MONTE-CARLO SIMULATION OF MARKOV CHAINS USING A HIGH-LEVEL MODELLING TECHNIQUE

Johann Christoph Strelen
Rheinische Friedrich–Wilhelms–Universität Bonn
Roemerstr. 164, 53117 Bonn, Germany
E-mail: strelen@cs.uni-bonn.de

KEYWORDS

Monte Carlo Simulation, Rare Events, Hybrid Simulation, Markov Chain, Modelling Paradigm.

ABSTRACT

A method for Monte-Carlo simulation of Markov chains is considered. Here, Markov chains are described with a high-level modelling technique called transition classes. State spaces may be very large, Markov chains may be stiff, i.e. rare events may occur, simulations can be executed highly in parallel, and hybrid evaluation is possible.

INTRODUCTION

Monte-Carlo simulation for discrete-event models [Mit82] is a very popular and general technique in model-based performance or reliability evaluation which is easily to apply. However, this simulation has some drawbacks, and problems appear. Some of them are addressed in this paper. Estimators may have a bias, and their variance may be high which leads to large confidence intervals. The last occurs when rare events are present. Ingenious variance reduction methods have been invented [HH64, Fis78, Mit82, Rip87, Ros89, McG92], e.g. conditioning, control variates, discrete time methods [HIS96], hybrid methods [SS84], antithetic variates, importance sampling, the restart technique [VAVA91], injection simulation [vM93], to enumerate a few. Some of them are difficult to apply, some are restricted to certain models, some reduce the variance not so much - new ideas are looked for.

Simulation is inherently costly with respect to computing time. Here, variance reduction is helpful, because it allows to obtain desired accuracy with shorter simulations. An obvious idea to shorten the long response times of simulations is parallelization. But this seems to be difficult for discrete event simulation - since years, this is a hot topic of research, but a general solution was not yet found.

It is well known that Markov chains are very useful for model-based evaluation, but here, too, many problems and drawbacks bother the modeller. Closed-form solutions rarely exist, and numerical analysis is by no means trivial [Ste94]. Only exponential holding times are covered in continuous time Markov chains, and constant times in the discrete case. Dependent load processes or general distributions must be modelled indirectly with Markov modulated Poisson processes (MMPP), Coxian or phase-type (PH) distributions, Markovian arrival processes (MAP), or similar. These techniques work, but the state space explodes which, in turn, makes numerical analysis impossible in many cases. This problem can be tackled with approximate methods, e.g. [Str97c]. Moreover, if there are many phases, the Markov chain can become stiff. Stiffness which corresponds with rare events in simulation is another problem when numerical analysis is applied [Ste94, Str89]. Courtois' approximate technique for nearly completely decomposable systems [Cou77] copes with stiffness, the largeness problem is alleviated, and accuracy is the better the stiffer the model is.

In this paper, we consider Markov chain models which may have a large state space; they may include MMPPs to model autoregressive processes, PH distributions or MAPs for general probability distributions, and so on. These Markov chains are analyzed with Monte-Carlo simulation. Thus we obtain a solution for three problems: Markov chains with very large state spaces can be analyzed, many simulation runs can be executed in parallel, and Courtois' method is adapted to simulation models with rare events.

For the definition of the Markov chains we use a high-level modelling technique which we call transition classes. Transition classes structure the design of the Markov models and provide a compact description of the state space and the transition probabilities or rates. They are a suited interface to analysis algorithms: Transition class models can be simulated immediately, it is possible to generate the Markov chain automatically, and transition classes are useful for aggregation/disaggregation techniques because the numbering of the states is not essential, things like row or column interchanges of the transition probability matrix are not needed. Hence, the same model can be analyzed with different techniques, numerical methods for the Markov chain or simulation. Hybrid evaluation is straightforward: parts of a model may be simulated, for other parts state probabilities may be calculated numerically.

PRELIMINARIES

Irreducible Markov chains, homogeneous in time, are considered, usually with finite state space $\mathcal{Z}^T$, $|\mathcal{Z}^T| = n \in I\!N$. Either they are aperiodic and discrete in time, $(\mathbf{Z}(t),\ t \in I\!N)$ (DTMC), or continuous in time, $(\mathbf{Z}(t),\ t \in I\!R)$ (CTMC). They stem for example from stochastic models of computer, communication, or manufacturing systems. These systems consist of interacting subsystems, their *(system) components*. At every moment, each component is in one of its possible *component states*. This is why we denote the whole system states by *state tuples* $\mathbf{z} = (z_1, \ldots, z_K)$, where z_k indicates the state of component k. The states z_k of component k belong to the component state space $\mathcal{S}_k$, for all components $k = 1, \ldots, K$. Hence, the state space is $\mathcal{Z}^T \subseteq \mathcal{S}_1 \times \ldots \times \mathcal{S}_K$.

