SPNP: Stochastic Petri Nets. Version 6.0

Christophe Hirel, Bruno Tuffin and Kishor S. Trivedi

Center for Advanced Computing and Communication
Department of Electrical and Computer Engineering
Duke University, Durham, NC 27708-0291, U.S.A.
{chirel,tuffin,kst}@ee.duke.edu

1 Introduction

The Stochastic Petri Net Package (SPNP) [2] is a versatile modeling tool for solution of Stochastic Petri Net (SPN) models. The SPN models are described in the input language for SPNP called CSPL (C-based SPN Language) which is an extension of the C programming language [8] with additional constructs which facilitate easy description of SPN models. Moreover, if the user does not want to describe his model in CSPL, a Graphical User Interface (GUI) is available to specify all the characteristics as well as the parameters of the solution method chosen to solve the model.

Earlier versions of SPNP provided the capabilities of automatically generating and solving Markov reward models starting with the Stochastic Reward Nets (SRNs) [2], an extension of Petri Nets. The new version of SPNP provides major extensions in three directions:

1. Non-Markovian SPNs as well as Fluid stochastic Petri Nets (FSPNs) [3] can be described and solved.
2. Besides the analytic numeric solution of Markovian models discrete-event simulation is now available.
3. A user-friendly GUI interface is now available.

A number of important Petri net constructs such as marking dependency of firing times, variable cardinality arc and enabling functions (or guards) [2] facilitate the construction of models for complex systems. Also available are priorities as well as different resampling policies (preemptive resume, preemtive repeat identical and preemptive repeat different) when the (enabled) transition is disabled by the firing of a competitive transition and later becomes enabled again or when the firing time of a still enabled transition is affected by the firing of another transition. The package also allows logical analysis on the Petri net whereby any general assertions defined on the Petri net are checked for each marking of the net. Hooks are available to solve a set of interconnected models via fixed-point iteration.

The distributions for the firing times of transitions currently implemented in SPNP are the following: exponential, immediate, constant, uniform, geometric, Weibull, truncated normal, lognormal, Erlang, hyperexponential, hypoexponential, Pareto, truncated Cauchy, Poisson, binomial, gamma and beta. This list is going to be suplemented by the negative binomial, Cox2, triangular, loglogistic, and defective Exponential with mass at origin. Another extension being considered is the use of samples given in a file by the user.

2 Solution methods

The solution methods can be divided in two main categories, the numeric-analytic methods and the simulation methods. A high level view of methods can be seen in Figure 1.

2.1 Analytic-numeric methods

For steady-state evaluation (of a CTMC or a DTMC), the user can choose among Steady-State SOR (Successive Overrelaxation)[13], Steady-State Gauss-Seidel[13], and Steady-State Power method [15]. Even if SOR

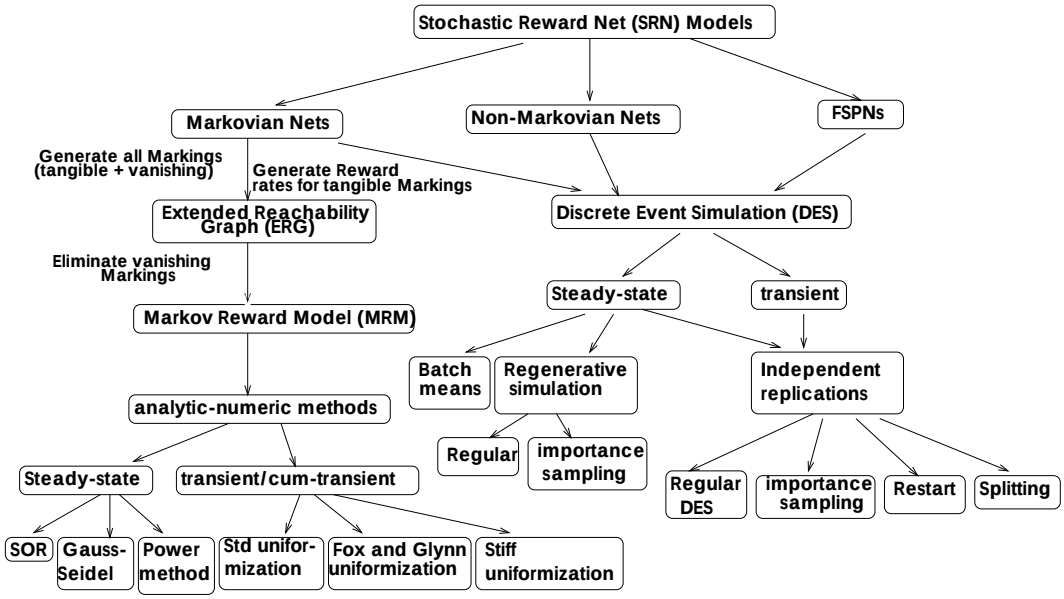

Fig. 1. Chart for SPNP

is usually the fastest method, its convergence is not always guaranteed. In such cases Gauss-Seidel may be found to converge. The Power method is guaranteed to converge but the rate of convergence is generally slower than SOR.

For transient-state solution for the CTMC, the possible methods are standard uniformization and uniformization using the Fox and Glynn method for computing the Poisson probabilities. A steady-state detection in transient analysis is also possible and is very useful for stiff models [11].

