

ACCELERATING DISCRETE-EVENT SIMULATION VIA STATE SPACE REDUCTION

Stefan Heller,
DaimlerChrysler Research,
FT3/AS
70546 Stuttgart
Germany

Michael Luber, Graham Horton,
Computer Science Department,
University of Erlangen,
Cauerstrasse 6, 91058 Erlangen
Germany

KEYWORDS

State space reduction, discrete-event simulation.

ABSTRACT

In this paper we describe some approaches for accelerating discrete-event simulations of a certain class of model. The simulation model is assumed to be in the form of a state space and our goal is to reduce the computation time necessary for the simulation. To this end, we use techniques for reducing the size of the model and also show an approach to accelerating the simulation mechanism per se.

INTRODUCTION

Our approach is based on the observation that in many simulations, the results required by the user are based on a (possibly small) subset of the states of the state space. We may thus partition the state space into two sets of states: those which contribute to the representation of the result, and those which do not. One example would be the simulation of a simple queue, whereby the goal is to estimate the probability that the queue is empty. Here, the result is represented by one state only. We denote those states whose solution values are not needed to represent the result as reducible, and the others as irreducible. Our goal is to reduce the number of states in the model as far as possible by aggregating certain combinations of neighbouring reducible states. This reduction in the number of states will in turn lead to reduction in the number of events to be simulated, which in turn should reduce the computation time of the simulation. One further possibility is to unify parallel arcs in the model. This leads to a reduction in the number of random numbers that must be generated by the simulation program and thus also to a speedup.

One further possibility for accelerating the simulation which we consider is obtained by reconfiguring the simulator to treat branching probabilities and delays separately in those parts of the model which contain a fork. This requires a pre-processing step in which branching probabilities are computed from the distribution functions and then these are adapted accordingly. This allows the simulation program to determine which branch of the fork to choose quickly, after which only one delay must be computed. This

technique also allows us to save computation time during the simulation.

Naturally, these modifications to the model must not change its behaviour in any way (from the point of view of the irreducible states). A large part of the work described here is devoted to deriving modifications with the corresponding properties.

This paper is structured as follows: First, methods for reducing the size of the state space are presented. Then, the technique for separating branching probabilities and delays is explained. This allows us to then develop a further method for reducing the state space. Each of these methods is explained and assessed. This assessment is based on a simulation program which represents the probability distributions numerically using 80 sampling points. This allows us to compute the distributions numerically.

ACCELERATION TECHNIQUES

We consider a class of simulation models which can be represented as a state space, as illustrated in Figure 1. States are represented by circles and events by directed arcs. Each arc represents a change of state and has a probability distribution associated with it. We will denote states by S_i and probability distributions by X_i .

A simulation of this state space model proceeds as follows: Starting from the current state, a random number is generated for each arc leaving the state according the corresponding probability distribution. This number represents the time at which the event is scheduled to occur. Then, the arc with the shortest associated firing time is selected and the event occurs. This procedure is then repeated for the new state.

Evidently, the time required for the simulation to reach a steady state will increase with the number of states in the model. This provides the motivation for our acceleration technique, which attempts to remove unneeded states from the model.

Unifying Parallel Arcs

The left-hand side of Figure 2 shows the two states S_1 and S_2 which are connected by two parallel arcs with delays described by the distributions X_1 and X_2 . During the simulation, each change of state from S_1 to S_2 requires a random number to be generated from each distribution X_1 and X_2 and the smaller of these values to be chosen to fire. Effectively, the state change occurs

according to a distribution which is given as the minimum of X_1 and X_2 .