Example. In a queueing network with blocking, Poisson arrivals, if any, and $\cdot/\mathrm{M}/1/\nu_k/\mathrm{FCFS}$ nodes, the system components are the nodes. Their states are the numbers z_k of jobs in them, $z_k \le \nu_k$, $k = 1, \ldots, K$. Therefore, a system state is given by the $K-$tuple $\mathbf{z} = (z_1, \ldots, z_K)$. In an open queueing network, $\mathcal{Z}^T$ equals $\mathcal{S}_1 \times \ldots \times \mathcal{S}_K$ where $\mathcal{S}_k = [0 : \nu_k]$ is the component state space, and in a closed queueing network

$$\mathcal{Z}^T = \left\{ \mathbf{z} \in \mathcal{S}_1 \times \ldots \times \mathcal{S}_K \mid z_1 + \ldots + z_K = N \right\}$$

is the state space where $\mathcal{S}_k = [\tau_k : \nu_k],\ \tau_k = \max\{0, N - \sum_{j \neq k} \nu_j\}$: A state $\tau_k > 0$ in node k occurs if in no other node space is left. □

A *composed system component* c consists of some components, say components $h, h', \ldots \in \mathcal{H}_c \subseteq [1 : K]$. The states of such composed system components are tuples $\mathbf{j} = (j, j', \ldots) \in \mathcal{N}_c \subseteq \mathcal{S}_h \times \mathcal{S}_{h'} \times \ldots$. Components which are not composed are also termed *elementary components*.

Sometimes virtual system components are also useful, for example a "component" which indicates the number of jobs in an open queueing network.

Aggregates are sets of system states. An aggregate can be defined by the state of a system component, for example the aggregate $\mathcal{Z}_{c,\mathbf{j}}$ is the set of all system states where the component c is in state $\mathbf{j}$, irrespective of the states of the other components:

$$\mathcal{Z}_{c,\mathbf{j}} = \{\mathbf{z} \in \mathcal{Z}^T \mid z_h = j,\ z_{h'} = j', \ldots\}.$$

For c fixed, all aggregates $\mathcal{Z}_{c,\mathbf{j}},\ \mathbf{j} \in \mathcal{N}_c$, form a partition of the state space.

$p^{(t)}(\mathbf{z}) \triangleq \mathrm{P}\{\mathbf{Z}(t) = \mathbf{z}\},\ \mathbf{z} \in \mathcal{Z}^T$, denotes the state probabilities of the Markov chain at time t. These probabilities are collected in the n-vector $\mathbf{p}^{(t)}$. For a DTMC, $p(\mathbf{x}, \mathbf{y}),\ \mathbf{x}, \mathbf{y} \in \mathcal{Z}^T$, are the state transition probabilities. Sometimes, if we are not interested in a particular time, or if steady state probabilities are considered, we omit the specification (t).

Each aggregate $\mathcal{Z}_{c,\mathbf{j}}$ defines a *macro probability* or *aggregate probability* $\mathrm{P}\{\mathbf{Z} \in \mathcal{Z}_{c,\mathbf{j}}\} = \sum_{\mathbf{z} \in \mathcal{Z}_{c,\mathbf{j}}} p(\mathbf{z})$.

TRANSITION CLASSES, A HIGH-LEVEL MODELLING TECHNIQUE

Transition classes (TC) structure the design of the Markov models, provide a concise description of the state space, and even for huge state spaces, the transition probability or generator matrix is defined by a few rules. Transition classes are a suited interface to analysis algorithms: TC models can be simulated immediately, it is possible to generate the Markov chain automatically, and transition classes are useful for aggregation/disaggregation techniques: the numbering of the states is not essential, things like row or column interchanges of the transition probability matrix are not needed.

Other high-level models like queueing networks or stochastic Petri nets can be transformed automatically into TC models.

Markov chain transitions are classified, each transition from a state into another state belongs to one and only one transition class.

A transition class describes some kind of state transitions, e.g. the arrival in a specific node of a queueing network, or the transition of a job from a specific node into another specific node, and so on.

