The cross-entropy method with patching for rare-event simulation of large Markov chains

Bahar Kaynar* and Ad Ridder
Department Econometrics and Operations Research
Vrije University Amsterdam, Netherlands
email \{bkaynar,aridder\}@feweb.vu.nl

Abstract

There are various importance sampling schemes to estimate rare event probabilities in Markovian systems such as Markovian reliability models and Jackson networks. In this work, we present a general state dependent importance sampling method which partitions the state space and applies the cross-entropy method to each partition. We investigate two versions of our algorithm and apply them to several examples of reliability and queueing models. In all these examples we compare our method with other importance sampling schemes. The performance of the importance sampling schemes is measured by the relative error of the estimator and by the efficiency of the algorithm. The results from experiments show considerable improvements both in running time of the algorithm and the variance of the estimator.

Keywords: Cross-Entropy, Rare Events, Importance Sampling, Large-Scale Markov Chains.

1 Introduction

Markov chain modeling is an important tool that is used heavily in many Applied Probability and Stochastic Operations Research studies of systems such as communication, production, reliability, queueing, inventory, etc. (Dayar and Stewart, 2000; Stewart 2007). For realistic system modeling these Markov chains become enormously large with state spaces in the order

*Research supported by NWO grant 400-06-044.
of billions or even more. This large number of states that we observe in Markov chain modeling results from the exponential increase of the number of states with the number of state variables (the ‘curse of dimensionality’). The advantage of Markov chain modeling is found in its computable quantities by solving systems of linear equations, for instance for the stationary distribution, or for absorption probabilities. Clearly, when the state space is large enough, direct methods (e.g. Gauss elimination) are inefficient. Other numerical techniques have been developed and analyzed which are based mostly on iterative methods, projection techniques, and aggregation/disaggregation. The applicability and efficiency of a method depend strongly on the structure of the Markov chain under study (Dayar and Stewart 2000; Stewart 1999).

Our interest is the computation of rare event probabilities of extremely small order, say $10^{-9}$ or less. Rare events are found typically in applications such as reliability systems where the set of states in which the whole system fails, should almost never occur, or in queueing systems that model telecommunication technologies where loss of traffic packets should be unlikely. These rare events correspond to a set of absorption states under Markov chain modeling. In this paper we consider estimation of these absorbing probabilities by Monte Carlo simulation because we had the following arguments not to choose one of the numerical techniques that we mentioned earlier.

1. We need the probabilities in a few states only whereas the numerical methods compute these quantities in all states.

2. As the state space gets larger, it gets even harder to implement numerical methods. At this point advanced numerical techniques should be applied which falls beyond the scope of this paper.

3. We like to have fast and efficient algorithms.

4. We do not want to run the programs on supercomputers or grid networks. They should be implementable on personal computers, or laptops.

5. We do not know in advance the order of magnitude of the target probabilities, thus this makes the stopping criterion in iterative methods uncertain, or maybe unnecessary small.

Specifically our preference of using a ‘simple’ personal computer limits the size of the problem when applying numerical techniques such as Gauss-Seidel iteration. For the sake of argument,
let us assume that our computer has 4GB of physical RAM, and that the state-transition graph
of the Markov chain consists of \( n \) states and \( m \) transitions. Furthermore we assume that the
associated matrix of transition probabilities is sparse, i.e., \( m << n^2 \), which is not uncommon in
stochastic operations research systems. For ease of computing we set \( m = \rho n \), where \( \rho \) is a small
number. For instance, in a Jackson queueing network consisting of \( d \) queues there are at most
\( 2d + d^2 \) transitions out of any state (arrivals, departures, transfers). The matrix is stored as a
sparse matrix using information \((i, j, p_{ij})\) per transition, i.e., two integers (4 bytes each) and one
double (8 bytes). For the Gauss-Seidel iteration we need two vectors of size \( n \) (‘previous’ and
‘current’ solutions). Furthermore, we consider the standard IEEE number representations for
integers and doubles. Then an easy calculation gives us the maximal problem sizes as functions
of \( \rho \), see Table 1.

<table>
<thead>
<tr>
<th>( \rho )</th>
<th>( n )</th>
<th>( m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>51.20</td>
<td>204.80</td>
</tr>
<tr>
<td>8</td>
<td>28.44</td>
<td>227.56</td>
</tr>
<tr>
<td>12</td>
<td>19.69</td>
<td>236.31</td>
</tr>
<tr>
<td>16</td>
<td>15.06</td>
<td>240.94</td>
</tr>
<tr>
<td>20</td>
<td>12.19</td>
<td>243.81</td>
</tr>
</tbody>
</table>

Let us illustrate argument 5 of our list by considering a problem that concerns the computa-
tion of an absorption probability in a Markov chain related to a reliability model. In Section 5.1
we will give the details of the model, at this moment it suffices to know that we have computed
the probability by Gauss-Seidel iteration using software with arbitrary-precision arithmetic. The
Gauss-Seidel iteration is terminated when the Euclidean norm of the difference of two consecu-
tive solutions—called error—is less than some predefined \( \epsilon \). For this particular problem we have
set \( \epsilon = 10^{-50} \), and plotted the logarithms (base 10) of the errors up to termination, see Figure 1.
The absorption probability was computed from the solution vector after each iteration (see (1)),
and it was found that the first 16 significant digits remained constant after iteration 1000. Thus,
we are confident that the result is correct at termination. However, not knowing in advantage
the magnitude of the probability we could have easily set the stopping criterion too high. When
we used the standard 16-digit precision, the iteration terminated after 1000 iterations with a
‘zero’ error, albeit with the correct computed probability, although we were not confident about it.

Figure 1: Differences in two consecutive solutions of the Gauss-Seidel iteration.

Since we deal with rare event probabilities of extremely small order, we execute the simulation under an importance sampling measure. In many problems involving rare events, importance sampling has provided efficient estimators with huge variance reduction and substantial running time improvement, see the recent surveys Bucklew 2004; Juneja and Shahabuddin 2006; Rubino and Tuffin 2009. The challenge of importance sampling is to come up with a biasing scheme or change of measure that indeed gives these improvements. There is substantial literature available on importance sampling for rare event problems in Markov chains, both in a more generic setting, and in specific models. Concerning the generic setting, the basic idea is to approximate the zero-variance change of measure which is theoretically known but practically not implementable (Glynn and Iglehart 1989). We mention (Bucklew et. al. 1990) who applied large deviations techniques, (Dupuis and Wang 2005) who considered the importance sampling problem as a stochastic game, (Ahamed et. al. 2006; Randhawa and Juneja 2004) who considered the rare event problem as an expected average reward in a regenerative Markov chain and applied stochastic approximation techniques. Several adaptive techniques have been proposed which attempt to learn the zero-variance change of measure for Markov chains by an iterative procedure: a number of sample paths is generated and based on the ‘scores’ of these paths the current change of measure is updated (Desai and Glynn 2001; Kollman et. al. 1999).
Many more research studies have been carried out to construct importance sampling algorithms that are tailor-made for specific problems, we refer to the surveys Bucklew (2004), Heidelberger (1995), Juneja and Shahabuddin (2006), Rubino and Tuffin (2009) for some details. Our paper has the intention to propose an importance sampling algorithm that is generally applicable to rare events in multi-dimensional Markov chains, although our examples will be taken from queueing and reliability. Our algorithm builds on the cross-entropy method (Rubinstein and Kroese 2004) which we will describe in more detail in Section 2.4. At this point it suffices to understand that the cross-entropy method updates iteratively the change of measure by maximizing a parameterized nonlinear program, where the parameters are part of the probability distribution functions that determines the statistical law of the system. Since we deal with Markov chains, the optimal change of measure assigns new probabilities to all transitions that have initially non-zero probability (Glynn and Iglehart 1989, L’Ecuyer and Tuffin 2007). Thus, we choose specifically the transition probabilities of the Markov chain as the parameters in the cross-entropy optimization program. This approach runs into numerical problems (efficiency, memory, round-off errors) when there are too many parameters, i.e., when we deal with large Markov chains.

To tackle the problem of too many parameters we propose to partition the state space (and their associated transitions) in smaller parts, and then apply the cross-entropy iterations on these smaller sets. We consider two versions of this approach. In the first version we let each transition probability be a parameter that is to be determined optimally by the cross-entropy method. Once the probabilities of a part of the state space are determined, we keep them fixed in the cross-entropy iterations on subsequent iterations. This is the general outline of the patching algorithm which will be given in more detail in Section 3. The second version assumes a Markov chain with a face-homogeneous structure such as we encounter typically in Jackson queueing networks (Ignatiouk 2005). When we keep this structure in the change of measure, the number of parameters is already reduced to a tractable size. Then we apply our patching algorithm but now we do not keep the parameters fixed while iterating subsequent parts of the state space. Other attempts of reducing the parameter space in the cross-entropy method has been reported in de Boer and Nicola (2002). The authors use also a grouping idea, however they group states along boundary layers that depend on the contents of the queues.

The rest of the paper is organized as follows. In Section 2 we define the type of rare events
that we shall consider in this paper, we provide a general explanation of importance sampling simulation, and we describe the classical cross-entropy method for Markov chains followed by the performance measures to compare importance sampling estimators. In Section 3 we present our patching algorithm that forms the basis of our study. Section 4 contains the face-dependent methods, and Section 5 shows results of our methods applied to reliability models and to Jackson queueing networks.

2 Preliminaries

In this section we will briefly discuss our problem of interest, the main ideas behind importance sampling (IS) simulation and specifically the cross-entropy (CE) method. For more details and review we refer to Rubino and Tuffin (2009) and Rubinstein and Kroese (2004).

2.1 The rare event problem

Let \( (X(t))_{t=0}^{\infty} \) denote a discrete-time Markov chain on a finite but large state space \( \mathcal{X} \) with underlying probability space \( (\Omega, \mathcal{A}, \mathbb{P}) \) such that the chain has stationary transition probabilities and is irreducible over its state space. We assume that a family of rare events \( \{A_n\} \subset \mathcal{A} \) is defined on the state space and parameterized by \( n \) such that \( \lim_{n \to \infty} \mathbb{P}(A_n) = 0 \). Let \( P = (p(x, y))_{x,y \in \mathcal{X}} \) signify the matrix of transition probabilities, where \( p(x, y) = \mathbb{P}(X(t+1) = y | X(t) = x) \). Actually, we allow two versions, the first version being that the rarity parameter \( n \) is associated with the problem size which is increasing in \( n \); in the second version we assume a constant problem size and that the rarity parameter is associated with those transition probabilities that are decreasing in \( n \). In the latter case we should have denoted the probability measure by \( \mathbb{P}_n \), and the transition probabilities by \( p_n(x, y) \). It is immediately clear in a specific problem which version applies, and thus to avoid too much detailed notation we drop conveniently the rarity parameter \( n \) whereever possible.

We restrict to rare events \( A \) in Markov chains that are given by hitting times to absorption sets of the state space. Consider a partition of the state space into three sets: \( \mathcal{G} \) is a small set of ‘good’ states, \( \mathcal{F} \) is a small set of ‘failed’ or ‘bad’ states, and the other states form the large set of ‘internal’ states \( \mathcal{T} = \mathcal{X} \setminus (\mathcal{G} \cup \mathcal{F}) \). For each internal state \( x \in \mathcal{T} \) let \( \gamma(x) \) be the probability that the Markov chain will hit the failure set before the good set when the chain starts in state \( x \).
x. For ease of notation we assume that the good set consists of a single state, denoted by 0, and that the Markov chain jumps immediately out of it, i.e., \( p(0, 0) = 0 \). Then typically we are interested in the probability that when the chain starts in the good state 0, it will hit the failure set \( \mathcal{F} \) before returning to 0. The associated event is \( A = 1\{X(T) \in \mathcal{F}\} \) with hitting time

\[
T = \inf\{t > 0 : X(t) \in \{0\} \cup \mathcal{F}\}.
\]

The probability \( \mathbb{P}(A) = \mathbb{P}(X(T) \in \mathcal{F}) \) will be denoted also by \( \gamma(0) \). Notice that

\[
\gamma(0) = \sum_{x \in T} p(0, x) \gamma(x).
\] (1)

2.2 Importance sampling

Let \( Y \) be the unbiased crude Monte Carlo estimator of \( \mathbb{P}(A) \) obtained by averaging i.i.d. replications of \( 1\{A\} \) (the indicator function of the rare event \( A \)) which are generated under the original probability \( \mathbb{P} \):

\[
Y = \frac{1}{N} \sum_{i=1}^{N} 1\{X^{(i)}(T) \in \mathcal{F}\}.
\]

Using standard statistical techniques one shows that the required sample size to obtain a confidence interval with a predefined accuracy (such as “within 10% relative to the estimate”), is of the order of the reciprocal of \( \mathbb{P}(A) \) (Juneja and Shahabuddin 2006). Since we assume \( \mathbb{P}(A) \to 0 \), a large simulation effort is needed.