We can thus replace the system on the left by that on the right, whereby Y is defined as follows:

$$Y = \min(X_1, X_2)$$

The distribution Y can be computed as follows:

$$P(Y > t) = P(X_1 > t) \cdot P(X_2 > t)$$

$$1 - P(Y < t) = (1 - P(X_1 < t)) \cdot (1 - P(X_2 < t))$$

$$1 - F_Y = (1 - F_{X_1}) \cdot (1 - F_{X_2})$$

$$F_Y = 1 - ((1 - F_{X_1}) \cdot (1 - F_{X_2}))$$

The general form for n parallel arcs can be written as:

$$F_Y = 1 - \prod_{i=1}^n (1 - F_{X_i})$$

In the replacement model, in which the parallel arcs have been unified, the simulation program only needs to evaluate one arc instead of n . Our simulator needed 77 μs to unify two arcs represented numerically with 80 sampling points. The time needed to sample a distribution function was 1.33 μs . In one example, which contained 5 parallel arcs, this yielded an overall time of 6.7 μs . After reduction, the sampling only required the 1.33 μs for the single remaining arc. A simple computation shows that the reduction will pay off after about 60 samples have been made.

Sequence

Another structure in the state space which can be easily reduced is the sequence of three states, shown on the left of Figure 3. Here we assume that the states S_1 and S_2 are reducible.

Our goal is to aggregate the states S_1 and S_2 to form the new state $S_{1,2}$, (Figure 3, right). This requires us to determine a new probability distribution Y as a function of X_1 and X_2 . In this case, Y is obtained by convolution of X_1 and X_2 as follows:

$$Y = \text{conv}(X_1, X_2)$$

$$Y(t) = \int_0^{\infty} F_{X_1}(\tau) \cdot F_{X_2}(t - \tau) d\tau$$

Our simulator requires 0.68 ms for the numerical convolution with 80 sampling points. During the simulation, only one distribution must be sampled, whereas previously there were two. In other words, 1.33 μs of computation time can be saved, resulting in a payoff after about 523 traversals of the sequence. This number is relatively high, because numerical convolution is an expensive operation. However, this is offset by savings in the statistical computations associated with the visits to each state of the model, which can be considerable.

Merge

The merge structure is similar to the sequence. As shown in Figure 4 (left), it can be considered to be a pair of sequences (S_1, S_3, S_4) and (S_2, S_3, S_4) .

We assume that the states S_1, S_2 and S_3 are all reducible. In this case, the two sequences can be reduced using convolution, as described in Section 2.2, whereby two new arcs with distributions Y_1 and Y_2 , are created (Figure 4 right). If no other arcs enter state S_3 , then the arcs X_1, X_1 and X_3 may be deleted from the model.

This reduction step requires the computation of two convolutions. As described in Section 2.2, these each require 0.68 ms. Each traversal of the merge during the simulation therefore yields a saving of 1.33 μs , whereby the situation holds as for the sequence. However, the gain obtainable in the statistical computation in the case of the merge is halved, since only state and one arc are eliminated.

Branching Probabilities

One further possibility for accelerating the simulation is to separate the computation of the delay and the branching probabilities in the case of the fork or split. State space models often contain structures such as that shown in Figure 5 (left).

In a traditional simulation, when the model is in state S_2 , a sample is computed for both random variables X_2 and X_3 . Then, the smaller of the two samples is chosen and the simulator changes to state S_3 or S_4 accordingly. A different approach would be to first pre-compute the probability for each arc being chosen. This allows the simulator to choose an arc each time very quickly, since only a simple probability must be computed. Having chosen an arc at random, the simulator then only needs to sample one (instead of two) random variates in order to determine the delay in the event. The probability p_2 for choosing X_2 can be computed as follows:

$$p_2 = P(X_2 < X_3)$$

$$p_2 = \int_{\tau=0}^{\infty} f_{X_2}(\tau) \cdot P(F_{X_3} > \tau) d\tau$$

The formula can be easily generalised to the case of n arcs:

$$p_j = P((X_j < X_1) \wedge \dots \wedge (X_j < X_{j-1}) \wedge (X_j < X_{j+1}) \wedge \dots \wedge (X_j < X_n))$$

$$p_j = \int_{\tau=0}^{\infty} f_{X_j}(\tau) \cdot \left[\prod_{i=0 \wedge i \neq j}^n P(F_{X_i} > \tau) \right] \cdot d\tau$$

This change in strategy requires modifications to the distribution functions X_1 and X_2 , since now they are no longer competing with each other (we have already chosen one of them a priori).

