PENELOPE

dependability evaluation and the optimization of performability

Hermann de Meer and Hana Ševčíková

Department of Computer Science, TKRN, University of Hamburg
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
email: {demeer@,sevcikov@ro2.}informatik.uni-hamburg.de

Abstract

A new performance and performability modeling tool is introduced in this paper. PENELOPE is the first tool which incorporates evaluation and optimization algorithms. It is the result of a combination between the performability modeling concept and Markov decision theory. Different algorithms are adopted and included in the tool under the unifying paradigm of reconfigurability as the basis for adaptation and optimization. In addition to transient and steady-state performability measures, also transient and stationary control functions can be computed and graphically presented. Model specification and specification of transient or stationary control functions can be separately performed and deliberately combined with each other. Besides providing a new modeling paradigm, the tool supports model creation, experimentation, storage and presentation of results by means of an easily usable interface and an integrated model data base system.

1 Introduction

During the last decade there has been an increasing interest in performability modeling [14]. The development of software tools supporting performability modeling and analysis has been an active area of research.

Metaphor [4], developed 1984, was the first tool for performability modeling. It addressed only a limited set of Markov models; input and output were textual. The tool SPNP [2] is a stochastic Petri nets package, which supports specification, generation, and solution of continuous time Markov chains (CTMCs). Steady-state, transient, and cumulative performability measures, defined by Markov reward models (MRMs), can be computed. The model description is done via C. The tool UltraSAN [3] is based on stochastic activity networks. In addition to numerical algorithms, UltraSAN provides also simulation methods. Surf-2 [1] has been developed for dependability and performability evaluation. Models can either be MRMs or generalized stochastic Petri nets. SPNP, UltraSAN and Surf-2 provide an output in tabular form. Additionally, Surf-2 allows a graphical representation of the results. Deterministic and general type time distributions are complementing exponential distribution in DSPNexpress [7] and in other work. Many more tools do exist, most of them being covered in the overview paper by Haverkort and Niemegeers [5].

This paper describes the new software package PENELOPE [10]. PENELOPE is the first tool which incorporates *evaluation* and *optimization* algorithms. It can be applied for the integrated computation of perfor-

mance/performability functions and of optimal control strategies. It constitutes the implementation of the concept of a combination between performability modeling and Markov decision theory [13].

In addition to transient and steady-state *performability measures*, also transient and steady-state *control functions* can be computed and graphically presented. Model specification and specification of transient or steady-state control structure can also be separately performed and deliberately combined with each other. This allows the immediate comparison of the impact of various control strategies for a given model on the resulting performance measures.

The specification of control strategies is built on the paradigm of reconfigurability [8, 9]. The intuition behind is, decisions must be made to reconfigure or not to reconfigure a system from one state to another in order to optimize a given performance/performability measure. The mapping of the reconfiguration options on internal model representations suitable for the optimization algorithms and the application of appropriate algorithms is hidden from the user.

PENELOPE provides a friendly usable interface for model generation and experimentation. In particular, the creation of model variations is supported as well as the execution of series of experiments and the integrated presentation of the results of those experiments. This includes the presentation of performability functions and, in particular, the presentation of control strategies. No interference of the user is necessary to prepare the graphical presentation of control strategies and performability functions. The control functions are automatically related to the original specification of reconfigurability options and series of experiments. Thus, the execution of an optimization study can be considered as a meta-experiment that comprises many single experiments which are related to each other.

This paper is organized as follows. Section 2 contains a description of the general functionality of PENELOPE. Section 3 illustrates by means of a simple example important features of the tool, such as model generation, experiment set-up and execution, or presentation of results. Section 4 concludes the paper.

2 Description of PENELOPE

PENELOPE is based on the theory of *extended Markov reward models* (EMRMs) [8, 9]. It offers a modeling methodology that combines MRMs and Markov decision processes [13].

PENELOPE allows to create parameterized models of arbitrary finite size and to provide automatically the models with concrete values. To each parameter an arbitrary set of concrete values can be allocated. For each possible combination of parameter values, PENELOPE performs an experiment. Whole series of experiment can thus be easily specified and executed.

Additionally, PENELOPE offers the following functionalities: automated checking of model consistency, mechanism for hierarchical and iterative modeling, graphical preparation of experimental results, interactive preparation of computed strategies, printing of models and results for documentation purposes.