An alternative way is to apply importance sampling simulation. Although importance sampling simulation can be applied with various changes of measures, within this work we restrict the change of measure \( \mathbb{P}^* \) so that we retain a Markov chain, say with a matrix of transition probabilities \( P^* \). We restrict even further by demanding for the individual transition probabilities to satisfy \( p^*(x, y) > 0 \) if and only if \( p(x, y) > 0 \), which is sufficient for the absolute continuity requirement \( 1\{A\} d\mathbb{P} \ll d\mathbb{P}^* \). We denote by \( \mathcal{P} \) the set of all feasible matrices of transition probabilities with this property.

Let \( Y^* \) be an unbiased importance sampling estimator obtained by averaging i.i.d. replications of \( 1\{A\} \) generated under the change of measure, and weighting by the likelihood ratio of a random sample path \( X \), denoted by \( L(X) = (d\mathbb{P}/d\mathbb{P}^*)(X) \):

\[
Y^* = \frac{1}{N} \sum_{i=1}^{N} 1\{X^{(i)}(T) \in \mathcal{F}\} \frac{d\mathbb{P}}{d\mathbb{P}^*}(X^{(i)}),
\]
where \(X^{(i)}\) stands for the \(i\)-th sample path. The aim is to find an optimal change of measure, or equivalently choosing a transition probability matrix \(P^*\) among all feasible matrices of transition probabilities \(P\) which minimizes the variance.

The optimal change of measure to estimate \(P(A)\) would be (Glynn and Iglehart 1989)

\[
d_{\mathbb{P}^{\text{opt}}} (X) = \frac{1 \{ A \} d\mathbb{P}(X)}{\mathbb{P}(A)},
\]

which would result in a zero-variance importance sampling estimator. Clearly, it is not implementable since it requires the unknown probability.

### 2.3 The performance measures

The performance of an importance sampling estimator is measured commonly through its relative error, the exponential decay rate of its second moment, and its effort (Asmussen and Rubinstein, 1995; Heidelberger, 1995). These three performance measures of the importance sampling estimator with rarity parameter \(n\) are

\[
\text{RE} [Y^*_n] = \frac{\sqrt{\text{Var}_n [Y^*_n]}}{\mathbb{E}_n [Y^*_n]},
\]

\[
\text{RAT} [Y^*_n] = \frac{\log \mathbb{E}^*_n [(Y^*_n)^2]}{\log \mathbb{E}^*_n [Y^*_n]},
\]

\[
\text{EFF} [Y^*_n] = - \log \left( \frac{\text{Var}_n [Y^*_n]}{\text{CPU}[Y^*_n]} \right),
\]

where CPU means the running time of the importance sampling algorithm on our computer. Notice that we take here expectations w.r.t. the probability measure \(\mathbb{P}^*_n\). Clearly, our target is to obtain small variances (relative to the mean), which is expressed by a small relative error \(\text{RE}\). As mentioned above in Section 2.2, the zero-variance estimator associated with the optimal change of measure \(\mathbb{P}^{\text{opt}}_n\), (2), is not implementable. A ‘second best’ estimator would have bounded relative error, one says that the estimator is strongly efficient,(L’Ecuyer et. al. 2008): \(\lim \sup_{n \to \infty} \text{RE}[Y^*_n] < \infty\). It would mean that the number of samples required to achieve a fixed relative error is constant for all \(n\). Finding an importance sampling estimator with this property is in most cases a hard problem, and thus one seeks usually an asymptotically optimal, or logarithmically efficient estimator (L’Ecuyer et. al. 2008): \(\lim \inf_{n \to \infty} \text{RAT}[Y^*_n] \geq 2\). It would mean that, assuming that the rare event probabilities \(\mathbb{P}(A_n)\) decay exponentially fast, the required sample sizes grow at most polynomially. Finally, it is more expensive to obtain a sample using importance sampling than standard simulation. However substantial reduction in variance
generally compensates for the increase in computational time. We include the EFF-measure which takes into account both the computing time of the importance sampling algorithm and the variance to compare the actual efficiency improvement. The $-\log$ is included in the formula only for getting convenient numbers for ease of comparison. An alternative decreasing function that is found in the literature $1/(\text{Var}_n[Y^*_n] \times \text{CPU}[Y^*_n])$, which gives equivalent conclusions.

2.4 The cross-entropy method

It has been shown by Glynn and Iglehart (1989) that the zero-variance change of measure (2) for our Markov chain problem is obtained by assigning transition probabilities

$$p^\text{opt}(x, y) = p(x, y) \frac{\gamma(y)}{\gamma(x)}$$

(3)

to the Markov chain. This expression contains the unknown hitting probabilities $\gamma(\cdot)$, however, it suggests to consider a technique that provides for a matrix of transition probabilities $P^*$ that approximates the optimal matrix $P^\text{opt}$. This is exactly what the cross-entropy method is doing. Recall that the relative entropy or Kullback-Leibler distance between two probability measures $\mu$ and $\nu$ is defined by

$$D_{KL}(\nu \| \mu) = \int \log \frac{d\nu(x)}{d\mu(x)} d\nu(x),$$

assuming the absolute continuity $\nu \ll \mu$, see, e.g., Section 2.3 in Cover and Thomas (2006). Hence, the cross-entropy method proposes to minimize the Kullback-Leibler distance of the optimal measure $P^\text{opt}$ from a family of implementable changes of measure $P^*$. Since these implementable changes of measure are associated with the feasible set $\mathcal{P}$ of matrices of transition probabilities, we get the optimization program

$$\inf_{P^* \in \mathcal{P}} D_{KL}(P^\text{opt} \| P^*) = \inf_{P^* \in \mathcal{P}} \mathbb{E}_{P^\text{opt}} \left[ \log \left( \frac{dP^\text{opt}}{dP^*}(X) \right) \right]$$

$$= \inf_{P^* \in \mathcal{P}} \mathbb{E} \left[ \frac{dP^\text{opt}}{dP}(X) \log \left( \frac{dP^\text{opt}}{dP^*}(X) \right) \right].$$

Substitute the expression (2) of the optimal change of measure, and manipulate the resulting formula in a straightforward manner to get the optimization program

$$\sup_{P^* \in \mathcal{P}} \mathbb{E}[1\{X(T) \in \mathcal{F}\} \log dP^*(X)].$$

(4)

Notice that the sample path probability $dP^*(X)$ is a product:

$$dP^*(X) = \prod_{t=1}^{T} p^*(X(t-1), X(t)),$$

(5)
and notice that we take expectations with respect to the original probability measure $\mathbb{P}$. There are several ways of attacking the cross-entropy optimization program (4), for instance, solving the first order conditions. However, in almost all cases the program is not analytically nor numerically tractable, and thus one resorts to simulation. Again, several simulation techniques may be applied, for instance stochastic approximation, however, it turns out that sample path optimization is most convenient; i.e., one considers the stochastic counterpart of (4), see Rubinstein and Kroese (2004). Since the program involves the rare event, we first apply a change of measure:

$$
\mathbb{E}[1\{X(T) \in \mathcal{F}\} \log d\mathbb{P}^*(X)] = \mathbb{E}^{(0)} \left[ \frac{d\mathbb{P}}{d\mathbb{P}^{(0)}} 1\{X(T) \in \mathcal{F}\} \log d\mathbb{P}^*(X) \right].
$$

This is done for a probability measure $\mathbb{P}^{(0)}$ such that (i) the Markov property is retained; (ii) the associated matrix of transition probabilities is feasible $P^{(0)} \in \mathcal{P}$; (iii) the set $\mathcal{F}$ is ‘not so’ rare under $\mathbb{P}^{(0)}$. The program (6) is solved iteratively by estimation: let $X^{(1)}, \ldots, X^{(N_1)}$ be i.i.d. sample paths of the Markov chain generated by simulating the states according to a matrix of transition probabilities $P^{(j)}$ until absorption in the good or bad states, then we calculate for $j = 0, 1, \ldots$

$$
P^{(j+1)} = \arg \max_{P^* \in \mathcal{P}} \frac{1}{N_1} \sum_{i=1}^{N_1} \frac{d\mathbb{P}}{d\mathbb{P}^{(0)}} (X^{(i)}) 1\{X^{(i)}(T) \in \mathcal{F}\} \log d\mathbb{P}^*(X^{(i)}).
$$

We repeat this ‘updating’ of the change of measure a few times until the difference $P^{(j+1)} - P^{(j)}$ is small enough (in some matrix norm).

It is straightforward to solve the optimization program (7) by considering its first order conditions, and using the product form expression of the sample path probability (5). Denote by $N^{(i)}(x,y)$ the number of times that the $i$-th sample path of the Markov chain makes the transition from state $x$ to state $y$, and denote by $L^{(i)}$ the likelihood ratio of the $i$-th sample path.

Then we get for the individual entries

$$
p^{(j+1)}(x,y) = \frac{\sum_{i=1}^{N_i} L^{(i)} 1\{X^{(i)}(T) \in \mathcal{F}\} N^{(i)}(x,y)}{\sum_{i=1}^{N_i} L^{(i)} 1\{X^{(i)}(T) \in \mathcal{F}\} \sum_{y \in \mathcal{X}} N^{(i)}(x,y)}.
$$

Clearly, we might get a zero update for a transition $x \to y$ which is not observed in the sample but which has originally a positive probability $p(x,y)$. To overcome this problem it is suggested
to apply a weighting combination by Rubinstein and Kroese (2004); Section 4.2, i.e., denote by $p^{(j+1)}(x, y)$ the expression of (8), then
\[ p^{(j+1)}(x, y) = \alpha p(x, y) + (1 - \alpha)p^{(j+1)}(x, y), \]
for some $\alpha \in (0, 1)$.

3 Adaptation of the cross-entropy method by patching

In Section 2.4 we described the cross-entropy method for Markov chains where each transition is taken as a parameter to be updated by the formulas (8) and (9). Since in this work Markov chains with large state space will be under consideration, under the cross-entropy setting we will have a large number of parameters to update. For example let us think of a two-dimensional birth-death process with unit jumps. Typically, it models reliability systems as we will see in Section 5. Such systems may easily contain a few millions states, say two million, which means that there are about eight million transitions, i.e., parameters, to update in every iteration of the cross-entropy iteration (8). Such high-dimensional parameter estimation suffers from degeneracy (p. 249 in Rubinstein and Kroese 2007) and creates numerical problems for obtaining accurate estimates unless we take a large sample size which we do not want. Notice that for generating sample paths of the Markov chain, and for keeping track of which transitions have occurred, we have to store only an array of the non-zero transition probabilities, and the associated array $N(x, y)$, see the update formula (8). Sizes of up to hundreds of millions of non-zero transitions do not cause storage problems in modern PC-computers as we discussed in the Introduction.

In this section we consider an adaptation of the standard cross-entropy method that reduces the number of parameters in each iteration. The main idea of our method is to implement the iteration rule (7) recursively in small portions of the state space, called patches. In the next sections we describe our patching algorithm, and we discuss some implementation issues and comparisons with other importance sampling techniques for Markov chains.