In many interesting models, e.g. performance or reliability models, there is only a moderate number of transition classes due to similarity of transitions and small numbers of states which are reachable in one transition (sparsity). This is important for efficiency.

Transition classes are triplets $\tau = (\mathcal{U}, u, \alpha)$. A transition according to transition class τ from state $\mathbf{y}$ into state $\mathbf{z}$ may occur only if $\mathbf{y} \in \mathcal{U}$, where $\mathcal{U}$ denotes the *source states set* of this transition class. $\mathcal{U} \cap \mathcal{Z}^T$ is the set of possible old states, from where the transitions can occur ($\mathcal{U}$ may additionally contain some infeasible state tuples). The *destination state function* $u : \mathcal{Z}^T \to \mathcal{Z}^T$ defines the new states, i.e. a transition of class τ leads from state $\mathbf{y} \in \mathcal{U}$ into state $u(\mathbf{y})$. u may depend on the old state $\mathbf{y}$. The tuple u consists of functions $u_k,\ u(\mathbf{y}) = \big(u_1(\mathbf{y}), \ldots, u_K(\mathbf{y})\big)$. Transitions of class τ occur with probability or rate $\alpha(\mathbf{y})$. α may depend on the old state $\mathbf{y}$.

Example. We consider a queueing network with repetitive service blocking. It consists of $K \in I\!N$ nodes. In each node $k \in [1..K]$ is a single exponential server, service rate μ_k. Buffer space is available for no more than ν_k jobs. If the model is open, jobs arrive according to a Poisson process with parameter λ_k but only if buffer space is available. When the service of a job in node k is finished the job tries to go to node l with probability $t_{k,l},\ l \in [1 : K]$. If there is no buffer space available, the job remains in node k and instantaneously receives another service (*repetitive-service with random destination policy*). In an open network, a job which is completely served in node k leaves the network with probability $t_{k,0}$. In a closed network, there are $N \in I\!N$ jobs, and we assume without loss of generality $\nu_k \leq N,\ k = 1, \ldots, K$.

The system may change its state due to an arrival at node k from outside. The source state set of this transition class is $\mathcal{U} = \{\mathbf{z} \in \mathcal{Z}^T \mid z_k < \nu_k\}$. That means, there is space left in node k for an arriving job. Only the state of node k changes, the other remain unchanged. Hence the destination state function is given by $u_k(\mathbf{z}) = z_k + 1$, identity otherwise: $u_i(\mathbf{z}) = z_i$ if $i \neq k$. The transition rate is $\alpha = \lambda_k$.

Other state transitions take place when a job of node k leaves the network. The according transition class is defined as follows. The source states set is $\mathcal{U} = \{\mathbf{z} \in \mathcal{Z}^T \mid z_k > 0\}$, the destination state function $u_k(\mathbf{z}) = z_k - 1$, identity otherwise, and the transition rate is $\alpha = \mu_k t_{k,0}$.

The third kind of state transitions occur when a job leaves node k for node $i,\ i \neq k$. In node k, there must be at least one job, and in node i buffer space must be available. Therefore, the source states set is $\mathcal{U} = \{\mathbf{z} \in \mathcal{Z}^T \mid z_k > 0,\ z_i < \nu_i\}$. Only the states of the nodes k and i change, $u_k(\mathbf{z}) = z_k - 1,\ u_i(\mathbf{z}) = z_i + 1$, and the transition rate is $\alpha = \mu_k t_{k,i}$. □

Other examples can be found in [Str97d, Str97b, Str97a].

A TC model consists of $\mathcal{TC}$, the set of transition classes, and a single feasible state $\mathbf{y}$.

Generating the Markov Chain

The state space is generated by enumeration. We present a procedure for that:

procedure $\mathit{EnumerateStates}(\mathbf{y} : \mathit{State})$;
if $\mathbf{y} \notin \mathcal{Z}^T$ **then**
  $\mathcal{Z}^T :=$
    $\mathcal{Z}^T \cup \{\mathbf{y}\}$; {augment the set of states $\mathcal{Z}^T$ }
  for all $(\mathcal{U}, u, \alpha) \in \mathcal{TC}$ {transition classes}
    where $\mathbf{y} \in \mathcal{U}$ **do**
    $\mathit{EnumerateStates}\big(u(\mathbf{y})\big)$;
  od
fi.