The appropriate density function can be computed as follows:

$$f_{Y_2}(t) = f_{X_2}(t) | (X_2 < X_3)$$

$$\Rightarrow f_{Y_2}(t) = f_{X_2}(t) \cdot \frac{(1 - F_{X_3}(t))}{P(X_2 < X_3)}$$

This formula can also be generalised for the case of n arcs:

$$f_{Y_j}(t) = f_{X_j}(t) | ((X_j < X_1) \wedge \dots \wedge (X_j < X_{j-1})$$

$$\wedge (X_j < X_{j+1}) \wedge \dots \wedge (X_j < X_n))$$

$$\Rightarrow f_{Y_j}(t) = f_{X_j}(t) \cdot \frac{\prod_{i=1 \wedge i \neq j}^n (1 - F_{X_i}(t))}{P_{X_j}}$$

This pre-processing step for the split structure requires 272 μ s on the computer system used. This included computing the branching probabilities for two arcs and the computation of the new probability density functions. The original simulation required the sampling of two general random variables, which needed 2.66 μ s. After the modification, the simulator needs only to sample a uniform distribution for choosing an arc and then one general distribution to determine the delay. This required 1.83 μ s in total. Thus, a saving of 0.83 μ s is made for each traversal of the state. In this example, the state only possessed two outgoing arcs; states with a higher number would allow a greater saving to be made.

Split

Another structure which allows a reduction of the size of the state space is the split or fork, shown in Figure 6 (left). It is based on the same structure as that of Section 2.

The split structure on the left of Figure 6 is transformed into the structure shown on the right. Two new arcs with distributions Y_1 and Y_2 are introduced. The first step of the transformation is to separate the computation of the branching probabilities from that of the delay, as was described in Section 2.4. The branch containing X_2 and X_3 is replaced by the probabilities p_1 , p_2 and the modified densities. After these modifications, the new arcs Y_1 and Y_2 can be inserted. In order to ensure an unchanged flow of probability through the states S_3 and S_4 , the probabilities must be associated with the densities as shown in Figure 6 (right). The distributions of the new arcs can be computed by convolution of X_1 with the modified densities X_2 and X_3 as described in Section 2.2. Finally, the arc X_1 can be removed. The state S_2 and the arcs X_2 and X_3 can also be removed, as the state S_2 can no longer be reached.

The costs for this transformation consist of the costs for computing the probabilities, modifying the densities and computing the two convolutions. A value of 0.952 ms was measured for our simulator. Before the transformation, three random variables had to be sampled, costing 3.99 μ s per traversal of the structure.

After the modifications, only one random variable must be sampled, plus the generation of a uniform random number. This required only 1.58 μ s. Thus a saving is made after approximately 400 traversals of the structure.

EXAMPLE

In order to test our techniques, we generated a state space with 1024 states, of which 30 % were chosen at random to be irreducible. These states were connected by 2048 arcs, whose delays were chosen from varying distributions, whose parameters were chosen at random. To solve this original model to steady state required 0.158 sec of computation time.

The model was then transformed as follows. First, all parallel arcs were unified. Then the outgoing probabilities were computed for all states with more than one outgoing arc. Then, all sequence, merge and split structures were simplified. Then, any resulting parallel arcs were once again eliminated. This procedure was repeated until no more simplifications were possible. The entire reduction process required 8.71 seconds. After the reduction, the simulation time only 0.0793 sec. This example shows that a savings can be achieved after about 100 replications.

SUMMARY AND OUTLOOK

The techniques described in this paper for accelerating discrete-event simulations can be applied to models in the form of a state space. These include, for example, Markov chains. The important property is that the state of the model can be characterised entirely by the state it is currently in and the time spent there. This property is not, for example, true of stochastic Petri nets, and for models such as these it not yet clear to what extent these techniques could be applied.