3.1 The patching algorithm

We start with a small irreducible subset of the state space containing the good state $0$. We specify some of its states as being ‘bad’, in the sequel we shall indicate how to choose these. Being a small subset of the state space we call it a patch. Then we restrict the Markov chain
to this patch and apply the cross-entropy iteration (7) for finding a change of measure for the transition probabilities in the patch. These updated transition probabilities are kept fixed for the rest of the algorithm. Next, we enlarge the patch and apply the same procedure to the parameters which are introduced newly in the expanded patch. We continue enlarging the patch and updating the parameters until the size of the patch matches the original state space. The construction of the patches should be done in such a manner that the Markov chain restricted to a patch remains irreducible.

As a simple example, consider a two-dimensional Markov chain on

\[ \mathcal{X} = \{0, 1, \ldots, 10\}^2 = \{x = (x_1, x_2) : x_1, x_2 \in \{0, 1, \ldots, 10\}\}, \]

with transitions \( \pm e_1 \) and \( \pm e_2 \) (whenever possible) where \( e_1 = (1, 0) \) and \( e_2 = (0, 1) \). The state space is partitioned into the three sets as mentioned in Section 2.1: the good state \( 0 = (0, 0) \), the bad set \( \mathcal{F} = \{(10, 10)\} \), and the internal set \( \mathcal{T} \) of all other states, see Figure 2.

![Figure 2: The state space of the simple example](image)

Let the first patch be Patch\(_1\) = \(\{0, 1, 2, 3, 4\}^2\), which contains the good state 0. Notice that Patch\(_1\) is proportional to the state space \(\mathcal{X}\) by the same reduction factor in both dimensions. Thus it is natural to let \(\mathcal{F}_1 = \{(4, 4)\}\) be the bad set in Patch\(_1\). Next, we need to define the matrix of transition probabilities \(P_1\) by restricting \(P\) to Patch\(_1\). Define the “border states”

\[ B_1 = \{(4, x_2) : 0 \leq x_2 \leq 4\} \cup \{(x_1, 4) : 0 \leq x_1 \leq 4\}. \]
We keep $p_1(x, y) = p(x, y)$ for all transitions $x \to y$ when $x$ is not a border state. For the border states we renormalize the probabilities of transitions to states in $\mathcal{P}_{1}$:

$$p_1(x, y) = \frac{p(x, y)}{\sum_{z \in \chi} p(x, z)}, \quad x \in \mathcal{B}_1, \text{ and } y, z \in \mathcal{P}_{1}.$$ 

Finally, to start off the cross-entropy iterations with a matrix of transition probabilities $P_1^{(0)}$ such that the bad set $\mathcal{F}_1$ is less rare than simulating with matrix $P_1$, we take the uniform transition probabilities, i.e.,

$$\begin{align*}
p_1(x, y) > 0 & \iff P_1^{(0)}(x, y) = \frac{1}{\# \text{ of transitions out of state } x \text{ in } \mathcal{P}_{1}}.
\end{align*}$$

Then by using expressions (8) and (9), we update the transition probabilities for the change of measure a few times. The resulting change of measure $p_1^*(x, y)$ for transitions out of $x \in \mathcal{P}_{1} \setminus \mathcal{B}_1$ are stored permanently, and will be used both in the subsequent steps of the algorithm, and in the final matrix of transition probabilities $P^*$ on the whole state space. We denote the set of these transitions by $\mathcal{U}$ and we call them permanent transitions.

After dealing with $\mathcal{P}_{1}$, we enlarge it to

$$\mathcal{P}_{2} = \{ x = (x_1, x_2) : 0 \leq x_1 \leq 6, 0 \leq x_2 \leq 6 \}.$$

The bad set becomes $\mathcal{F}_2 = \{6, 6\}$. We get the matrix of transition probabilities $P_2$ on $\mathcal{P}_{2}$ by restricting the original matrix $P$ in the same manner as above: $p_2(x, y) = p(x, y)$ for all transitions $x \to y$ when $x$ is not a border state $(6, x_2)$ or $(x_1, 6)$. For the border states we renormalize. Now consider the matrix $P_2^{(0)}$ to start off the cross-entropy iterations on $\mathcal{P}_{2}$. We let it be the probabilities determined in $\mathcal{P}_{1}$ for the permanent transitions, and otherwise they are uniform:

$$p_2^{(0)}(x, y) = \begin{cases} 
p^*(x, y) & \text{for } (x \to y) \in \mathcal{U}, \\
\frac{1}{\# \text{ of transitions out of state } x \text{ in } \mathcal{P}_{2}} & (x \to y) \notin \mathcal{U} \text{ and } p(x, y) > 0,
\end{cases}$$

Alternatively, also for the states $x \in \mathcal{B}_1 \setminus \mathcal{F}_1$ in the (previous) border set (except the bad states), we might use the transition probabilities $p_1^*(x, y)$ (where $y \in \mathcal{P}_{1}$) and renormalize these after adding $p(x, y)$ (where $y \in \mathcal{P}_{2} \setminus \mathcal{P}_{1}$). In our implementations we apply the latter.

Having set the initial matrix $P_2^{(0)}$ and the transitions whose probabilities need to be updated, we apply again the cross-entropy method for finding a change of measure $P_2^*$ on $\mathcal{P}_{2}$. Recall
that we keep $p_2^{(j)}(x, y) = p_1^*(x, y) = p^*(x, y)$ for all transitions $(x \to y) \in \mathcal{U}$ during the cross-entropy iterations, thus the updating rule (8) is applied to the other transitions. With the resulting matrix $P_2^*$ we update the permanent set $\mathcal{U}$ by adding the probabilities $p_2^*(x, y)$ of transitions $(x \to y)$ that were not permanent and for which $x \in \text{Patch}_2 \setminus B_2$ (clearly, $B_2$ is the set of border states of Patch$_2$).

This procedure is repeated by enlarging Patch$_2$ to Patch$_3$, and later on to Patch$_4$ which is the whole state space $\mathcal{X}$, see Figure 3. After these steps we have found our change of measure $P^*$ for running the importance sampling simulations and estimating the original rare event probability.

Figure 3: The steps in the patching algorithm.

Summarizing, the general cross-entropy patching algorithm reads as follows.

**Algorithm 1.**

1. Set $i = 1$.

2. Choose Patch$_i$.

3. Form $P_i^{(0)}$ for Patch$_i$.

4. Generate $N_1$ sample paths and apply the updating formulae (8) and (9).

5. Stop when convergence has been obtained, otherwise set $j = j + 1$ and go to step 3.

6. Stop when Patch$_i$ matches with the original state space, otherwise set $i = i + 1$ and go to step 2.

7. Estimate the rare event probability $\mathbb{P}(A)$ by using importance sampling with $N$ sample paths.
3.2 Implementation issue: how to choose the parameters?

In order to apply Algorithm 1, we need to decide on the values of its ingredients such as (A) the shapes and sizes of the patches, (B) the sample sizes \( N_1 \) in the iterations, and (C) the weighting factor \( \alpha \) in the updating rule (9). In this section we will concentrate on these issues. For visualization let us have a look at a small two-dimensional reliability system of two types with 12 components each (see Section 5 for more details on this application). The system is modeled by a two-dimensional birth-death process with unit jumps. The state space is

\[
\mathcal{X} = \{0, 1, \ldots, 12\}^2 = \{x = (x_1, x_2) : x_1, x_2 \in \{0, 1, \ldots, 12\}\}.
\]

Let the birth rates be \( \lambda_1(x) = (12 - x_1)\varepsilon \) and \( \lambda_2(x) = (12 - x_2)5\varepsilon \), and the death rates are \( \mu_i(x) = 1 \) for \( i = 1, 2 \). The failure set is the single state \( \mathcal{F} = \{(12, 12)\} \).

(A) Shape and size of patches; choice of bad states.

In order for importance sampling simulation to give more accurate estimates, the Markov chain structure obtained by the change of measure should not be so much different from the actual structure of the problem, Andradóttir et. al. (1995). This suggests to imitate in each patch the actual shape of the state space as well as the definition of the failure sets. If in the original problem, the failure state is defined as the state where all components of all types are down, then this definition is preserved for each patch. In this section we assume these choices. In the next section we shall discuss alternative versions.

Additionally, independent of the number of patches we would like to implement, we observed that choosing the patches equidistant from each other gave the best results in terms of relative error. This may be explained by an argument that one uses also in the splitting method for rare event simulations (more about splitting in Section 3.3): let \( p_i \) be the conditional probability that a random sample path will hit the boundary of Patch\(_i\), given that it has reached the boundary of Patch\(_{i-1}\), then we observed that the variance is reduced mostly, when these conditional probabilities were constant. For splitting methods this has been shown to hold in case of level crossings, see Glasserman et. al. (1999), Garvels (2000). For example if we would like to apply three patches, the patches should be taken as

\[
\begin{align*}
\text{Patch}_1 &= \{x = (x_1, x_2) : 0 \leq x_1 \leq 4, 0 \leq x_2 \leq 4\} \\
\text{Patch}_2 &= \{x = (x_1, x_2) : 0 \leq x_1 \leq 8, 0 \leq x_2 \leq 8\} \\
\text{Patch}_3 &= \{x = (x_1, x_2) : 0 \leq x_1 \leq 12, 0 \leq x_2 \leq 12\}.
\end{align*}
\]
We chose four different patching structures: with 2, 3, 4 and 6 number of patches. Furthermore, we used iteration sample sizes $N_1 = 1000$, and for the final estimation we used always sample size $N = 5000$. In order to compare fairly, we introduce the notion of (simulation) \textit{budget} to be the total number of events that occurred during the simulation. We assume that the simulation of any event takes the same execution time.

![Figure 4: Comparison of different patching sizes for fixed budget.](image)

Figure 4 plots the relative error in these four patching schemes and shows that it remains more or less constant as we continue iterating and simulating (the $x$-axis shows the total number of generated events whereas the $y$-axis shows the relative error of the final estimator). Thus one or two iterations per patch suffices. Also we see that the scheme with the most patches (six) performs slightly better than the others.

\textbf{(B) The choice of the sample sizes.}

In order to see the effect of the sample size we fix sample size $N_1$ in the iterations to 200 and 1000, respectively. Figure 5 shows that the performances for the larger sample size $N_1 = 1000$ perform better than for the smaller $N_1 = 200$, as expected.

\textbf{(C) The choice of the weighting factor.}

Concerning the weighting factor $\alpha$ given above in equation (9), we executed the simulation with our patching algorithm with six patches, and sample size $N_1 = 1000$ in the iterations. In Rubinstein and Kroese (2004), it is suggested that $\alpha$ should be chosen between 0.10 and 0.20 in
the cross-entropy method to prevent assigning probability zero to non-zero transitions. However this problem does not occur as we implement Algorithm 1 with a high number of patches. In Figure 6, we see the relative error values for different values of $\alpha$. We observe that $\alpha = 0.01$ gives considerably lower relative values with only one iteration.

Finally, in Figure 7 we show the improvement of the performance of the standard cross-entropy method by applying our adaptation with the patches.

### 3.3 Discussion and comparisons

In this section we first return to the choice of shape of the patches, and their associated failure sets. Besides the choice described in the previous section (called the basic version), we have considered two alternative versions. Firstly, let the shapes of the patches be determined by the importance function of the splitting method, Garvels (2000), Glasserman et. al. 1999. The basic idea of the splitting method is to consider the rare event as the intersection of nested sequence of events, that are determined by levels or thresholds in the problem. In the problem that we described in Section 3.2, these levels are specified by an importance function of the form (see Garvels (2000))

$$f(x) = \frac{1}{2} \log \frac{\gamma(x)}{\gamma(0)}.$$
Hence, the patches have border sets \( \{ x : f(x) = \text{constant} \} \) which we let act also as their failure sets. The second alternative imitates the shape of the state space but lets the whole border set acting as the failure set.

We have executed our patching algorithms with the three versions, each for (the same) four different choices of sample sizes and weighting factors, resulting in the performances given in Table 2. We conclude that the second alternative is not a good method for this problem (notice also the bad estimates). Although the other two versions differ less, the basic version gives the best performance.