The procedure is used as follows:

$\mathcal{Z}^T := \emptyset$; $\mathit{EnumerateStates}(\mathbf{y})$;

Here, $\mathbf{y}$ is a feasible state.

Using transition classes, the state transition probabilities of a DTMC can be expressed as follows:

$$p(\mathbf{y},\mathbf{z}) = \sum_{(\mathcal{U},u,\alpha)\in\mathcal{TC}} I(\mathbf{y}\in\mathcal{U})\, I\Big(u(\mathbf{y})=\mathbf{z}\Big)\,\alpha(\mathbf{y}),$$

$$p(\mathbf{y},\mathbf{y}) = 1-\sum_{(\mathcal{U},u,\alpha)\in\mathcal{TC}} I(\mathbf{y}\in\mathcal{U})\,\alpha(\mathbf{y}),$$

$\mathbf{y}\neq\mathbf{z},\ \mathbf{y},\mathbf{z}\in\mathcal{Z}^T$, where $I:\{\text{true, false}\}\to\{0,1\}$ denotes the indicator function.

An engineer should mainly use very high-level paradigms (programming languages, model description languages, ...). The reasons for that are well known. Only occasionally if the expressive power of such a paradigm hides an important detail, or if a feature of the model can not be modelled properly because a suitable language element is missing, the modeller should be able to go below the uppermost level of abstraction and use the next lower level.

We consider transition classes as universal Markov chain modelling means on the second level from above. We imagine different very high-level modelling tools on top of TC which accept abstract model descriptions, for example queueing networks or stochastic Petri nets, and transform them into transition classes. These are analyzed with different solvers and simulators which accept TC models.

Hence we are developing a tool for quite general queueing networks which accepts abstract model descriptions. Such models are transformed into transition classes which are - in turn - passed to one of the solvers, or are simulated. The transition classes are coded in a readable language, so that a modeller can modify them, or add some transition classes for a model feature which is not covered properly on the uppermost level.

MONTE-CARLO SIMULATION

Monte-Carlo simulation algorithms are proposed for Markov chains. They can be applied to estimate state probabilities or sums of them like component state probabilities. From these probabilities, other performance or reliability measures can be derived as usual.

The Markov chains are defined with transition classes, and the algorithms act immediately on a description of these transition classes. The system states are not enumerated, nor a matrix of transition probabilities or rates is used, only the component states and their transitions are considered instead. Hence, state space explosion does not cause a problem, in general.

We present algorithms which allow immediately for parallel execution: Many simulation runs can be done in parallel; thus the desired accuracy, i.e. small confidence intervals, are obtained earlier. Each run must first pass the transient phase; this introduces inefficiency when the model is stiff because the transient phase here is long and useless if steady state probabilities are to be estimated. But in section 4.2, an algorithm is presented which omits these long transient phases.

A Universal TC Simulation Algorithm

This first simulation algorithm can be applied either for transient state probabilities or for steady state probabilities if the Markov chain is not stiff. In the simulation terminology this means that there are no rare events. We call it the *universal TC simulation algorithm*. Here is assumed that the third elements of the transition classes are transition probabilities.

Universal TC simulation algorithm

```
y := InitialState;
repeat {all simulation steps}
    Sum := 0;
    RandomNumber := UniformBetween(0, 1);
    for all (U, u, α) ∈ TC   {transition classes}  do
        if y ∈ U then
            {transition class is feasible in state y}
            Sum plus α(y);
            if Sum ≥ RandomNumber then
                {transition takes place, new state:}
                y := u(y);
                goto 1;
            fi
        fi
    od;
    1 : {Collect data};
until LastSimulationStep;
{Evaluate collected data}.
```

If the third elements of the transition classes are rates, randomization can be applied. That means, a DTMC is considered instead of the CTMC. The DTMC has the same steady-state probabilities, and the transient probabilities of the CTMC can be calculated from those of the DTMC. The rates $\alpha(\mathbf{z})$ are replaced by transition probabilities $\alpha(\mathbf{z})\Delta$. Δ is a randomization constant which is limited from above, $\Delta\leq 1/\sum_{\tau\in\mathcal{TC}}\max_{\mathbf{z}\in\mathcal{Z}^T}\alpha(\mathbf{z})$. The larger the randomization constant Δ is, the more efficient is the simulation. Usually, $\Delta = 0.99/\sum_{\tau\in\mathcal{TC}}\max_{\mathbf{z}\in\mathcal{Z}^T}\alpha(\mathbf{z})$ is taken. The factor 0.99 omits serious problems with rounding errors, and forces the Markov chain to be aperiodic.