The mathematical background of EMRMs will be briefly presented in this section as far as it is necessary to introduce the analysis techniques of our tool. Let $Z = \{Z(t), t \geq 0\}$ denote a CTMC with finite state space $\mathbf{C}$. To each state $s \in \mathbf{C}$ a real-valued *reward rate* $r(s)$, $r : \mathbf{C} \to \mathbb{R}$, is assigned, such that if the CTMC is in state $Z(t) \in \mathbf{C}$ at time t, then the *instantaneous reward rate* of the CTMC at time t is defined as $X(t) = r_{Z(t)}$. In the time horizon $[0, t)$ the *total reward* $Y(t) = \int_0^t X(\tau) d\tau$ is accumulated. Note that $X(t)$ and $Y(t)$ depend on $Z(t)$ and on an initial state. The distribution function $\Psi(y, t) = P(Y(t) \leq y)$ is called the *performability*. For ergodic models the instantaneous reward rate and the time averaged total reward converge in the limit to the same overall reward rate $\lim_{t\to\infty} E[X(t)] = \lim_{t\to\infty} \frac{1}{t} E[Y(t)] = E[X]$.

EMRMs provide a framework for the combined evaluation and optimization of reconfigurable systems by introducing some new features for MRMs. A *reconfiguration* arc, which can be placed between an arbitrary Markov state and any other state in a model, specifies an optional, instantaneous state transition that can be *controlled for optimization*. Zero, one, or more reconfiguration arcs may originate from any Markov state. The resulting strategy provides optimal reconfiguration decisions for each option in the model. At every point of time a different decision is possible for each reconfiguration arc. A strategy $\mathbf{S}(t)$ comprises a tuple of decisions for all options in the model at a particular point of time t, $0 \leq t \leq T$. Strategies can be time dependent, $\mathbf{S}(t)$, or time independent, $\mathbf{S} = \mathbf{S}(t)$.

The so-called *branching* states provide another feature of EMRMs. No time is spent in such states, but a pulse reward may be associated with them. The introduction of branching states has motivation similar to the introduction of immediate transitions to stochastic Petri nets [2].

Two types of methods are offered for computation of optimal strategies and performance functions:

- **Transient Optimization**, where the expected accumulated reward $E[Y_i(t)]$ is used as an optimization criterion. The algorithm which has been introduced in earlier work [8, 9] is applied for an analysis within a finite period of time $[0, t)$. Transient optimization for acyclic CTMC was introduced and its correctness proved by Lee & Shin [6]. The algorithm was adopted by de Meer [8] and extended to general CTMC. Additionally, the correctness of EMRM aproach in terms of Markov decision processes was proved. The algorithm is based on Taylor series[1].
- **Stationary Optimization**, which is performed for an infinite time horizon $[0, \infty)$. As optimization criteria, we distinguish between *time averaged mean total reward in steady-state*, $E[X] = E[X_i] = \lim_{t\to\infty} \frac{1}{t} E[Y_i(t)]$ for all i, where $E[X_i]$ is independent of initial state i for a particular strategy, and the *conditional accumulated reward until absorption*, $E[Y_i(\infty)] = \lim_{t\to\infty} E[Y_i(t)]$, which is computed for non-ergodic models containing absorbing and transient states. $E[Y_i(\infty)]$ is dependent on initial

[1] Taylor series for transient analysis (without optimization) has also later been discussed by Stewart [12].

state i. The optimization itself is performed by deployment of variants of *value iteration* or *strategy iteration* type methods [8], relying on numerical algorithms such as Gaussian elimination, Gauss-Seidel iteration, successive over-relaxation (SOR), and the power method. All these methods are implemented in the tool and can be deliberately chosen for a computation.

With $X^{\mathbf{S}}$, $Y_i^{\mathbf{S}}(\infty)$, $Y_i^{\mathbf{S}(t)}(t)$ denoting performability measures gained *under strategy* $\mathbf{S}$ *or* $\mathbf{S}(t)$ respectively, a strategy $\hat{\mathbf{S}}(t)$ is optimal, iff

$$\begin{aligned}
E[Y_i^{\hat{\mathbf{S}}(t)}(t)] &\geq E[Y_i^{\mathbf{S}(t)}(t)]\ \forall \mathbf{S}(t)\ \forall i && \text{transient optimization,}\\
E[X^{\hat{\mathbf{S}}}] &\geq E[X^{\mathbf{S}}]\ \forall \mathbf{S} && \text{stationary optimization (ergodic),}\\
E[Y_i^{\hat{\mathbf{S}}}(\infty)] &\geq E[Y_i^{\mathbf{S}}(\infty)]\ \forall \mathbf{S}\ \forall i && \text{stationary optimization (nonergodic).}
\end{aligned}$$