Trying to explain these numerical results, we investigated how sample paths reached the rare event in a `most likely way' compared to the `optimal way'. We refer to the large deviations literature for the mathematical background of these concepts (e.g. Shwartz and Weiss (1995)), but we might visualize it by plotting the drift vectors in each state. The drift vector \( d(x) \) in state \( x \) is defined by

\[
d(x) = \sum_{y \in \mathcal{X}} p^*(x, y)(y - x),
\]

where \( p^*(x, y) \) is the transition probability under the appropriate importance sampling scheme. Figures 8 and 9 show that the sample paths in the basic version are the ones most close to the optimal paths. Hence, we conjecture that the patch shapes and the choice of their associated bad states should reflect the behavior of the optimal paths to the rare event.

We remark that we have determined the optimal drifts and the optimal importance function

Figure 6: Comparison of \( \alpha \) values with fixed budget for Algorithm 1 with 6 patches.
by solving for the exact absorption probabilities using the Gauss-Seidel iteration, which is clearly tractable in this small example. The importance function turned out to be \( f(x) = x_1 + 2x_2 \).

The main purpose of our adapted version of the classical cross-entropy method is to reduce the dimensionality of the parameter estimation problem. However, similar to the classical method, we still need to store the non-zero transition probabilities. Although this is no problem for models up to hundreds of millions of non-zero transitions, when we go beyond this limit, the partitioning of the state space allows us to do the storage concurrently with the patches, which means that we implement distinct arrays, where each array contains the nonzero transitions of its corresponding patch. In this way we keep the numbers of parameters and non-zero transitions reduced to tractable sizes. The simulation of sample paths can be executed with these distinct transition probability matrices against a slight increase in simulation time due to administration overhead.

We discuss two other techniques of designing importance sampling algorithms for rare events in Markov chains. In our simulation experiments in Section 5 we have implemented these techniques in order to compare with our algorithm.

1. As explained in Section 2.4, the cross-entropy method approximates the optimal transition probabilities \( p(x, y)\gamma(y)/\gamma(x) \) by minimizing the relative entropy. Recently L’Ecuyer and Tuf-
Table 2: Comparison of performances for different patching versions.

<table>
<thead>
<tr>
<th>Version</th>
<th>Estimation</th>
<th>RE</th>
<th>RAT</th>
<th>EFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic:</td>
<td>7.3976E-23</td>
<td>0.1296</td>
<td>1.8104</td>
<td>45.5317</td>
</tr>
<tr>
<td>same shape</td>
<td>8.5653E-23</td>
<td>0.2317</td>
<td>1.8457</td>
<td>46.1780</td>
</tr>
<tr>
<td>and same</td>
<td>1.2933E-22</td>
<td>0.0177</td>
<td>1.8877</td>
<td>46.3140</td>
</tr>
<tr>
<td>failure set</td>
<td>1.3895E-22</td>
<td>0.0223</td>
<td>1.8845</td>
<td>46.1823</td>
</tr>
<tr>
<td>alternative 1:</td>
<td>1.1329E-22</td>
<td>0.0814</td>
<td>1.8313</td>
<td>45.5878</td>
</tr>
<tr>
<td>shape by the</td>
<td>1.2743E-22</td>
<td>0.0909</td>
<td>1.8282</td>
<td>45.3885</td>
</tr>
<tr>
<td>importance function</td>
<td>1.5182E-22</td>
<td>0.0375</td>
<td>1.8630</td>
<td>45.3241</td>
</tr>
<tr>
<td></td>
<td>1.5387E-22</td>
<td>0.0346</td>
<td>1.8626</td>
<td>45.3011</td>
</tr>
<tr>
<td>alternative 2:</td>
<td>6.0558E-25</td>
<td>0.2102</td>
<td>1.8125</td>
<td>49.5468</td>
</tr>
<tr>
<td>same shape</td>
<td>1.4196E-24</td>
<td>0.3477</td>
<td>1.8041</td>
<td>48.7377</td>
</tr>
<tr>
<td>with whole border</td>
<td>7.4670E-24</td>
<td>0.1027</td>
<td>1.8411</td>
<td>47.5752</td>
</tr>
<tr>
<td>as failure set</td>
<td>1.1325E-23</td>
<td>0.2117</td>
<td>1.8128</td>
<td>46.6656</td>
</tr>
</tbody>
</table>

fin (2007) considered approximating these probabilities by approximating the hitting probabilities $\gamma(\cdot)$. This so-called zero-variance approximation (ZVA) depends heavily on the specific problem of interest whereas we claim that our patching algorithm is more general applicable. For instance, the approximation $\gamma^{app}(\cdot)$ suggested in L’Ecuyer and Tuffin (2007) seems to hold for reliability models and not for queueing models. On the other hand, when one is able to express analytically the approximating function $\gamma^{app}(\cdot)$, it is shown in L’Ecuyer and Tuffin (2007) that under certain conditions the resulting importance sampling estimator shows logarithmic or strong efficiency.

2. Alternative importance sampling techniques for Markov chains exist which do not require storage of the transition probabilities. These methods are usually based on heuristic arguments and bias the transition probabilities so that more probability mass is assigned to transitions ‘in the direction’ of the rare event. For instance, transitions in reliability models consist of failures and repairs. Under the original measure the associated transition probabilities are small for failures, say at most $\varepsilon$, and high for repairs. A biasing scheme may increase the total failure transition probability to some higher weight $\theta$ (and decrease the repair transition probability accordingly). The advantage of such biasing schemes is that they are state-independent and can be executed for very large systems. The disadvantages is
that they are not designed to approximate the zero-variance measure, thus with a bad choice of biasing the performance is poor.

Finally, we give a few remarks on the splitting method for rare event simulation of Markov chains which has similarities with our method. Above we gave simulation results of the splitting method for a two-dimensional model. Clearly, the patch shapes may be determined by the importance function, but we obtained improvements by considering patch shaped based on the same structure as the original problem. Additionally, in order to use importance function, supplementary information, such as estimation of absorption probabilities from every state, has to be provided whereas in patching original structure is used as a guidance. Although by definition patching and splitting sounds similar, the purpose of these two methods is different. Patching estimates the optimal change of measure, where in the last step the change of measure is used to estimate the actual rare event probability, whereas the splitting method estimates the conditional probabilities of reaching the next level, given the current level. The number of levels and how to choose them plays a crucial role in the efficiency of the splitting method. This seems to be the same issue as our choice for the patches.
4 Face-dependent cross-entropy

In the previous section we considered the cross-entropy method for general finite-state discrete-time Markov chains, and we described our approach to update the transition probabilities in smaller groups or patches. In this section we shall consider face-homogeneous Markov chains with bounded jumps, to be explained shortly (see for instance Ignatiouk 2005). It is even allowed that the state space of the chain is infinite.

We encounter face-homogeneous Markov chains typically in Jackson queueing networks. These networks are modeled by continuous-time Markov chain, but by embedding at the jump times we obtain a discrete-time Markov chain, \((X(t))_{t=0}^{\infty}\), with state space \(\mathcal{X} = \mathbb{Z}_+^d\) (assuming \(d\) queues). Let \(\mathcal{S}\) be the power set of the indices \(\{1, 2, \ldots, d\}\), and define the function \(f : \mathbb{Z}_+^d \rightarrow \mathcal{S}\) by letting \(f(x)\) be the set of indices \(i\) for which \(x_i > 0\), i.e.,

\[ f(x) = \{i \in \{1, 2, \ldots, d\} : x_i > 0\}, \quad x \in \mathbb{Z}_+^d. \]

For any subset of indices \(\Lambda \in \mathcal{S}\), define face \(F_\Lambda\) by

\[ F_\Lambda = \{x \in \mathbb{Z}_+^d : f(x) = \Lambda\}. \]

For example, we have

\[ F_0 = \{(0, \ldots, 0)\}; \quad F_{\{1\}} = \{(x_1, 0, \ldots, 0) : x_1 > 0\}. \]
The face-homogeneity assumption is that the transition probabilities \( p(x, x + y) \) do not depend on \( x \) for all \( x \) in the same face,

\[
p(x, x + y) = p_\Lambda(y), \quad \forall x \in F_\Lambda,
\]

and for all jumps \( y \in \mathbb{Z}^d \) such that \( x + y \in \mathbb{Z}_+^d \) is a state. Equivalently, \( p(x, x + y) = p_{f(x)}(y) \).

This means that the transition probabilities out of every state belonging to the same face will be the same. Hence, for any subset \( \Lambda \in \mathcal{S} \) of indices we define the random variable \( \xi_\Lambda \) on \( \mathbb{Z}^d \) with probability mass function

\[
P(\xi_\Lambda = y) = p_\Lambda(y), \quad y \in \mathbb{Z}^d.
\]

We call this an index-jump variable, and we let these \( 2^d = |\mathcal{S}| \) different index-jump variables be independent. If we suppose that \( \xi_\Lambda(1), \xi_\Lambda(2), \ldots \) are i.i.d. copies of \( \xi_\Lambda \), then the Markov chain \( (X(t)) \) evolves according to

\[
X(t + 1) = X(t) + \xi_{f(X(t))}(t + 1).
\]

Notice that for each index set \( \Lambda \) the partial sums \( \left( \sum_{\tau=1}^{\infty} \xi_\Lambda(t) \right)_{n=1}^\infty \) of the index-jump variables constitutes a homogeneous random walk in \( \mathbb{Z}^d \), thus the Markov chain \( X(t) \) might be called a face-homogeneous random walk.

Again we assume some rare event probability \( \mathbb{P}(A) \) given by absorption in the bad set before absorption in the good set. We apply the cross-entropy method for finding a change of measure (or transition probability matrix) \( P^* \) under which the importance sampling simulations will be executed. We restrict the class of feasible transition matrices \( \mathcal{P} \) in the optimization program (4) to those for which the face-homogeneity is retained. Hence, the sample path probability (5) can be expressed as

\[
d\mathbb{P}^*(X) = \prod_{t=1}^{T} p^*_{f(X(t))}(X(t + 1) - X(t)).
\]

Therefore, if we let \( M_\Lambda \) denote the number of jumps of the index-jump variable \( \xi_\Lambda \), we get a total of \( \sum_{\Lambda \subseteq \{1, 2, \ldots, d\}} M_\Lambda \) parameters to update during the cross-entropy iterations, whatever the size of the state space of the Markov chain is. If we denote by \( N_\Lambda^{(i)}(y) \) the number of times that the \( i \)-th sample path of the Markov chain makes the jump \( \xi_\Lambda = y \), i.e., the associated transition
\((x \to x + y)\) for some \(x \in F_\Lambda\), and denote by \(L^{(i)}\) the likelihood ratio of the \(i\)-th sample path, then the update rule (8) in the \(j\)-th cross-entropy iteration becomes

\[
p^{(j+1)}(y) = \frac{\sum_{i=1}^{N_1} L^{(i)} \{X^{(i)}(T) \in \mathcal{F}\} N_\Lambda^{(i)}(y)}{\sum_{i=1}^{N_1} L^{(i)} \{X^{(i)}(T) \in \mathcal{F}\} \sum_y N_\Lambda^{(i)}(y)}.
\]

The weighting expression (9) is still used with some parameter \(\alpha \in (0, 1)\).

A similar idea of reducing the parameter space has been studied in de Boer and Nicola (2002) for state-dependent tilting under the method called “Boundary Layers”. The fundamental assumption this method is based on is that when a queue’s content becomes large, the transition probabilities (in an optimal change of measure) will hardly depend on that queue’s content. Therefore, it is suggested to assign the same set of transition probabilities to a group of such states. Thus, the grouping is based on the levels of the queues. We, on the other hand, consider the intrinsic homogeneous face structure of the Markov chain which we retain when implementing the cross-entropy method. Specifically for queueing applications, the fundamental observation for face-dependent cross-entropy method is that it is independent of the level of the queues.

