

The performability tool *P'ility*

Lucia Cloth, Boudewijn R. Haverkort

Faculty of Electrical Engineering, Mathematics and Computer Science
Centre for Telematics and Information Technology — University of Twente

[lucia,brh]@ewi.utwente.nl

Abstract

The performability distribution is the distribution of accumulated reward in a Markov reward model (MRM) with state reward rates. Since its introduction, several algorithms for the numerical evaluation of the performability distribution have been proposed. Many of these algorithms only solve specialised MRMs, for example, with only 0 and 1 as reward rates or compute the expected value of the accumulated reward. The *P'ility* tool implements four algorithms that allow for the computation of the performability distribution in its full generality.

1 Performability

The concept of performability as the joint evaluation of performance and dependability has been introduced by Meyer in 1980 [5]. One way to evaluate performability uses Markov reward models (MRMs): CTMCs equipped with reward rates. The reward rates represent cost or bonus, or can be employed to mark states (0-1 rewards). Reward accumulates over time; its distribution $\Pr\{Y(t) \leq y\}$ is also called the *performability distribution*. Many performability measures are directly derived from this distribution [9].

The performability distribution is described by the set of hyperbolic partial differential equations [6]

$$\frac{\partial \Upsilon(t, y)}{\partial t} + \mathbf{R} \cdot \frac{\partial \Upsilon(t, y)}{\partial y} = \mathbf{Q} \cdot \Upsilon(t, y),$$

where $\Upsilon_{ij}(t, y)$ is the probability of an accumulated reward of at most y at time t and ending in MRM state j , having started in state i ; \mathbf{R} is the diagonal matrix containing the state rewards and \mathbf{Q} is the generator matrix of the MRM. If $\mathbf{p}(0)$ is the initial distribution of states, the performability distribution is

$$\Pr\{Y(t) \leq y\} = \sum_i p_i(0) \sum_j \Upsilon_{ij}(t, y).$$

In [1] and [2] we discussed five algorithms for the numerical solution.

The now available software tool *P'ility* offers for the first time an implementation of the four really worthwhile algorithms. Additionally, the expected reward rate at a given

time or in steady state, and the expected accumulated reward can be computed.

2 Experiments in *P'ility*

P'ility allows the user to define one or more *experiments*. An experiment consists of a Markov reward model, the time and reward parameter(s) one wants to compute a performability measure for, and the needed algorithm with its special options.

2.1 Markov reward models

P'ility does not include functionality to specify MRMs. It can read MRMs from the file system, that are either stored in the .tra/.rew format [4], or in its own simple XML-based format. Internally, MRMs consist of two parts, a simple sparse representation of the generator matrix and a vector storing the reward rates. Depending on the algorithm chosen for evaluation, these data structures have to be copied and altered. Especially reordering is often necessary.

The initial distribution also has to be specified. Either a single initial state is indicated or a general initial distribution is stored in a file.

2.2 Time- and reward parameter

Four types of parameter combinations can be chosen:

1. Single fixed values t and y . This results in the single probability $\Pr\{Y(t) \leq y\}$.
2. Fixed t and an interval $[y_1, y_2]$ for y together with the number N of sampling points. The result is a list of values $\Pr\{Y(t) \leq y_1 + i \cdot \frac{y_2 - y_1}{N}\}$ for $i = 0, \dots, N$ which can be used for a two-dimensional plot. *P'ility* can derive automatically a suitable interval $[y_1, y_2]$. The first (quick) approach just determines the interval, where the probability is going to be non-zero but below 1. However, it might be the case that the probability mass is concentrated in a much smaller area. With the second approach, the interval is computed in such a way that it contains a given fraction γ of the total probability mass.

3. Fixed y and an interval $[t_1, t_2]$ for t together with the number N of sampling points which can be used for a two-dimensional plot. The result is a list of values $\Pr\{Y(t_1 + i \cdot \frac{t_2-t_1}{N}) \leq y\}$ for $i = 0, \dots, N$. Also the interval $[t_1, t_2]$ can be determined automatically.
4. Intervals for $[t_1, t_2]$ for t and $[y_1, y_2]$ for y together with the numbers of sampling points N_t and N_y . The result is a list of values $\Pr\{Y(t_1 + i \cdot \frac{t_2-t_1}{N_t}) \leq y_1 + j \cdot \frac{y_2-y_1}{N_y}\}$ for $i = 0, \dots, N_t$ and $j = 0, \dots, N_y$ which can be used for a three-dimensional plot. If one of the intervals is fixed the other can be determined by the means described for cases 2) and 3).