Randomization can also be applied for the calculation of transient state probabilities [Gra91].

Rare Events

In Markov models, rare events can be described as follows. There are sets of states, i.e. aggregates, in which the system remains usually for a long time, compared with the duration of stay in the states. The transitions between two of these aggregates are rare events. We call these aggregates *long-term aggregates*.

In [Str90], we presented a Monte-Carlo method for stiff Markov models which is similar to the simultaneous iteration technique in [Str89]. Here, we develop a method which relies on the idea of Courtois. We call it Courtois-type simulation.

We model long-term aggregates with a special (compound) system component, say 1, with the component state space $\mathcal{N}_1$. A state $\mathbf{j}$ of this component defines in which long-term aggregate the system is; we call these component states *long-term*. These aggregates are denoted $\mathcal{Z}_{1,\mathbf{j}}$, $\mathbf{j}\in\mathcal{N}_1$.

We assume that the transitions between long-term states occur only via a subset of the transition classses which have low transition probabilities or rates. The subset of these *long-term* transition classes is denoted $\mathcal{TC}^{\mathrm{lt}}$. An example for rare events are failures and repairs.

Markov chains with this kind of rare events can be analyzed with Courtois' famous approximation technique [Cou77]. In section 5, we indicate how this technique can be elegantly applied using transition classes.

The following simulation method is based on Courtois' approximation technique, and the estimators are biased according to the approximation error. The simulation is executed in two steps.

Here, we restrict ourselves to CTMC but the method is very similar for DTMC.

First, the short-run dynamics are considered. This is done in the *short-term simulation*. If component state probabilities are to be estimated, for example the probabilities that a component $c > 1$ is in state $\mathbf{l}$, i.e. $\mathrm{P}\{\mathbf{Z} \in \mathcal{Z}_{c,\mathbf{l}}\}$, $\mathbf{l} \in \mathcal{N}_c$, the conditional probabilities

$$r_c^{[\mathbf{j}]}(\mathbf{l}) \triangleq \mathrm{P}\{\mathbf{Z} \in \mathcal{Z}_{c,\mathbf{l}} \mid \mathbf{Z} \in \mathcal{Z}_{1,\mathbf{j}}\},$$

are estimated for all $\mathbf{j} \in \mathcal{N}_1$. We call them the *conditional component state probabilities*. The estimated values are denoted by $\bar{r}_c^{[\mathbf{j}]}(\mathbf{l})$.

This simulation is done with the universal TC simulation algorithm. Here, only the *short-term* transition classes $\tau \in \mathcal{TC}^{\mathrm{st}}$ are applied. This is the set $\mathcal{TC}$ of transition classes without the long-term transition classes, $\mathcal{TC}^{\mathrm{st}} = \mathcal{TC} \setminus \mathcal{TC}^{\mathrm{lt}}$. When the conditional component state probabilities for $\mathbf{j} \in \mathcal{N}_1$ are simulated, the initial state must be in $\mathcal{Z}_{c,\mathbf{j}}$.

Secondly, the long-run dynamics are analyzed. This is done in the *long-term simulation*. Here, we estimate the steady-state probabilities of being within the long-term aggregates, that is the steady-state component probabilities $P\{\mathbf{Z} \in \mathcal{Z}_{1,\mathbf{j}}\}$ of component 1, for all $\mathbf{j} \in \mathcal{N}_1$.

These long-term probabilities are the steady-state probabilities of a Markov chain with the state space $\mathcal{N}_1$ which is defined by the following transtion classes.

From each transition class $\tau = (\mathcal{U}, u, \alpha) \in \mathcal{TC}^{\mathrm{lt}}$, a modified transition class $\tau' = (\mathcal{U}', u', \alpha')$ is derived (or some modified transition classes, as explained later on).

We use a function $up : \mathcal{Z}^T \to \mathcal{N}_1$ where $up(\mathbf{z}) = \mathbf{j}$ iff $\mathbf{z}$ is element of aggregate $\mathcal{Z}_{1,\mathbf{j}}$. This means that the function up indicates for each state the long-term aggregate to which it belongs.

For the moment we assume the following. (*) For all states $\mathbf{z}$ for which the transition class τ can be applied, i.e. $\mathbf{z} \in \mathcal{U}$, and for which the long-term aggregate is the same, say $up(\mathbf{z}) = \mathbf{j}$, the new long-term aggregate is the same, say $\mathbf{l}$, i.e. $up(u(\mathbf{z})) = \mathbf{l}$.