## 4.1 The patching algorithm

As mentioned in Section 3, the main idea of the patching algorithm is to update the change of measures for the whole state space in small portions. However, what matters in case of face-dependency is the size of the total number of transitions \(\sum_{\lambda \subseteq \{1, 2, \ldots, d\}} M_\lambda\) of the \(d\) random walks constituting the Markov chain. To explain the patching idea, we return to the original setting in Section 2 and denote the rare event by \(A_n\) where the rarity parameter \(n \to \infty\). For instance in the Jackson network applications, the rarity parameter \(n\) will be an overflow level of one or more queues.

For the patching version of our algorithm we apply the same technique as the standard cross-entropy method for level-crossing probabilities (Rubinstein and Kroese 2004), and let the rarity parameter \(n\) increase as the iterations go on. For the first Patch we set \(n_0\) to some low value. We update the change of measures according to this level. The parameters updated in Patch \(i\) form a good initial probability matrix for Patch \(i+1\). By gradually increasing the value of the parameter \(n\), the sample paths are more and more pushed to the rare event set. However, as will be seen in following section, the starting value plays a crucial role in the reduction of the variance.
4.2 Implementation issue: how to choose the parameters?

![Graph showing comparison of face-dependent cross-entropy with Algorithm 1 with 6 number of patches with $N_1 = 1000$. The x-axis shows the total number of generated events.]

Figure 10: Comparison of Face-dependent Cross-Entropy with Algorithm 1 with 6 number of patches with $N_1 = 1000$. The x-axis shows the total number of generated events.

In order to observe the effects of (A) different values of rarity parameter $n$ (sizes of the patches), (B) the sample size $N_1$ in the iterations and (C) the weighting factor $\alpha$, we consider a small two-dimensional Jackson network problem with two single server queues. The jobs arrive at the first queue according to a Poisson process of rate $\lambda = 1$. There are single servers at both of the queues and exponentially distributed service times with rates $\mu_1 = 12$ and $\mu_2 = 10$. The routing matrix $R$ of the system is as follows:

$$R = \begin{pmatrix} 0 & \frac{1}{10} \\ \frac{1}{10} & 0 \end{pmatrix}.$$

The failure set is reached whenever at least one of the queues exceeds $n = 12$.

For this small example we implemented our face-dependent cross-entropy algorithm, and our original patching Algorithm 1 with four different patching structures: with 2, 3, 4 and 6 number of different rarity levels $n$. In order to see the effect of the sample sizes we fix sample size $N_1$ in the iterations to 5 and 1000, respectively. For the final estimation we used samples size $N = 5000$ for all methods.

Figure 10 shows that for fixed sample size $N_1 = 1000$, the relative error in these four patching schemes gets higher as we increase the number of patches. Hence, it is not helpful
to apply Algorithm 1 when face-dependent cross-entropy is already giving good results. Figure 11 shows that the performances for the larger sample size $N_1 = 1000$ does not differ so much from the smaller $N_1 = 5$. However, we also have to keep in mind that this example is a small generic example to understand the general behavior of the algorithm. Although it will provide a guidance in Section 5, different problem settings can give slightly different results.

Concerning with the weighting factor $\alpha$, similarly to what we did in Section 3.2, we executed the simulation with our face-dependent cross-entropy algorithm and sample size $N_1 = 1000$ in
the iterations. In Figure 12, we see the relative error values for different values of $\alpha$. Here we observe that lower $\alpha$ values, such as 0.001, 0.01 and 0.05, give lower relative values.

5 Numerical examples

In this section we will discuss the performance of our algorithms mentioned in the previous sections for two different models: highly reliable Markovian systems and Jackson networks. These systems differ in terms of rare event setting as mentioned in Heidelberger (1995). In reliability models there are a few events happening that create the notion of rare events whereas in Jackson networks rare events happen as a combination of large number of events. It is well known in the literature that different kinds of importance sampling schemes should be applied to both types of problems. Since, our aim is to come up with a general algorithm which is applicable to any type of Markov chain, we will discuss the performance of the algorithms for both models.

CE-D and Patch-D refer to the cross-entropy algorithms in Sections 2.4 and 3.1 without and with patching, respectively. Similarly, CE-F and Patch-F notify that we have implemented the face-dependent cross-entropy methods described in section 4. In the notation CE-D($k;n;\alpha$), $k$ refers to the number of iterations the cross-entropy is run, $n$ refers to the number of sample paths produced in each iteration, $N_1$ in Algorithm 1, and $\alpha$ refers to the weighting factor in equation (9). The same notation is valid also for the Patch-D, CE-F and Patch-F. For the state-dependent and face-dependent patching algorithms we also add a number at the end of the usual notation to indicate the distance we take between each patch. That means, as the number decreases, we will have more patches. In Figure 13 we give a few examples: Patch-F($k;n;\alpha$)$-2$ and Patch-F($k;n;\alpha$)$-5$.

The results for each example contain the probability estimation and the three performance measures discussed in Section 2.3. We run the cross-entropy method with a predefined number of iterations with a predefined sample size which refers to a fixed budget for running the algorithms. Then, we implement the importance sampling to estimate the rare event probability and performance measures. We do not check whether the cross-entropy method has converged or not. Our aim is to show the performance of the algorithms with a fixed budget.
5.1 Reliability

Our first example is a model of a highly reliable Markovian system which has been studied largely in relation to importance sampling estimation of system failure and mean time to failure, for instance Alexopoulos and Shultes (2001); L’Ecuyer and Tuffin (2007); Goyal et. al. (1992); Juneja and Shahabuddin (2001a,2001b); Shahabuddin (1994a, 1994b). In this reliability system there are \( d \) different component types, where type \( i \) contains \( K_i \) identical components. Each component fails, independently of other events, after an exponentially distributed failure time, with failure rate \( \lambda_i \) for type \( i \) components. Failed components may be repaired by one or more service men. The repair times of type \( i \) components are exponentially distributed with rate \( \mu_i \).

We make a few system assumptions.

Assumption 1.

(i) There is no failure propagation.

(ii) There is no group repair.

(iii) The failure rates are functions of a parameter \( \epsilon \) according to

\[
\lambda_i = \epsilon^{r_i},
\]

where \( \epsilon > 0 \) small and \( r_i \geq 1 \).

(iv) The repair rates are several orders of magnitude larger: \( \mu_i \geq \mu_{\text{min}} = 1 \).

However, we do allow priority of service. The system is modeled by a continuous-time Markov chain with states \( x = (x_1, \ldots, x_d) \) representing the number of failed components per type. Thus
the state space is

$$\mathcal{X} = \prod_{i=1}^{d} \{0, 1, \ldots, K_i\},$$

with cardinality $|\mathcal{X}| = \prod_{i=1}^{d} (K_i + 1)$. Under Assumption 1, (i) and (ii), the Markov chain has jumps of size one, viz. an operational component fails, or a failed component is repaired. It is easy to see that the chain is irreducible over its state space. We might generalize to allow failure propagation, group repair and other transition structures based on state information, but at the moment we consider the most simple models.

The associated discrete-time Markov chain $(X(t))_{t=0}^{\infty}$ is constructed by embedding the continuous-time Markov chain at its jump times. The good set $\mathcal{G}$ consists of the single ‘perfect’ state $0 = (0, 0, \ldots, 0)$ with all components operational, and the ‘bad’ or failure set $\mathcal{F}$ represents all states in which the system is down, for instance $\mathcal{F}$ is the single state $(K_1, K_2, \ldots, K_d)$ with all components of all types failed, or the set of states in which all components of at least one type have failed. Recall that $\gamma(x)$ is the probability of hitting the failure set when the chain starts in an internal state $x \in \mathcal{T}$, and that we are interested in these hitting probabilities, since our target is the failure probability (FP)

$$\gamma(0) = \sum_{x \in \mathcal{T}} p(0, x) \gamma(x),$$

which we wish to estimate by simulation. Besides its own interest, the failure probability is used for determining the system unavailability and the mean time to failure, (Goyal et. al. 1992).

The rarity parameter in this reliability model is $n = \lceil 1/\varepsilon \rceil$ (see (11) for $\varepsilon$), thus we might denote the failure probability by $\gamma_n(0)$. Under the conditions of Assumption 1 it has been shown in Juneja and Shahabuddin (2001a) that $\gamma_n(0) \to 0$ polynomially as $n \to \infty$, viz., let $r = \min \{r_i : i = 1, 2, \ldots, d\}$, then

$$\lim_{n \to \infty} \frac{1}{\log n} \gamma_n(0) = -r.$$  

Several importance sampling schemes have been proposed to give bounded relative error (in certain cases) or asymptotically optimal estimators, notably: simple failure biasing (SFB) (Shahabuddin 1994a), balanced failure biasing (BFB) (Shahabuddin 1994a), balanced likelihood ratio method (BLR) (Alexopoulos and Shultes 2001), and more recently a zero-variance approximation method (ZVA) (L’Ecuyer and Tuffin 2007).
• SFB with parameter $\theta \in (0, 1)$: increase the total failure-transition probability at any state to a constant $\theta$ while keeping their original mutual proportionalities; similarly, the repair-transition probabilities are reduced to $1 - \theta$.

• BFB with parameter $\theta \in (0, 1)$: similar to SFB($\theta$), however it distributes the total probability $\theta$ equally to the failure-transitions.

• BLR: the crucial observation is that each repair must be preceded by a failure. Then one can force the product of respective pairs of event likelihood ratios to be one, and thus, the likelihood ratio of a sample path to be bounded above by one.

• ZVA: see Section 3.3 for a short summary and the underlying ideas of this method. Recall that the importance sampling transition probabilities become

$$p^*(x, y) = \frac{p(x, y)\gamma_{\text{app}}(y)}{\sum_{y \in X} p(x, y)\gamma_{\text{app}}(y)}. \tag{12}$$

In L’Ecuyer and Tuffin (2007) it is suggested to let the approximated hitting probabilities $\gamma_{\text{app}}(y) = v_0(y)$, where $v_0(y)$ is the maximal probability of paths of states $y = y_0 \rightarrow y_1 \rightarrow \cdots \rightarrow y_k \in \mathcal{F}$ going from state $y$ to the bad set $\mathcal{F}$. Notice that the denominator in the approximation (12) is simply the normalizing constant. Also we have considered a refinement suggested in L’Ecuyer and Tuffin (2007) to replace $v_0(y)$ by $v_1(y) = v_0(y)^\alpha$, where $\alpha$ is determined by first executing an importance sampling procedure to get a rough estimate $\hat{\gamma}$ of the failure probability $\gamma(0)$, and then matching $v_0(0)^\alpha = \hat{\gamma}$.

We have implemented the SFB, BFB and ZVA schemes to compare with our method in both statistical and computational performance. We give three examples: the first example is a small test example with 1331 states and 7260 nonzero transition matrix entries (parameters). The second example is from L’Ecuyer and Tuffin (2007) with 2197 states and 12168 nonzero matrix entries, and the third example has a large state space of 4084101 states and over $20 \times 10^6$ parameters. To assess the estimations we computed numerically the hitting probabilities in the two smaller examples by applying the Gauss-Seidel iteration using software with arbitrary-precision arithmetic.

Example 1

There are 3 types, each with 10 components, with failure rates $\lambda_1 = \varepsilon$, $\lambda_2 = 5\varepsilon$ and $\lambda_3 = 8\varepsilon$
(all failure rates are of the same order as functions of $\varepsilon$: this is called “balanced”) and repair rates $\mu_i = 1$ for all $i$. $\varepsilon$ is taken to be 0.01. There is a single repairman but no priorities are applied between types. All the repairs are made one by one as soon as one has failed, no group repairs are allowed. Finally, the system breaks down when all components of all types have failed, which corresponds to $\mathcal{F} = \{10, 10, 10\}$. The associated failure probability is found by the Gauss-Seidel method to be $1.057E - 24$.

The results for all the models can be found in Table 3. All results are averages of 5 repetitions, where the sample size in the (final) importance simulations for estimating the rare event probability is $10^6$ in all experiments.

Recall the three performance measures of the importance sampling estimators introduced in Section 2.3: RE (relative error), RAT (logarithmic ratio) and EFF (efficiency). The EFF measure takes into account the total running time on the computer which includes the execution of the cross-entropy iterations and the final importance sampling simulation. Better performance is obtained by a smaller RE, higher RAT, and larger EFF.