In addition to optimization, PENELOPE offers procedures for computations of performability measures under fixed deliberately eligible strategies:

- **Simulation**, where model evaluations can be simulated under fixed strategies.
- **Transient analysis**, where transient numerical evaluations can be carried out under fixed transient or stationary strategies.
- **Stationary analysis**, where stationary numerical evaluations can be carried out under fixed stationary strategies.

3 A Simple Example

3.1 Description

The features of PENELOPE will be demonstrated by means of a simple example. In packet-switched networks there exists the special case that applications, generating mixed traffic (data, video, voice etc.) with different quality-of-service requirements, are communicating via a single switching node. This is a typical situation in particular in a local-area environment, where the switch may be used as a PBX(Private Branch Exchange). One of the most important problems to be solved with respect to traffic management is related to congestion control. Congestion control for real-time traffic by selectively discarding packets has been investigated by Schulzrinne et al. [11]. We adopt a similar scenario in a simplified setting as depicted in Fig. 1(left). n classes of packet streams are distinguished, where the classes differ from each other with respect to response time limits and loss thresholds. Packets are continuously fed into the system according to independent Poisson processes with arrival rates $\lambda_1, ..., \lambda_n$. The packets are processed with rate μ using the service strategy *first come first served*. The problem of congestion control through discarding of last arriving packets will be investigated.

Keeping it as simple as possible, a system with $n = 2$ arrival streams will be considered. The different response time limits[2] of both classes are assumed to be

[2]We define response time as waiting time plus service time.

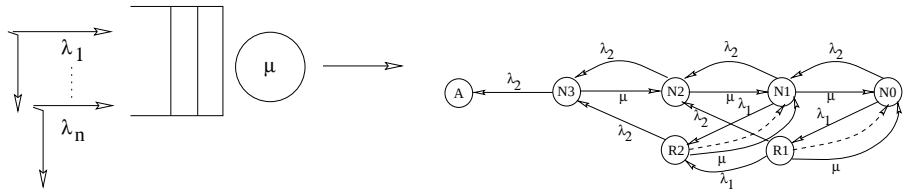


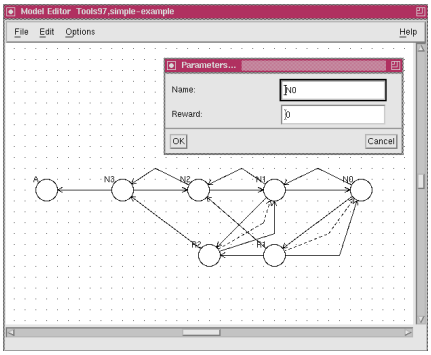
Figure 1: A queueing system with n classes of arriving packets (left) and the baseline model as an EMRM (right).

proportional to the mean service time $\frac{1}{\mu}$ with different factors. The first class is *loss-tolerant* but more delay-sensitive. While having no limit for losses, the mean response time limit of these packets is assumed to be $\frac{2}{\mu}$ time units. There is no use in keeping class one packets if their expected response time is larger than that, that is, if there are already at least two packets in queue upon their arrival. The second class is highly loss-sensitive, no packet of this class may get lost. The less restrictive mean response time limit, on the other hand, is assumed to be $\frac{3}{\mu}$ units of time. Packets of the second class can be accepted as long as there are at most two packets in the queue ahead of them. Since loss of class two packets cannot be tolerated, service is assumed to be immediately stopped if such an event occurs. The admission policies, optimizing different reward functions, will be studied in this simple example for purpose of demonstrating the features of the tool.

The baseline model is depicted as an EMRM in Fig. 1(right). In states N_j, $j \in \{0,1,2,3\}$, j packets are in the system. In states R_i, $i \in \{1,2\}$, i packets are in the system and the last arriving packet is a packet of the first class. Whenever the system is in state R_i, a decision has to be made whether the last arriving packet should be dropped or not. In case of a positive decision the system is reconfigured to the corresponding state N_{i-1}. The reconfiguration options are graphically indicated by dashed lines in the EMRM of Fig. 1(right). If loss of a second class packet occurs, then the absorbing state A is reached and all further arriving packets, regardless of their class, are considered to be lost. First, we will investigate the strategy, which maximizes the mean throughput in terms of the number of packets served in finite time or before service interruption occurs. Then, the strategy which minimizes the expected number of lost packets and differences in the strategies will be discussed.