The techniques shown here require strongly varying investments for the necessary modifications and thus to varying conditions for achieving savings. For example, the numerical convolution operation is very expensive, which means that a large number of traversals is necessary. On the other hand, the separation of branching probability and delay computations is comparatively cheap, and thus a saving can be gained more quickly. We can draw the general conclusion that all the proposed modifications will result in savings in computation time when a high degree of accuracy (i.e. a large number of replications or long simulations) is required.

One disadvantage of the methods presented here is the fact that they cannot be applied to all classes of simulation models. In the future, therefore, in addition to developing further acceleration techniques, we will attempt to apply them to more general situations.

FIGURES

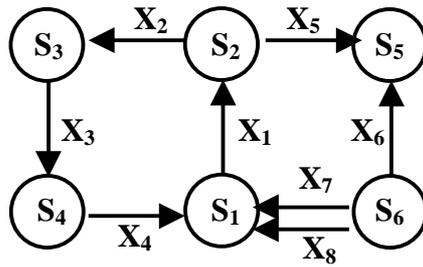


Figure 1: State Space

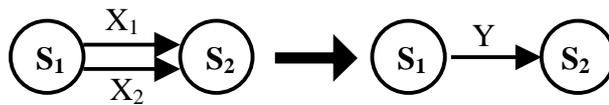


Figure 2: Reduction of Parallel Arcs



Figure 3: Sequence of States

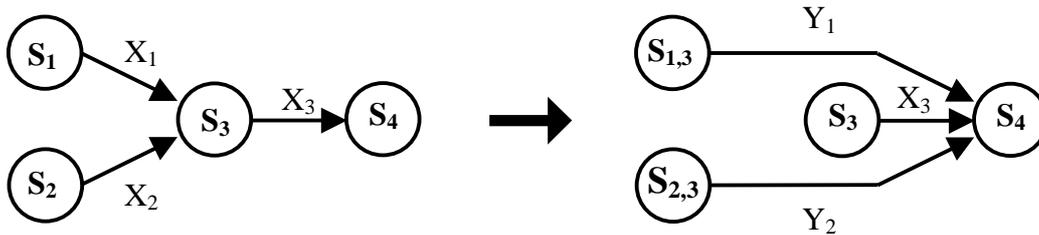


Figure 4: Merge

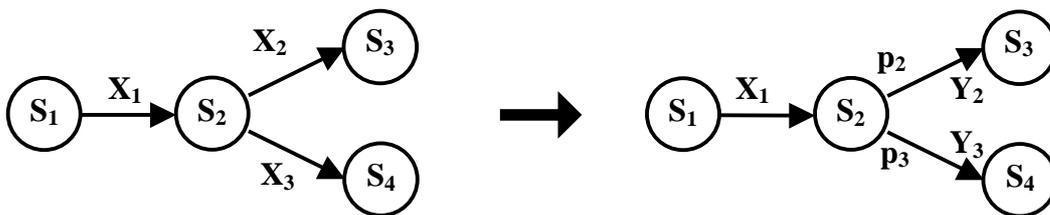


Figure 5: Branching Probabilities

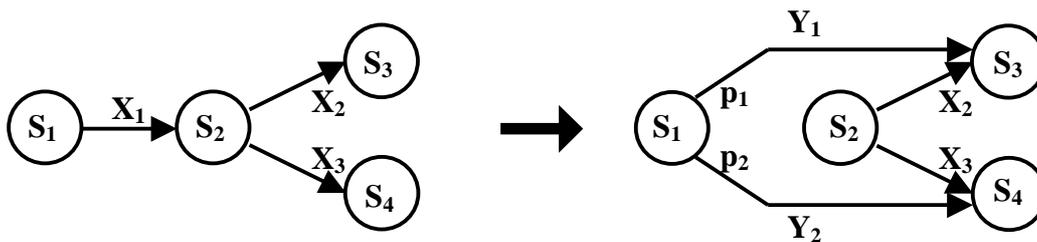


Figure 6: Split