For each state $\mathbf{j} \in \mathcal{N}_1$, $\mathbf{j}$ is in the modified source state set $\mathcal{U}'$ iff $\mathcal{Z}_{1,\mathbf{j}} \cap \mathcal{U} \neq \emptyset$, the modified destination state function is given by $u'(\mathbf{j}) = up(u(\mathbf{z}))$ iff $up(\mathbf{z}) = \mathbf{j}$ for all $\mathbf{z} \in \mathcal{U}$, and the modified rate or transition probability is

$$\alpha'(\mathbf{j}) = \sum_{\mathbf{z} \in \mathcal{Z}_{1,\mathbf{j}} \cap \mathcal{U}} \alpha(\mathbf{z}) P\{\mathbf{Z} = \mathbf{z} \mid \mathbf{Z} \in \mathcal{Z}_{1,\mathbf{j}}\}.$$

The last sum must be evaluated efficiently. If all values $\alpha(\mathbf{z})$ are the same for all $\mathbf{z} \in \mathcal{Z}_{1,\mathbf{j}} \cap \mathcal{U}$, say α, the sum is $\alpha P\{\mathbf{Z} \in \mathcal{Z}_{1,\mathbf{j}} \cap \mathcal{U} \mid \mathbf{Z} \in \mathcal{Z}_{1,\mathbf{j}}\}$. This probability must be estimated in the short-term simulation. If there are some different values $\alpha(\mathbf{z})$ for all $\mathbf{z} \in \mathcal{Z}_{1,\mathbf{j}}$, the modified rate or transition probability is calculated similarly.

The assumption (*) can be omitted as follows. For each new long-term aggregate, i.e. for each different $up(u(\mathbf{z}))$, an individual modified transition class is created instead of the single modified transition class τ'.

Let $\mathcal{TC}^*$ denote the set of modified transition classes. They define the *long-term Markov chain*.

The desired steady-state probabilities $P\{\mathbf{Z} \in \mathcal{Z}_{1,\mathbf{j}}\}$, $\mathbf{j} \in \mathcal{N}_1$, can be estimated with the universal TC simulation algorithm, using the modified transition classes. The estimated values are denoted by $\bar{r}_1(\mathbf{j})$.

Here, the randomization constant can be chosen much larger than in the short-term simulation because the transition rates are small: $\Delta^{\mathrm{lt}} = 0.99/\sum_{\tau' \in \mathcal{TC}^*} \max_{\mathbf{j} \in \mathcal{N}_1} \alpha'(\mathbf{j})$. The fact that this randomization constant is larger than for the short-term simulation is crucial: this makes the long-term simulation efficient.

The probabilities $\mathrm{P}\{\mathbf{Z} \in \mathcal{Z}_{c,\mathbf{l}} \cap \mathcal{Z}_{1,\mathbf{j}}\}$ are estimated with $\bar{r}_c^{[\mathbf{j}]}(\mathbf{l})\bar{r}_1(\mathbf{j})$, $\mathbf{j} \in \mathcal{N}_1$, $\mathbf{l} \in \mathcal{N}_c$, and $\mathrm{P}\{\mathbf{Z} \in \mathcal{Z}_{c,\mathbf{l}}\}$ with $\sum_{\mathbf{j} \in \mathcal{N}_1} \bar{r}_c^{[\mathbf{j}]}(\mathbf{l})\bar{r}_1(\mathbf{j})$, $\mathbf{l} \in \mathcal{N}_c$.

If the aggregates $\mathcal{Z}_{c,\mathbf{l}}$ are chosen such that $|\mathcal{Z}_{c,\mathbf{l}} \cap \mathcal{Z}_{1,\mathbf{j}}| \le 1$ holds for all $\mathbf{j}$ and all $\mathbf{l}$, all state probabilities are estimated. In this special case, our two-step simulation technique corresponds with Courtois' method.

HYBRID ANALYSIS

A TC model with rare events is evaluatd in two steps. First, for each component state $\mathbf{j} \in \mathcal{N}_1$, a TC model is analyzed which is defined by a state $\mathbf{z}$ with $up(\mathbf{z}) = \mathbf{j}$ and the transition classes $\mathcal{TC}^{\mathrm{st}}$. This is the short-term analysis. Secondly, for the long-term analysis, the TC model with the transition classes $\mathcal{TC}^*$ and a single state $\mathbf{j} \in \mathcal{N}_1$ is evaluated.