In PENELOPE the resulting model can be easily created with the help of the graphical model editor as illustrated by Fig. 2, which is provided as part of the user interface.

$E[Y_i(t)]$, as a measure of accomplishment, is used as a criterion of optimization. Series of experiments for different reward structures will be investigated. It is interesting to compare the impact of various control strategies on the resulting performance measures. As it will turn out, transient strategies yield better performance results than stationary strategies do. In addition, the difference will be investigated between using the mean accumulated reward in finite



state	reward rate
N_0	0
N_i	μ
R_j	μ
A	0

$i \in \{1,2,3\}, j \in \{1,2\}$

Figure 2: Model editor of PENELOPE.

time as an optimization criterion and applying the mean accumulated reward until absorption as a criterion.

3.2 Experiments

3.2.1 Maximization of Throughput

To specify the throughput per unit of time, the reward rate is defined as the service rate *times* the number of active processors in each state. The resulting reward structure is attached to Fig. 2. The set of parameters is summarized in Tab. 1. Users can define parameter values either directly in the model editor or in the parameter set editor, which allows one to explicitly specify an arbitrary set of concrete values to be substituted for formal parameters. One can equivalently specify parameter value-ranges and step widths, where for each of the resulting values a computation will automatically be executed. Usage of the parameter set editor is exemplified by Fig. 3(left).

parameter	value	meaning
λ_1	0.02	arrival rate of the class 1
λ_2	0.02	arrival rate of the class 2
μ	$[0.01, ..., 0.13]$	service rate

Table 1: Set of parameters.

Using the experiment editor, which is shown in Fig. 3(right), complete series of experiments can be specified and executed.

Transient Optimization In our example transient optimization is performed in the time horizon $[0, 6000)$. Assuming one unit of time corresponding to 10 msec, the total arrival rate $\lambda_1 + \lambda_2$ results in 1 packet/250 msec= 4/sec. The time horizon would cover 1 min. The resulting dynamic control strategy, which maximizes the throughput, is depicted in Fig. 4(left). With respect to

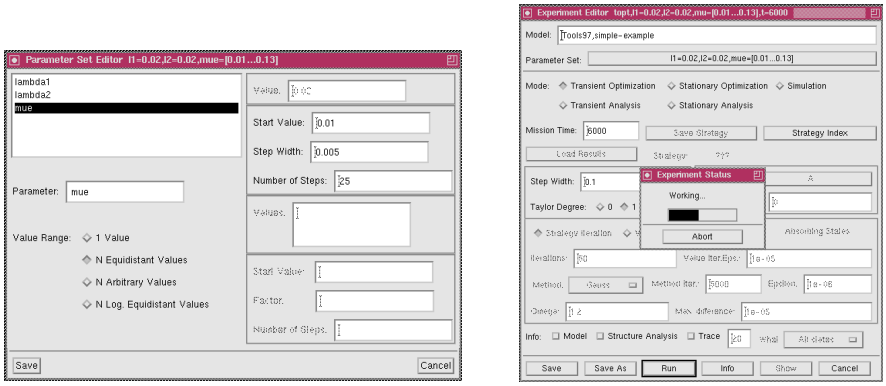


Figure 3: Parameter set editor (left) and experiment editor (right).

state R_2, the two-dimensional decision space, given by the covered set of parameter values of μ and the considered time interval, is partitioned into two regions, which are divided by the curve with the label '$R_2 \rightarrow N_1$' attached to it. In the region *above* the curve the indicated strategy applies: A message of class one should be discarded in favor of potentially arriving class two messages. If, however, conditions are given such that the current situation is classified to be in the region *underneath* the curve, the alternative strategy should be applied in state R_2 and class one message should not be discarded in order to maximize the overall throughput. The region is indicated by label '$R_2 \rightarrow R_2$'.

With respect to state R_1, the two-dimensional decision space comprises a single region. Neither time nor service rate affect the curve labeled '$R_1 \rightarrow R_1$'. This means that the strategy '*do not discard the last arriving packet in state* R_1' is applied in the whole decision space.