Parametric sensitivity analysis allows the user to evaluate the effect of changes in an input parameter on the output measures. This is useful in system optimization and bottleneck analysis.

2.2 Simulation

Simulation is used to solve the model when analytic-numeric methods fail because the state space is too large or because the restrictions on the model are too strong. It is used in particular for non-Markovian SRNs or on (Markovian or non-Markovian) FSPNs.

Different simulation methods available are the following:

- Simulation using independent replications [4] to compute cumulative or average instantaneous values up to a fixed simulation time.
- Simulation using batches [4] to compute steady state measures. A single run is then considered and cut in several blocks (assumed to be independent) to obtain the confidence interval.
- Importance splitting techniques (Restart and splitting) which are speed-up methods to estimate the probablities of rare events [14]. The basic idea is to split the simulation path when given thresholds are reached to make the rare event occur more often.
- Importance sampling [5, 6, 12] to speed-up the simulation using independent replications. The idea here is to modify the distributions (or probabilities) of transitions to make it more suitable for the analysis. Of course, the (new) system is then biased, but this bias (also called *likelihood ratio*) can be computed.

– Regenerative simulation [5, 7] to estimate steady-state measures. In this case the expectation of a reward function is estimated by considering different regenerative cycles between two returns in one given state and dividing the estimated accumulated reward during the state by the expectation of the length of a cycle. Markov regenerative Petri nets have been introduced in [1].

We are currently implementing the following simulation methods:

– Regenerative simulation with importance sampling [7, 12], which puts together the advantages of regenerative simulation and the ones of importance sampling.
– Thinning with independent replications [10], to compute cumulative or average instantaneous values up to a fixed simulation time. Thinning is a very suitable method to simulate certain stochastic processes including non-homogeneous Poisson processes.
– Thinning with importance sampling [9].
– Thinning with batches.

3 iSPN 6.0

We have developed an integrated environment for modeling using Stochastic Petri Nets, named iSPN. Careful consideration was given to the design and implementation of iSPN to facilitate the creation of SPN models. iSPN increases the power of SPNP by providing a means of rapidly developing stochastic reward nets. Input to SPNP is specified using CSPL C based SPN Language, but iSPN removes this burden from the user by providing an interface for graphical representation of the model.

The major components of the iSPN interface are a Petri net editor which allows graphical input of the stochastic Petri nets and an extensive collection of visualization routines to analyze output results of SPNP and aid for debugging. Each module corresponds to a page in the software. iSPN provides a high level input format to CSPL which provides great flexibility to users.

The previous development used the scripting language Tcl Tool Command Language, developed by Prof. John Ousterhout of U.C. Berkeley, and extension Tk, a toolkit for X windows. All the modifications made on the current version of SPNP and the development of the SHARPE gui using Java at Duke by the same team were good reasons to consider a new evolution for iSPN. The current version is also designed with Java, integrated in a commun gui with the SHARPE gui.Thus the output of a SPNP model can be used as an input of a SHARPE model. The hierarchy feature from SHARPE is reinforced by this commun design.

References

1. H. Choi et al. Markov Regenerative Stochastic Petri Nets. *Performance Evaluation*, 20(1-3):337–357, 1994.
2. G. Ciardo, J. K. Muppala, and K. S. Trivedi. SPNP: Stochastic Petri Net Package. *Proc. of 3rd International Workshop on Petri Nets and Performance Models*, pages 142–150, Kyoto, Japan, Dec. 1989.
3. G. Ciardo, D.M. Nicol, and K.S. Trivedi. Discrete-Event Simulation of Fluid Stochastic Petri-Nets. *IEEE Trans. on Soft. Eng.*, 25(2):207–217, 1999.
4. Computer Science Department. College of Willian and Mary. *On the Simulation of Stochastic Petri Nets.*
5. G.S. Fishman. *Monte Carlo: Concepts, Algorithms and Applications.* Springer-Verlag, 1997.
6. P. W. Glynn and D. L. Iglehart. Importance Sampling for Stochastic Simulations. *Management Science*, 35(11):1367–1392, November 1989.
7. A. Goyal et al. A Unified Framework for Simulating Markovian Models of Highly Dependable Systems. *IEEE Trans. on Computers*, 41(1):36–51, Jan. 1992.
8. S. P. Harbison and G. L. Steele Jr. *C — A Reference Manual.* Prentice-Hall, 3 edition, 1991.
9. P. Heidelberger et al. Bounded Relative Error in estimating transient Measures of Highly Dependable Non-Markovian Systems. *ACM Trans. on Modeling and Computer Simulation*, 4(2):137–164, April 1994.
10. P.A.W. Lewis and G.S. Shedler. Simulation of nonhomogeneous Poisson processes by thinning. *Naval Res. Logist. Quart.*, 26:403–413, 1979.

11. M. Malhotra et al. Stiffness-tolerant methods for transient analysis of stiff Markov chains. *Microelectron. Relib.*, 34(11):1825–1841, 1994.
12. V. F. Nicola et al. Fast Simulation of Highly Dependable Systems with General Failure and Repair Processes. *IEEE Trans. on Computers*, 42(12):1440–1452, December 1993.
13. W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1994.
14. B. Tuffin and K.S. Trivedi. Implementation of importance splitting techniques in stochastic petri net package. IPDS 2000.
15. W. B. van den Hout. *The Power-Series Algorithm: A Numerical Approach to Markov Processes.* Tilburg University, 1996.