When in this two-step technique all states are considered and all values are calculated with generated Markov chains, Courtois' method is executed. This is accomplished without considering neither the transition probability matrix nor the numbering of the states. Usually, the numbering of the states must be changed properly, rows and columns of the matrix must be interchanged in order to obtain a special block structure, and transition probabilities must be modified before the calculations begin. All this is not needed if transition classes are used.

Hybrid analysis is immediately possible because a TC model can be transformed automatically into a Markov chain which is solved as usual, or the TC model can be simulated directly. For example, the short-term analysis can be done with the universal TC simulation algorithm, and the long-term probabilities are calculated with the generated Markov chain.

In general, if a complex model is decomposed into submodels, for hybrid analysis, some of them can be simulated, the others are numerically evaluated. The structure of the model which is given by the transition classes, supports the process of decomposition.

EXAMPLE

The proposed algorithms were tried with a queueing network with queues in tandem. A small model with 5832 states has 3 queues with 5 buffers for jobs and 3 servers each, a large one with 24.3 million states has 5 queues with 9 buffers and 3 servers each. Servers may fail, and a failed server will be repaired. Jobs arrive at node 1 as Poison process with parameter 1/s, and the service time is exponentially distributed with rate 1.5/s in node 1, and rate 2/s otherwise. The blocking policy is repetetive service. $\bar{D}$ denotes the mean operational time of a server before failure, but a server fails only if at least one intact server remains in the node. Different values $\bar{D}$= 20 s, 200 s, 5.56 h, 11.6 days for the small model and $\bar{D}$= 333 s, 11.6 days, or 115.7 days for the large model correspond with different degrees of stiffness. At most one server is repaired at the same time, in the first node with the lowest number of functioning servers. The mean repair time is $\bar{D}/10$ for the small model and 20 s, 18.5 h, or 7.7 days, respectively, for the large model.

The long-term states are defined by the numbers of operational servers in each node.

The small TC model was analyzed with simultaneous vector iteration (SVI) [Str89] and Courtois' method, using generated states and transition probabilities, and both models with simulation, namely with the universal TC simulation algorithm, with Courtois-type simulation, and with a hybrid algorithm. For the hybrid analysis, the long-term probabilities were calculated numerically from the Markov chain which was generated from the modified long-term transition classes. For each estimated probability, an interval was determined which contains the median of the estimator with probability 96.8%. To this end, 6 independent simulation runs were performed. The simulations were continued until all these intervals were shorter than 0.02.

The table shows computing times on a Pentium with 166 MHz. The figures indicate that Courtois-type simulation is very efficient compared with usual simulation, and the hybrid technique is even more efficient. The reader may note that simple simulation is untractable for the very stiff models due to intolerable simulation times.

It is interesting to mention that each Courtois-type simulation and each hybrid simulation consists mainly in a lot of independent simulation runs, the short-term simulations. For the small model, these are 486 runs, for the large model 1458. Therefore, parallelization is easy to accomplish and is expected to result in a speed-up of nearly the number of processors, up to the quoted numbers of runs.

Numerical Solution			
Model	$\bar{D}$	SVI	Courtois
Small	20 s	179 s	95 s
	200 s	376 s	84 s
	5.56 h	636 s	86 s
	11.6 days	820 s	85 s

Simulation				
Model	$\bar{D}$	Courtois-type	Hybrid	Univ. TC Alg.
Small	20 s	692 s	87 s	316 s
	200 s	138 s	68 s	3069 s
	5.56 h	97 s	48 s	>43.3 h
	11.6 days	118 s	61 s	
Large	333 s	3735 s	1012 s	15.5 h
	11.6 days	7333 s	1066 s	
	115.7 days	5663 s	869 s	

CONCLUSION

The combined use of Markov chains and Monte-Carlo simulation avoids some problems and drawbacks which appear when each of these techniques is used alone, namely state space explosion, rare events, stiffness, and difficulties when simulation algorithms are to be parallelized. Together with the proposed high-level modelling technique, namely transition classes, this combination is easy to realize, and hybrid evaluation is possible. The estimators of the rare event simulation technique are biased, but the bias is the lower the stiffer the model is. The bias can be omitted; this is a topic of future research, as well as the determination of confidence intervalls.