<table>
<thead>
<tr>
<th>Method</th>
<th>Estimation</th>
<th>RE</th>
<th>RAT</th>
<th>EFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFB(0.7)</td>
<td>8.15E-25</td>
<td>0.3477</td>
<td>1.7950</td>
<td>48.7823</td>
</tr>
<tr>
<td>SFB(0.8)</td>
<td>7.20E-25</td>
<td>0.3167</td>
<td>1.7973</td>
<td>49.0323</td>
</tr>
<tr>
<td>SFB(0.9)</td>
<td>3.92E-25</td>
<td>0.2971</td>
<td>1.8022</td>
<td>49.6496</td>
</tr>
<tr>
<td>BFB(0.7)</td>
<td>8.19E-25</td>
<td>0.1735</td>
<td>1.8179</td>
<td>49.2005</td>
</tr>
<tr>
<td>BFB(0.8)</td>
<td>8.11E-25</td>
<td>0.1734</td>
<td>1.8201</td>
<td>49.4110</td>
</tr>
<tr>
<td>BFB(0.85)</td>
<td>7.39E-25</td>
<td>0.2141</td>
<td>1.8144</td>
<td>49.3780</td>
</tr>
<tr>
<td>ZVA($v_0$)</td>
<td>7.27E-26</td>
<td>0.4043</td>
<td>1.7964</td>
<td>50.9142</td>
</tr>
<tr>
<td>ZVA($v_1$)</td>
<td>2.95E-25</td>
<td>0.3298</td>
<td>1.8002</td>
<td>49.7402</td>
</tr>
<tr>
<td>CE-D(7:20000:0.15)</td>
<td>5.90E-29</td>
<td>0.5433</td>
<td>1.0649</td>
<td>*</td>
</tr>
<tr>
<td>CE-D(10:30000:0.15)</td>
<td>3.77E-29</td>
<td>0.5544</td>
<td>1.0867</td>
<td>*</td>
</tr>
<tr>
<td>CE-D(15:80000:0.15)</td>
<td>3.29E-27</td>
<td>0.5798</td>
<td>1.8029</td>
<td>*</td>
</tr>
<tr>
<td>Patch-D(10:20000:0.15)-2</td>
<td>5.07E-25</td>
<td>0.0916</td>
<td>1.8408</td>
<td>48.7110</td>
</tr>
<tr>
<td>Patch-D(10:20000:0.15)-2(4)</td>
<td>4.08E-25</td>
<td>0.0861</td>
<td>1.8424</td>
<td>48.9142</td>
</tr>
<tr>
<td>Patch-D(10:40000:0.15)-2(4)</td>
<td>7.16E-25</td>
<td>0.0815</td>
<td>1.8446</td>
<td>48.2375</td>
</tr>
<tr>
<td>Patch-D(10:40000:0.15)-2(4)</td>
<td>7.09E-25</td>
<td>0.0449</td>
<td>1.8663</td>
<td>48.7651</td>
</tr>
</tbody>
</table>

Before comparing the importance sampling methods, it is important to point out the difference between Gauss-Seidel estimation and the estimation obtained by these importance sampling
methods. We observed that our importance sampling methods gave slightly lower estimations than the Gauss-Seidel solution. This might be explained by the well known overbiasing effect that pops up easily in importance sampling implementations (Devetsikiotis and Townsend (1993) and Smith (2001)), specifically in multi-dimensional Markov chain problems, Andradottir et. al. (1995). The difficulty is that by a minor deviation of the optimal biasing parameters, the importance sampling simulation generates actually too many observations of the rare event, almost all with very small likelihood ratios. Generally, this is hard to solve, except either by adjusting the biasing scheme (like our patching proposal), or by using a larger sample size.

Although the number of states is not very high, we noticed from the experiments that this kind of Markovian reliability model with only one failure state is a hard problem for every approach. We experimented with a large range of $\theta$’s in the SFB($\theta$) and BFB($\theta$) schemes. Although they provide fast estimations, the relative error is not very small (relative error of 17% or worse). BFB($\theta$) gives better results compared to SFB($\theta$), but still poor. There is also not much difference in performance between ZVA and the biasing schemes. ZVA($v_1$) gives slightly good results compared to ZVA($v_0$). However, none of them gives relative error smaller than 10%.

Also, we experimented with various CE-D($k; n; \alpha$) and Patch-D($k; n; \alpha$) schemes. For CE-D($k; n; \alpha$), we could not get good performance (nor good estimates!) even not after increasing the number of sample paths in each iteration. Consequently, the EFF performance measure for CE-D($k; n; \alpha$) is not compatible with other methods. Though, Patch-D($k; n; \alpha$) schemes gave excellent performance when the number $k$ of iteration sample paths was sufficiently large. Clearly, Patch-D($k; n; \alpha$) schemes outperform the best bias schemes.

Using 20000 sample paths in standard cross-entropy did not give us good results. Hence keeping the sample size the same we report a few combinations with clustering in groups of 2 starting either with 2 or 4, because larger clusters showed worse results. As expected, if we increase the number of sample paths in each iteration, the relative error decreases. Although the patching algorithm consumes more computing time, the efficiency is about the same as the other algorithms due to its variance improvements.

Example 2
This is an example from L’Ecuyer and Tuffin (2007). There are again 3 types, this time each with 12 components, with failure rates $\lambda_1 = \varepsilon$, $\lambda_2 = 1.5\varepsilon$ and $\lambda_3 = 2\varepsilon^2$ (unbalanced) and repair
rates $\mu_i = 1$ for all $i$. We only give results for $\varepsilon = 0.001$. There are ample repairmen who work simultaneously on all failed components. There is no priority or group repair. The system breaks down as soon as at least one component type has less than 2 operational units, i.e.,

$$\mathcal{F} = \{x = (x_1, x_2, x_3) : \exists i \ x_i \geq 11\}.$$  

The associated failure probability is found by the Gauss-Seidel method to be $3.892E-28$. The results for all the models can be found in Table 4. All results are averages of 5 repetitions, where the sample size in the (final) importance sampling simulations for estimating the rare event probability is $10^6$ in all experiments.

<table>
<thead>
<tr>
<th>Method</th>
<th>Estimation</th>
<th>RE</th>
<th>RAT</th>
<th>EFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFB(0.7)</td>
<td>3.75E-28</td>
<td>0.6725</td>
<td>1.7597</td>
<td>54.0421</td>
</tr>
<tr>
<td>BFB(0.8)</td>
<td>4.43E-28</td>
<td>0.3749</td>
<td>1.7765</td>
<td>54.1473</td>
</tr>
<tr>
<td>BFB(0.9)</td>
<td>3.71E-28</td>
<td>0.3746</td>
<td>1.7766</td>
<td>54.3167</td>
</tr>
<tr>
<td>ZVA($v_0$)</td>
<td>3.88E-28</td>
<td>0.2563</td>
<td>1.8246</td>
<td>55.6483</td>
</tr>
<tr>
<td>ZVA($v_1$)</td>
<td>3.87E-28</td>
<td>0.0135</td>
<td>1.9306</td>
<td>58.7978</td>
</tr>
<tr>
<td>CE-D(7; 5000;0.15)</td>
<td>2.24E-28</td>
<td>0.3228</td>
<td>1.8824</td>
<td>60.1135</td>
</tr>
<tr>
<td>CE-D(7;10000;0.15)</td>
<td>2.05E-28</td>
<td>0.1801</td>
<td>1.9180</td>
<td>59.1779</td>
</tr>
<tr>
<td>CE-D(7;20000;0.15)</td>
<td>3.42E-28</td>
<td>0.1703</td>
<td>1.9089</td>
<td>57.7000</td>
</tr>
<tr>
<td>CE-D(7;50000;0.15)</td>
<td>3.83E-28</td>
<td>0.0048</td>
<td>1.9642</td>
<td>58.0076</td>
</tr>
<tr>
<td>Patch-D(2; 300;0.15)-2</td>
<td>3.24E-28</td>
<td>0.0554</td>
<td>1.9454</td>
<td>58.7695</td>
</tr>
<tr>
<td>Patch-D(3; 500;0.15)-2</td>
<td>3.84E-28</td>
<td>0.0086</td>
<td>1.9664</td>
<td>58.9652</td>
</tr>
<tr>
<td>Patch-D(1;1000;0.15)-2</td>
<td>3.90E-28</td>
<td>0.0140</td>
<td>1.9578</td>
<td>59.7262</td>
</tr>
</tbody>
</table>

Observations in terms of comparison of methods are generally similar to Example 1. ZVA($v_1$) provided better estimation, with a relative error around 1.3%. Cross-Entropy (without patching) showed small relative error when we increased the sample size to 50,000 in each iteration, whereas Patch-D($k; n; \alpha$) obtained the same relative error with much less sample paths. All the patching results are given in clusters of 2 and for fixed $\alpha = 0.15$.

To check empirically the asymptotic efficiency of our Patch-D($k; n; \alpha$) algorithm for this specific problem setting, the experiments are repeated for increasing number of components (where the failure set remains to be a layer of two), and for decreasing $\varepsilon$ values. All the simulation results for asymptotic efficiency can be found in Table 5. It is clear from the table...
that as $\varepsilon$ decreases, meaning as the event gets rarer, the method provides better solutions. In all of the cases we see that the relative error grows linearly with respect to level whereas the probability decreases exponentially fast. The last column with RAT values also approves the asymptotic efficiency observation. As the number of components increase, RAT values approach to 2 as in the definition of asymptotic efficiency.

Table 5: Test of asymptotic efficiency of Patch-D for Example 2.

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of Components</th>
<th>Estimation</th>
<th>RE</th>
<th>RAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon = 0.05$</td>
<td>8</td>
<td>1.14E-06</td>
<td>0.0033</td>
<td>1.8224</td>
</tr>
<tr>
<td>3 iterations</td>
<td>12</td>
<td>6.59E-11</td>
<td>0.0081</td>
<td>1.8229</td>
</tr>
<tr>
<td>$10^3$ replications</td>
<td>20</td>
<td>1.56E-19</td>
<td>0.0354</td>
<td>1.8372</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>1.39E-42</td>
<td>0.3844</td>
<td>1.8788</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>3.88E-68</td>
<td>0.7191</td>
<td>1.9190</td>
</tr>
<tr>
<td>$\varepsilon = 0.01$</td>
<td>8</td>
<td>5.43E-11</td>
<td>0.0025</td>
<td>1.9182</td>
</tr>
<tr>
<td>3 iterations</td>
<td>12</td>
<td>4.27E-18</td>
<td>0.0034</td>
<td>1.9371</td>
</tr>
<tr>
<td>$10^3$ replications</td>
<td>20</td>
<td>2.00E-32</td>
<td>0.0068</td>
<td>1.9473</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>1.53E-68</td>
<td>0.0401</td>
<td>1.9532</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>4.59E-105</td>
<td>0.1204</td>
<td>1.9603</td>
</tr>
<tr>
<td>$\varepsilon = 0.001$</td>
<td>8</td>
<td>5.09E-17</td>
<td>0.0036</td>
<td>1.9396</td>
</tr>
<tr>
<td>3 iterations</td>
<td>12</td>
<td>3.86E-28</td>
<td>0.0029</td>
<td>1.9647</td>
</tr>
<tr>
<td>$10^3$ replications</td>
<td>20</td>
<td>1.69E-50</td>
<td>0.0070</td>
<td>1.9662</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>1.12E-106</td>
<td>0.0320</td>
<td>1.9718</td>
</tr>
</tbody>
</table>

Example 3
There are 5 types, with 20 components each, with unbalanced failure rates; $2.5\varepsilon$, $3\varepsilon$, $\varepsilon^2$, $6\varepsilon$, $2\varepsilon^2$ and repair rates $\mu_i = 1$ for all $i$. $\varepsilon$ is taken to be 0.001. There is a single repairman who applies preemptive priority according to $1 > 2 > 3 > 4 > 5$. Type 1 has the highest priority. All types are repaired one by one as soon as one has failed. The system breaks down as soon as all components of at least one type have failed. All results are averages of 5 repetitions, with sample size $10^6$ for the final estimation.

