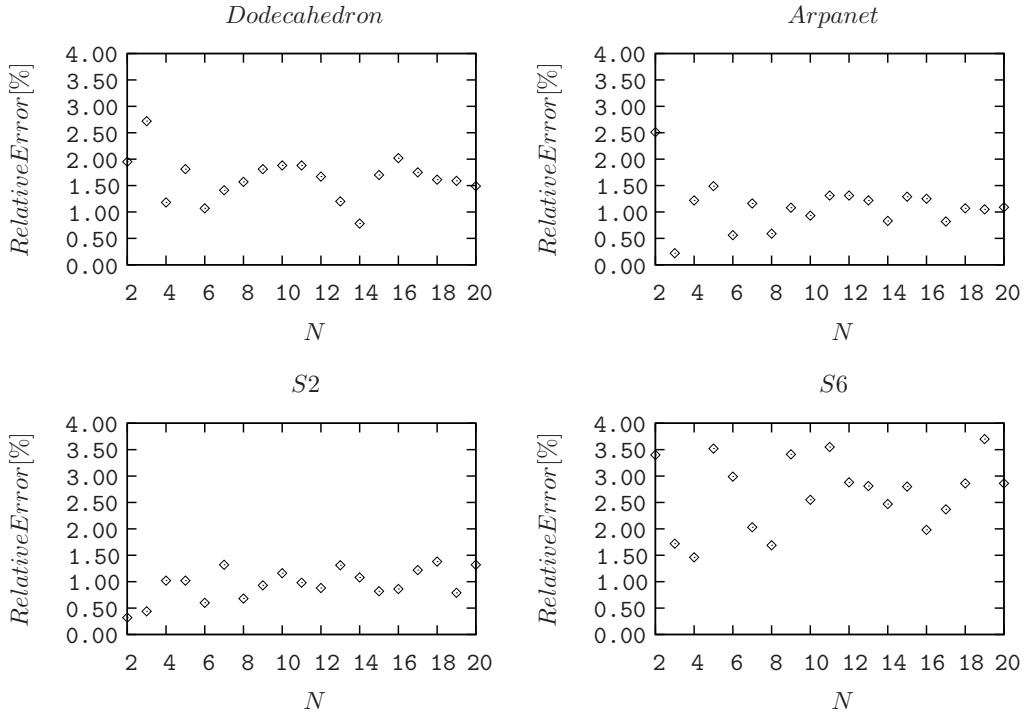

Figure 14: $N \times$ Effort = 400,000 $N < 20$

D Relative Error vs. Network Unreliability

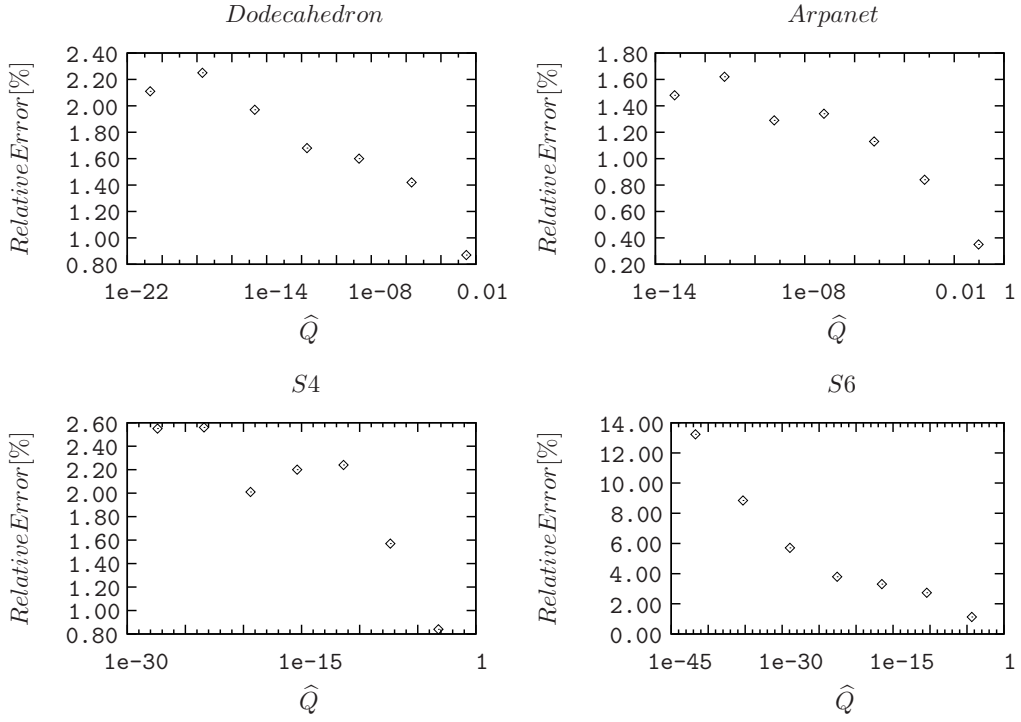

Figure 15: Relative Error vs. $\widehat{Q}$