The curves, also referred to as switching curves, represent for each state the instants of time where the strategy switches with respect to the remaining time. For example, if the service rate is $\mu = 0.08$, that is, 1 packet/125 msec, and if the time to go is 500 units, that is, 5 sec, an arriving packet of the first class should be accepted. The point $(0.08, 500)$ is located in the region $R_2 \rightarrow N_1$ and $R_1 \rightarrow R_1$. The throughput would decrease by dropping a packet in such a situation. The strategy reveals the following monotony: the smaller the service rate, the less packets of the first class are accepted. Of course, the loss risk of a second class message is higher with a decreasing service rate.

In Fig. 5, the throughput for the initial states N_0, N_1, N_2, and N_3 is depicted as a function of the service rate. The reward functions are computed assuming the optimal strategy. The throughput behaves in a way which might be anticipated: the higher the service rate, the higher the throughput. In the right part of Fig. 5, it is to be seen that the system reaches the highest throughput with the initial state N_1. Note that in PENELOPE one can arbitrarily select strategy/performance curves to be graphically presented. In Fig. 5, four curves were selected.

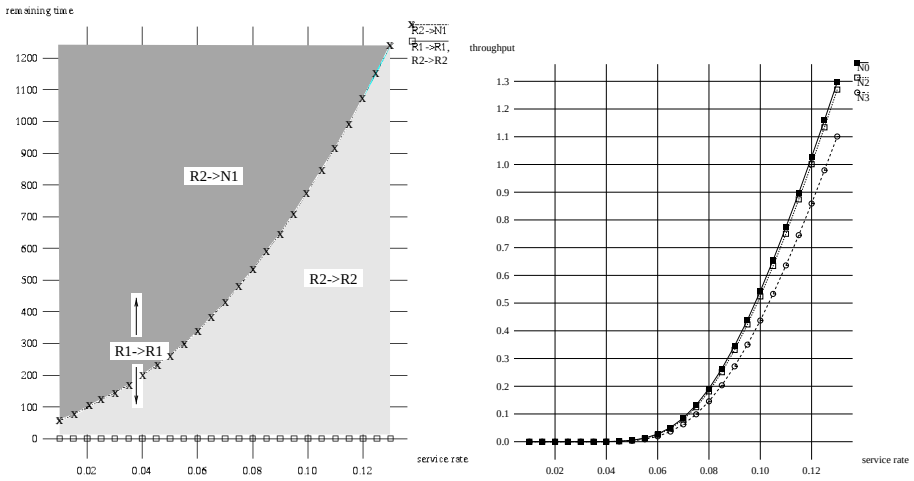


Figure 4: Transient strategies for maximizing the throughput (left) and difference graph of throughput computed under different strategies (right).

In what follows, the impact of arrival rates λ_1 and λ_2 on the optimal control strategy will be investigated, while $\mu = 0.04$ will be kept fixed. For the computations relating to the left part of Fig. 6, the arrival rate λ_1 of class one messages is varied, while $\lambda_2 = 0.02$ is kept fixed. In the right part, the arrival rate λ_2 of the second class messages is varied, while $\lambda_1 = 0.02$ is kept fixed. It is interesting to note that the strategy switching curves decrease with increasing arrival rates in both cases. But the strategy as a function of λ_2 is much more sensitive to an increase of the parameter value than in the first case. If λ_2 is significantly larger than λ_1 the option to discard class one message in state R_2 is "nearly always" selected. Note that in state R_1 a message should never be discarded.

Stationary Optimization The stationary optimization is performed for the same parameter set as in Tab. 1. The strategy iteration with Gauss-Seidel

service rate	**strategy**
$[0.01, ..., 0.13]$	$R_1 \rightarrow R_1$ $R_2 \rightarrow N_1$

Table 2: Optimal stationary strategy.

computation method is chosen due to the small size of the model. $E[Y_i(\infty)]$ is used as a criterion of optimization. The resulting strategy is summarized in Tab. 2. It applies for the whole interval of considered service rates. If one compares the stationary strategies in Tab. 2 with the transient strategies in Fig. 4(left), it can be seen how they relate to each other. In the long run