References

[Cou77] P. J. Courtois. *Decomposability, queueing and computer system applications.* Academic Press, London, 1977.

[Fis78] G. S. Fishman. *Principles of Discrete Event Simulation.* John Wiley and Sons, New York, 1978.

[Gra91] W. K. Grassmann. Finding transient solutions in Markovian event systems through randomization. In *Numerical Solution of Markov Chains*, pages 357–372. Marcel Dekker, New York, Basel, Hong Kong, 1991.

[HH64] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods.* Methuen and Co. Ltd., London, 1964.

[HIS96] A. Hordijk, D. Iglehard, and R. Schassberger. Discrete time methods for simulating continuous time Markv chains. *Advances in Applied Probability*, 8:772–788, 1996.

[McG92] C. McGeoch. Analyzing algorithms by simulation: Variance reduction techniques and simulation speedups. *ACM Computing Surveys*, 24(2):195–212, 1992.

[Mit82] I. Mitrani. *Simulation Techniques for Discrete Event Systems.* Cambridge University Press, Cambridge, 1982.

[Rip87] B. D. Ripley. *Stochastic Simulation.* John Wiley and Sons, New York, 1987.

[Ros89] S. M. Ross. *Introduction to Probability Models.* Academic Press, San Diego, California, fourth edition, 1989.

[SS84] J. G. Shanthikumar and R. G. Sargent. A unifying view of hybrid simulation/ analytic models and modeling. *Operations Research*, 31:1030–1052, 1984.

[Ste94] W. J. Stewart. *Introduction to the numerical solution of Markov chains.* Priceton University Press, 1994.

[Str89] J. Ch. Strelen. A fast simultaneous iteration technique for the analysis of Markov chains. Interner Bericht II/89/1, Institut für Informatik, Universität Bonn, 1989.

[Str90] J. Ch. Strelen. Massiv parallele Algorithmen für die Simulation von Markov-Modellen. In F. Breitenecker, I. Troch, and P. Kopacek, editors, *Fortschritte in der Simulationstechnik*, volume 1, pages 175–180. F. Vieweg, Braunschweig, 1990.

[Str97a] J. Ch. Strelen. Approximate analysis of queueing networks with Markovian arrival processes and phase type service times. In K.Irmscher, Ch.Mittasch, and K.Richter, editors, *Messung, Modellierung und Bewertung von Rechen- und Kommunikationssystemen*, pages 55–70, Berlin, Offenbach, 1997. VDE-Verlag.

[Str97b] J. Ch. Strelen. Approximate disaggregation-aggregation solutions for general queueing networks. In A.R. Kaylan and A. Lehmann, editors, *Proc. of the ESM 97 Conference*, pages 773–778. Society for Computer Simulation, 1997.

[Str97c] J. Ch. Strelen. Approximate product form solutions for Markov chains. *Performance Evaluation*, pages 87–110, 1997.

[Str97d] J. Ch. Strelen. Loss queueing networks with bursty arrival processes and phase type service times: Approximate analysis. In D.Kouvatsos, editor, *Fifth IFIP Workshop on Performance Modelling and Evaluation of ATM Networks - Participants Proceedings*, pages 87/1–87/10. University of Bradford, 1997.

[VAVA91] M. Villen-Altamirano and M. Villen-Altamirano. RESTART: A method for accelerating rare events simulations. In J.W. Cohen and C.D. Pack, editors, *Queueing, Performance and Control in ATM (ITC-13)*, pages 71–76, Amsterdam, 1991. North-Holland.

[vM93] Aad P.A. van Moorsel. *Performability Evaluation Concepts and Techniques.* PhD thesis, University of Twente, 1993.

Johann Christoph Strelen was born in Wiesbaden, Germany, in 1941. He received the Dipl.-Mat. and Dr. rer. nat. degrees in mathematics and the Habilitation degree in Computer Science from the Technische Hochschule Darmstadt, Germany, in 1968, 1973, and 1981, respectively. There he was affiliated to the Computing Center (1968-1973), and assistant at the Computer Science Department (1974-1982). In 1973 he was a post doctoral fellow at the IBM Scientific Center, Grenoble, France. Since 1982 he has been a Professor of Computer Science at the Rheinische Friedrich–Wilhelms–Universität Bonn, Germany. His reseach interests include performance evaluation, distributed systems, and simulation. Dr. Strelen is a member of the Gesellschaft für Informatik (GI) and the Gesellschaft für angewandte Mathematik und Mechanik (GAMM).