2.3 The algorithms

Four different algorithms are available for the numerical computation of the performability distribution. Some of them actually compute the complementary distribution $\Pr\{Y(t) > y\}$ which can easily be transformed to $\Pr\{Y(t) \leq y\} = 1 - \Pr\{Y(t) > y\}$.

1. The first algorithm, developed by Sericola [8] is based on uniformisation. It computes the performability distribution up to a predefined accuracy level. This algorithm is the default algorithm to be used, however, for MRMs with large numbers of transitions its computational costs are prohibitive.
2. The second algorithm, first presented by Qureshi and Sanders [7], explores explicitly the possible paths in the MRM. If the probability of a path drops below a given threshold, it is discarded. Afterwards a bound for the accuracy of the result can be indicated. Depending on the structure of the MRM this algorithm can yield very fast and accurate results but can also exhibit prohibitive run times.
3. The third algorithm discretises both, time and accumulated reward. It was invented by Tijms and Veldman [10]. No error bound can be given for the result. This result is only suitable if the reward rates have integer values. If this is not the case, the rates have to be rescaled to integers, possibly leading to prohibitive run times.
4. The fourth algorithm, the so-called Markovian approximation, has been developed by us [3]. It discretises the accumulated reward in such a way that one ends up with a pure CTMC. For this CTMC the transient distribution is computed using uniformisation and translated into the performability distribution. This algorithm is also suited for large MRMs, however, there is no statement about the accuracy of the results.

Depending on the chosen algorithm, the user has to specify a set of parameters, such as the accuracy level or the number of discretisation steps.

After the user has defined an experiment (MRM + parameters + algorithm), the tool can predict the number of floating point operations needed for actually running this experiment. For algorithms 1), 3) and 4) the prediction is quite accurate, whereas for algorithm 2) it is only a rough estimate.

The actual results are simply written to formatted files. For visualisation purposes the user has to rely on external plotting software (such as gnuplot).

3 Summary and future work

P'ility is a complete tool for the evaluation of performability measures. It offers its users a variety of algorithms to choose from, depending on the measure and the size of the MRM under study. It provides both, a graphical and a command line user interface. Hence, it can be used for the tentative exploration phase of MRMs but also for the extensive computation of results in batch mode. For the future we plan an extension that allows for the analysis of parametrised MRMs.

References

- [1] L. Cloth. *Model Checking Algorithms for Markov Reward Models*. PhD thesis, University of Twente, 2006.
- [2] L. Cloth and B. R. Haverkort. Five performability algorithms: A comparison. In *Proceedings of the Markov Anniversary Meeting (MAM'06)*, pages 39–54. Boson Books, 2006.
- [3] B. R. Haverkort, L. Cloth, H. Hermanns, J.-P. Katoen, and C. Baier. Model checking performability properties. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)*, pages 102–112. IEEE Press, 2002.
- [4] H. Hermanns, U. Herzog, K. U., V. Mertsiotakis, and M. Siegle. Compositional performance modeling with the TIPPTool. *Performance Evaluation*, 39(1–4):5–35, 2000.
- [5] J. F. Meyer. On evaluating the performability of degradable computing systems. *IEEE Transactions on Computers*, 29(8):720–731, 1980.
- [6] K. R. Pattipati, R. Mallubhatla, V. Gopalakrishna, and N. Viswanatham. Markov-reward models and hyperbolic systems. In *Performability Modelling*, pages 83–106. John Wiley & Sons, 2001.
- [7] M. A. Qureshi and W. H. Sanders. Reward model solution methods with impulse and rate rewards: an algorithm and numerical results. *Performance Evaluation*, 20(4):413–436, 1994.
- [8] B. Sericola. Occupation times in Markov processes. *Communications in Statistics — Stochastic Models*, 16(5):479–510, 2000.
- [9] R. Smith, K. S. Trivedi, and A. Ramesh. Performability analysis: Measures, an algorithm and a case study. *IEEE Transactions on Computers*, 37(4):406–417, April 1988.
- [10] H. Tijms and R. Veldman. A fast algorithm for the transient reward distribution in continuous-time Markov chains. *Operations Research Letters*, 26:155–158, 2000.