For results of Example 3, please refer to Table 6. The BFB and CE-D algorithms gave quite different estimates than the other algorithms, even after having increased the number of sample paths (for BFB only) form $10^6$ to $10^8$, and thus we conjecture that the correct estimates are
Table 6: Comparison of performances for Example 3.

<table>
<thead>
<tr>
<th>Method</th>
<th>Estimation</th>
<th>RE</th>
<th>RAT</th>
<th>EFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFB(0.8)</td>
<td>2.60E-30</td>
<td>0.8633</td>
<td>1.7872</td>
<td>*</td>
</tr>
<tr>
<td>BFB(0.85)</td>
<td>8.18E-30</td>
<td>0.6977</td>
<td>1.7697</td>
<td>*</td>
</tr>
<tr>
<td>BFB(0.9)</td>
<td>6.38E-31</td>
<td>0.9995</td>
<td>1.7351</td>
<td>*</td>
</tr>
<tr>
<td>ZVA($v_0$)</td>
<td>5.20E-23</td>
<td>0.9295</td>
<td>1.7663</td>
<td>44.5897</td>
</tr>
<tr>
<td>ZVA($v_1$)</td>
<td>3.60E-23</td>
<td>0.4775</td>
<td>1.7824</td>
<td>44.6055</td>
</tr>
<tr>
<td>CE-D(7;10000;0.15)</td>
<td>8.27E-28</td>
<td>0.4450</td>
<td>1.6712</td>
<td>44.6123</td>
</tr>
<tr>
<td>Patch-D(7;500;0.05)-2</td>
<td>1.16E-23</td>
<td>0.1420</td>
<td>1.8123</td>
<td>46.9334</td>
</tr>
<tr>
<td>Patch-D(10;500;0.05)-2</td>
<td>6.74E-24</td>
<td>0.1068</td>
<td>1.8249</td>
<td>47.6279</td>
</tr>
<tr>
<td>Patch-D(10;1000;0.05)-2</td>
<td>6.32E-24</td>
<td>0.0954</td>
<td>1.8321</td>
<td>48.2418</td>
</tr>
</tbody>
</table>

given by these other algorithms, i.e., ZVA and Patch-D. In this problem our patching algorithm outperforms the ZVA algorithms when these are implemented as suggested in L’Ecuyer and Tuffin (2007), viz. using the maximal probability of paths to the rare event as an approximation to the absorption probability. This means that we recommend to adapt this approximation.

5.2 Jackson network

Our second example comprises the well known product-form Jackson queueing networks. There is ample literature on importance sampling simulation for estimating excessive backlogs (or buffer overflows) in networks with more or less general network topologies, by algorithms with proven complexity or by heuristic algorithms. We mention here a few.

- Tandem queue and total population overflow: Parekh and Walrand (1989) gave a state-independent heuristic algorithm based on exchanging the arrival rate with the service rate of the bottleneck server. Later, this heuristic has been analyzed in de Boer (2006) and Glasserman and Kou (1995) to show that asymptotic optimality is obtained only in certain specific parameter regions. In Dupuis et. al. (2007) a state-dependent change of measure has been developed and proven to be asymptotically optimal. The method is based on connections between importance sampling, stochastic games and optimal control.

- Tandem queue and buffer overflow at the second queue: using a Markov additive process representation (Kroese and Nicola 2002) construct a state-dependent change of measure (dependent on the content at the first queue) and prove asymptotic optimality.
• This has been generalized to Jackson networks and buffer overflow at a single target queue in Juneja and Nicola (2005). The resulting change of measure is obtained as a solution to a set of conditions and proven to be asymptotically optimal.

• General topologies and population overflow: Frater et al. (1991) apply the large deviations heuristic of Parekh and Walrand (1989) to come up with a set of nonlinear equations whose (unique) solution serves as a state-independent change of measure. Note that this method is only applicable to method where there is only one server with the highest load. In Nicola and Zaburnenko (2007) heuristic state-dependent algorithms are constructed based on the observation that the state-dependency of the optimal (zero-variance) change of measure is strong along the boundaries and weak and diminishes in the interior. The proposed solution is a state-dependent combination of matrix solutions.

• General Markovian queueing models and rare events: de Boer and Nicola (2002) and de Boer et al. (2004) report cross-entropy based adaptive state-dependent algorithms. In de Boer and Nicola (2002) the algorithm uses boundary layers as we explained in Section 4. The method in de Boer et al. (2004) is a three stage adaptive importance sampling algorithm which uses cross entropy to update the arrival and departure parameters together with routing probabilities. First, they estimate the minimum cross-entropy tilting parameter for a small buffer level; next, they use this as the starting tilting parameter value for the estimation of the optimal tilting parameter for the actual buffer level. Lastly, the tilting parameter obtained is used to estimate the population overflow probability.

We will compare our algorithms with the heuristic methods of de Boer et al. (2004) (in all our examples), Frater et al. (1991)(first two examples) and Nicola and Zaburnenko (2006) (in example 6). To include efficiency also into the comparison, we implemented the algorithm given in de Boer et al. (2004) best as we can as described in the paper. The results for example 6 are directly taken from Nicola and Zaburnenko (2006). For ease of comparison the method discussed in Frater et al. (1991) will be denoted as FLA in all of the examples.

Suppose that the network consists of $d$ queues, Poisson arrivals with rates $\lambda_i$ ($i = 1, \ldots, d$), single servers at the queues and exponentially distributed service times with rates $\mu_i$ ($i = 1, \ldots, d$), routing probabilities $r_{ij}$ ($i, j = 1, \ldots, d$) such that the routing matrix $R$ is substochastic. We make the usual stability assumption: let $(\nu_i)_{i=1}^d$ be the traffic intensities determined as the
solution to the traffic equations

$$\nu_j = \lambda_j + \sum_{i=1}^{d} \nu_i r_{ij}, \quad j = 1, \ldots, d.$$ 

**Assumption 2.** For any $i = 1, \ldots, d$ the load $\rho_i = \nu_i / \mu_i < 1$.

The network is modeled by a continuous-time Markov chain with states $x = (x_1, \ldots, x_d)$ representing the number of jobs present at the queues (including the servers). Again we consider its associated discrete-time Markov chain $(X(t))_{t=0}^{\infty}$ by embedding at the jump times. The state space of the chain has infinite size, being the subspace of all nonnegative integer vectors, $\mathbb{Z}^d_{\geq 0}$. However, we consider only the following two bad sets:

1. Total population exceeds $n$, i.e.,

   $$\mathcal{F} = \{x : \sum_{i=1}^{d} x_i \geq n\}.$$

2. At least one queue exceeds $n$, i.e.,

   $$\mathcal{F} = \{x : \max_{i=1, \ldots, d} x_i \geq n\}.$$

In both cases we can reduce the infinite state space to a finite state space $\mathcal{X}$. The good set consists of the single empty state: $0 = (0, \ldots, 0)$. Thus, similarly as in the reliability example we consider the failure probability (FP)

$$\gamma(0) = \sum_{x \in \mathcal{T}} p(0, x) \gamma(x),$$

where $\gamma(x)$ is the probability of hitting the bad set $\mathcal{F}$ before hitting $0$, when the chain would start in state $x$.

In this example the rarity parameter is the buffer or the population overflow level $n$. It has been shown in [21] that

$$\lim_{n \to \infty} \frac{1}{n} \log \gamma_n(0) = -\theta.$$ 

**Example 4**

This example of a five-node Jackson network is taken from de Boer and Nicola (2002), see also de Boer et. al. (2004). We have implemented the three stage method of de Boer et. al. (2004),
which we denote by CE-3-Stage($k; n; l_0$): $k$ is the number of iterations where cross-entropy is implemented, $n$ denotes the number of repetitions used in each iteration whereas $l_0$ refers to the small buffer level to obtain the starting tilting parameters.

Jobs arrive at the first queue according to a Poisson process with rate $\lambda_1 = 3$. There are single servers at each queue along with exponentially distributed service times with rates $\mu_1 = 40$, $\mu_2 = 20$, $\mu_3 = 25$, $\mu_4 = 50$ and $\mu_5 = 60$ with all routing probabilities equal to 0.5. Other than server 3, which is the bottleneck queue, all the loads are equal to 0.1. Consider the estimation of the overflow probability that total population exceeds level $l = 80$.

In this example we have 32801517 states with 277848143 nonzero transitions, which is quite high to implement the state-dependent change of measure. That is why, here we will only present the results for the face-dependent cross-entropy methods. We used $10^5$ replications in the final estimation by importance sampling. In all Patch-F implementations we started with the lower level value of 10 and increased it by 10 in every Patch until we hit the actual overflow level of 80. All results in Table 7 are averages of 5 repetitions.

As we can see in Table 7, FLA method gives outstanding results with respect to efficiency since the change of parameters are obtained separately before starting the simulation. It is also noteworthy to remark that the parameters found by CE-3Stage are close to the ones calculated by the method FLA, as also noted in de Boer et. al. (2004). However, FLA is not applicable to every problem and also we will see in the next example that it does not always provide better results.

As expected we see in Table 7 that CE-F gives estimations with lower relative error than
Table 7: Comparison of performances for Example 4.

<table>
<thead>
<tr>
<th>Method</th>
<th>Estimation</th>
<th>RE</th>
<th>RAT</th>
<th>EFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE-3Stage(7;100000;5)</td>
<td>7.60E-55</td>
<td>0.0170</td>
<td>1.9576</td>
<td>107.6620</td>
</tr>
<tr>
<td>FLA</td>
<td>7.70E-55</td>
<td>0.0164</td>
<td>1.9738</td>
<td>110.1838</td>
</tr>
<tr>
<td>CE-F(30;100000 ;0.001)</td>
<td>7.72E-55</td>
<td>0.0110</td>
<td>1.9793</td>
<td>108.9570</td>
</tr>
<tr>
<td>CE-F(30;400000 ;0.001)</td>
<td>7.74E-55</td>
<td>0.0096</td>
<td>1.9813</td>
<td>108.5420</td>
</tr>
<tr>
<td>Patch-F(10;10000 ;0.001)</td>
<td>7.38E-55</td>
<td>0.0744</td>
<td>1.9498</td>
<td>108.5244</td>
</tr>
<tr>
<td>Patch-F(10;10000 ;0.001)</td>
<td>7.77E-55</td>
<td>0.0626</td>
<td>1.9521</td>
<td>107.6283</td>
</tr>
<tr>
<td>Patch-F(10;10000;0.001)</td>
<td>7.69E-55</td>
<td>0.0710</td>
<td>1.9499</td>
<td>107.3333</td>
</tr>
</tbody>
</table>

Patch-F. The main purpose of patching algorithm is to improve the cross-entropy when it does not provide good results with a given budget. However, here the situation is different. CE-F does give good results, hence patching cannot improve more. However, the results provided by CE-F are not so much different from CE-3Stage. We see that in terms of efficiency, CE-F gives slightly better results.

**Equal Loads**

Next, we simulate the same network, but with all servers having an equal load of 0.1, with $\lambda = 3$, $\mu_1 = 40$, $\mu_2 = 20$, $\mu_3 = 50$, $\mu_4 = 50$ and $\mu_5 = 60$, and all routing probabilities equal to 0.1. For the final estimation by importance sampling we used $10^6$ samples and all results are average of 5 repetitions.

The simulation results are presented in Table 8. We found that the number of sample paths, $N_1$, per patch could be less compared to the previous bottleneck problem. The same observation is also true for CE-F. If we compare the results in de Boer et. al. (2004) with CE-F and Patch-F, we see that CE-F outperforms for this problem. Since, every server has equal loads, FLA method is not applicable.

Remark: The efficiency of CE-3Stage is considerably low for the equal load example. This difference can be the result of our implementation technique. However, note that this distinction is not seen in any other example.

**Example 5**

In this example there are three single server queues. The jobs arrive at the first queue according to a Poisson process of rate $\lambda = 1$. 
Table 8: Comparison of performances for Example 4 with equal loads.

<table>
<thead>
<tr>
<th>Method</th>
<th>Estimation</th>
<th>RE</th>
<th>RAT</th>
<th>EFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE-3Stage(7;10000000;4)</td>
<td>7.95E-16</td>
<td>0.0377</td>
<td>1.8208</td>
<td>28.9835</td>
</tr>
<tr>
<td>CE-F(20;100000;0.001)</td>
<td>7.78E-16</td>
<td>0.0058</td>
<td>1.8988</td>
<td>32.1707</td>
</tr>
<tr>
<td>CE-F(20;2000000;0.001)</td>
<td>7.75E-16</td>
<td>0.0049</td>
<td>1.9081</td>
<td>31.4587</td>
</tr>
<tr>
<td>Patch-F(15;100000;0.001)</td>
<td>7.82E-16</td>
<td>0.0204</td>
<td>1.8265</td>
<td>30.7792</td>
</tr>
<tr>
<td>Patch-F(15;50000;0.001)</td>
<td>7.74E-16</td>
<td>0.0221</td>
<td>1.8226</td>
<td>30.9362</td>
</tr>
<tr>
<td>Patch-F(15;10000;0.001)</td>
<td>7.63E-16</td>
<td>0.0392</td>
<td>1.8265</td>
<td>30.7792</td>
</tr>
<tr>
<td>Patch-F(15;5000;0.001)</td>
<td>7.55E-16</td>
<td>0.0329</td>
<td>1.8075</td>
<td>31.0556</td>
</tr>
</tbody>
</table>

Figure 15: Jackson Network with 3 queues.

There are single servers at the queues each and exponentially distributed service times with rates $\mu_1 = 4$, $\mu_2 = 3$ and $\mu_3 = 5$. The routing matrix $R$ of the system is as follows:

$$R = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \\ \frac{1}{3} & 0 & 0 \end{pmatrix}.$$  

Each job can leave the system only after receiving service at the third station. The corresponding Jackson network is depicted in Figure 15.

The traffic intensities are $\lambda_1 = 1.5$, $\lambda_2 = 0.75$ and $\lambda_3 = 1.5$ with corresponding loads $\rho_1 = 0.375$, $\rho_2 = 0.25$ and $\rho_3 = 0.30$. Hence, the system is stable. The failure set is reached whenever at least one of the queues exceeds $n = 30$. Hence, we have 29791 states and 157500 transitions with nonzero probability. For the final estimation by importance sampling we used $10^6$ samples. We compare the estimations with the algorithm CE-3-Stage($k; n; l_0$) mentioned in the previous example. All results in Table 9 are averages of 5 repetitions.
Table 9: Comparison of performances for Example 5.

<table>
<thead>
<tr>
<th>Method</th>
<th>Estimation</th>
<th>RE</th>
<th>RAT</th>
<th>EFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE-3-Stage(7:100000:0.05)</td>
<td>7.34E-13</td>
<td>0.0329</td>
<td>1.8174</td>
<td>25.1203</td>
</tr>
<tr>
<td>CE-3-Stage(7:100000:0.07)</td>
<td>7.32E-13</td>
<td>0.0159</td>
<td>1.7837</td>
<td>24.7847</td>
</tr>
<tr>
<td>FLA</td>
<td>6.56E-13</td>
<td>0.0062</td>
<td>1.8761</td>
<td>27.0516</td>
</tr>
<tr>
<td>CE-D(15:25000:0.05)</td>
<td>8.74E-16</td>
<td>0.4853</td>
<td>1.6670</td>
<td></td>
</tr>
<tr>
<td>CE-D(15:25000:0.10)</td>
<td>5.17E-15</td>
<td>0.3685</td>
<td>1.6597</td>
<td></td>
</tr>
<tr>
<td>CE-D(15:25000:0.25)</td>
<td>3.87E-15</td>
<td>0.6077</td>
<td>1.6394</td>
<td></td>
</tr>
<tr>
<td>CE-D(10:100000:0.10)</td>
<td>6.71E-16</td>
<td>0.1797</td>
<td>1.7028</td>
<td></td>
</tr>
<tr>
<td>Patch-D(15:25000:0.10)-10</td>
<td>2.22E-13</td>
<td>0.3954</td>
<td>1.5994</td>
<td>22.7849</td>
</tr>
<tr>
<td>Patch-D(15:25000:0.25)-10</td>
<td>4.08E-13</td>
<td>0.1165</td>
<td>1.6730</td>
<td>23.3132</td>
</tr>
<tr>
<td>Patch-D(15:10000:0.25)-5</td>
<td>1.72E-13</td>
<td>0.1695</td>
<td>1.6569</td>
<td>23.8863</td>
</tr>
<tr>
<td>Patch-D(8:25000:0.25)-5</td>
<td>2.87E-13</td>
<td>0.1074</td>
<td>1.6771</td>
<td>23.6915</td>
</tr>
<tr>
<td>CE-F(15; 500:0.05)</td>
<td>6.60E-13</td>
<td>0.0036</td>
<td>1.9073</td>
<td>27.3627</td>
</tr>
<tr>
<td>CE-F(15; 500:0.15)</td>
<td>6.60E-13</td>
<td>0.0033</td>
<td>1.9129</td>
<td>27.3359</td>
</tr>
<tr>
<td>CE-F(15; 500:0.001)</td>
<td>5.90E-13</td>
<td>0.0262</td>
<td>1.7790</td>
<td>25.8970</td>
</tr>
<tr>
<td>CE-F(7;5000:0.15)</td>
<td>6.59E-13</td>
<td>0.0028</td>
<td>1.9214</td>
<td>27.4325</td>
</tr>
<tr>
<td>Patch-F(15; 500:0.05)-10</td>
<td>6.60E-13</td>
<td>0.0048</td>
<td>1.8869</td>
<td>26.7790</td>
</tr>
<tr>
<td>Patch-F(15; 500:0.05)-15</td>
<td>6.59E-13</td>
<td>0.0036</td>
<td>1.9085</td>
<td>27.2608</td>
</tr>
<tr>
<td>Patch-F(15; 500:0.001)-5</td>
<td>6.64E-13</td>
<td>0.0216</td>
<td>1.7810</td>
<td>24.9873</td>
</tr>
<tr>
<td>Patch-F(5;5000:0.05)-15</td>
<td>6.60E-13</td>
<td>0.0031</td>
<td>1.9167</td>
<td>27.4015</td>
</tr>
</tbody>
</table>
Similarly to Examples 1 and 3, we see that the original CE-D algorithm performs poorly, specifically it underestimates the target probability considerably. CE-3-Stage, the algorithm introduced in de Boer et. al. (2004) gives better results, relative error around 1%, compared to CE-D and Patch-D. However, for CE-F, sample size of 500 for 15 iterations is even enough to get relative error between 0.3%, 0.4%. However, unlike in Graph 11 in Section 4.2, \( \alpha = 0.05 \) gives lower relative errors. It is also noteworthy that the results provided by FLA slightly worse that the ones given by CE-F and Patch-F.

In the patching algorithms along with the weighting factor \( \alpha \), the number of patches also plays an important role. As we pointed out in Section 4.2, the results for 6 number of patches are considerably higher than the rest. For instance, if we compare the RE results for patches in clusters of 5 with the rest, we see that they are considerably higher. The best is obtained when we only take 2 patches with clusters of 15. That means, as we increase the number of patches, meaning as we get smaller and smaller clusters, the algorithm gets more and more distracted.

**Example 6**

This example of a four node feed-forward network is taken from Nicola and Zaburnenko (2006). We compare the results of our algorithms to those obtained by the methods of Parekh and Walrand (1989) (indicated by PW), de Boer and Nicola (2002) (SDA), Nicola and Zaburnenko (2006) (SDH), and de Boer et. al. (2004) (CE-3-stage). The results of the PW, SDA and SDH methods are given in Nicola and Zaburnenko (2006).

The network is depicted in Figure 16. Jobs arrive at the first queue according to a Poisson Process with rate \( \lambda = 0.074 \). There are single servers at each queue along with exponentially distributed service times with rates \( \mu_1 = 0.617, \mu_2 = 0.024, \mu_3 = 0.135 \) and \( \mu_4 = 0.15 \), with the routing probability in the first queue equal to 0.1. We are interested in the total-population overflow probability for level of 25. The parameters chosen in this problem are chosen from the region where PW heuristic does not work. This is shown empirically in Nicola and Zaburnenko (2006). All the results given in Table 10 are the average of 5 repetitions. In our methods, in the final implementation of importance sampling to estimate the overflow probability we use \( 10^6 \) sample paths.

Experimental results suggest that CE-F gives the best performance among the four algorithms we introduce in this paper for this example. CE-D and Patch-D do not converge even
Table 10: Comparison of performances for Example 6

<table>
<thead>
<tr>
<th>Method</th>
<th>Estimation</th>
<th>RE</th>
<th>RAT</th>
<th>EFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>PW</td>
<td>1.61E-6</td>
<td>0.0808</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SDA</td>
<td>1.70E-6</td>
<td>0.0011</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SDH</td>
<td>1.71E-6</td>
<td>0.0041</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CE-3-Stage(7;1000000;5)</td>
<td>1.71E-6</td>
<td>0.0173</td>
<td>1.5354</td>
<td>12.2764</td>
</tr>
<tr>
<td>CE-F(30:1000 ;0.25)</td>
<td>1.70E-6</td>
<td>0.0082</td>
<td>1.6826</td>
<td>13.3578</td>
</tr>
<tr>
<td>CE-F(30:3000 ;0.25)</td>
<td>1.68E-6</td>
<td>0.0074</td>
<td>1.6983</td>
<td>13.4429</td>
</tr>
<tr>
<td>Patch-F(10:50000 ;0.001)-2</td>
<td>1.69E-6</td>
<td>0.0087</td>
<td>1.6731</td>
<td>12.5455</td>
</tr>
<tr>
<td>Patch-F(20:10000 ;0.05 )-2</td>
<td>1.68E-6</td>
<td>0.0090</td>
<td>1.6676</td>
<td>12.6939</td>
</tr>
<tr>
<td>Patch-F(20:10000 ;0.15 )-2</td>
<td>1.68E-6</td>
<td>0.0097</td>
<td>1.6579</td>
<td>12.5621</td>
</tr>
</tbody>
</table>

with 500000 sample paths for 20 iterations. Implementations with more than 500000 sample paths cause the EFF performance measure to decrease due to the fact that the relative error is not low and the running time increases considerably. As observed in the previous Jackson Network examples, CE-F gives more stable estimations than Patch-F. Both CE-F and Patch-F provides better estimations than PW. There is also not much difference in the performance between CE-3-Stage and Patch-F, Patch-F giving slightly better results. However, Patch-F, CE-F and CE-3-Stage are worse compared to SDA and SDH. While comparing these methods we have to keep in mind that the results provided in Nicola and Zaburnenko (2006) are obtained by using $10^6$ replications in each iteration, where the number of iterations is unknown. Note that since there are two servers with maximum load, FLA is not applicable.

In all of the three examples studying Jackson networks we saw that Patch-F does not provide any better estimation if CE-F already converges. Although we do not face convergence problems with Patch-F, the lowest relative error we can get is higher than 1%.
6 Summary & conclusions

In the previous sections we considered state-dependent approach for two types of problems; reliability models and Jackson networks. We observed that for Jackson networks with higher dimensions the classical cross-entropy method does not give good results. However, if we look at these problems in a more detailed way, it is well known that the transition probabilities in reliability models depend on the state whereas in Jackson networks the transition probabilities are the same for specific types of states. That is the main notion we try to use in Section 4. As we have observed unnecessary usage of state dependency may cause

- an enormous increment in the number of decision variables,
- an increase in variance and computation time.

Another general observation is that the patching algorithms for both state and face-dependent methods do not provide any improvement if classical or face-dependent cross-entropy works. However, for Markovian systems where standard cross-entropy is facing some problems with converging, then patch algorithms help cross-entropy method to converge.

Acknowledgement

The authors are grateful to two anonymous referees for their comments and suggestions.

References