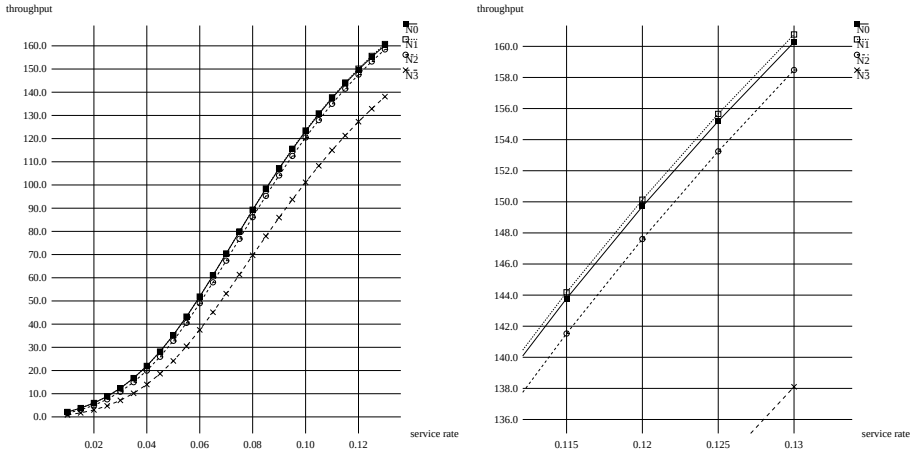


Figure 5: Throughput as a function of the service rate.

messages should always be discarded in state R_2 and never in state R_1. But in state R_2 the optimal strategy is strongly time-dependent for shorter periods of time.

Transient Analysis PENELOPE allows to save and modify computed strategies, or to create arbitrary new strategies, and to apply them in computations. Using this feature, it is possible to execute series of experiments under different strategies and to compare their impact on performance. To compare, for example, the impact of optimal transient strategies (OTS) and optimal stationary strategies (OSS) on the performance in our scenario, a transient analysis under the OSS from Tab. 2 is performed.

For easy comparison of the results, we use another feature of PENELOPE: the provisioning of difference graphs, which can be derived by combining arbitrary experiments. In Fig. 4(right), a difference graph for states N_0, N_2, and N_3 is depicted. Each point of this graph is a result of a subtraction $a-b$, where a is the throughput computed under the OTS (Fig. 4 left), and b is the throughput computed under the OSS (Tab. 2) for corresponding service rates. As is to be seen from Fig. 4(right), no difference in performance can be observed for service rates $\mu < 0.04$, that is, if $\frac{\lambda_1+\lambda_2}{\mu} > 1$. The relative high load causes the system to reach the absorbing state quickly, regardless of the type of optimization being applied. Therefore, the mean accumulated throughput of the system in the time horizon $[0, 6000)$ is equal to the mean throughput until absorption in both cases. If the service rate becomes sufficiently large, $\mu \geq 0.04$, the difference strongly increases. In other words, the higher the service rate, the better the resulting performance becomes when adopting the OTS as opposed to the OSS.

Simulation PENELOPE also provides simulation as a method of evaluation. Under fixed deliberately eligible strategies the behavior of the system can be sim-

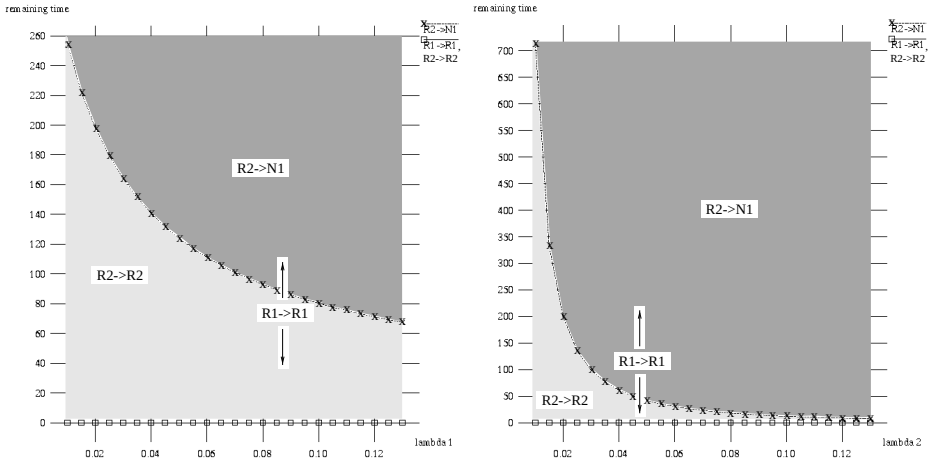
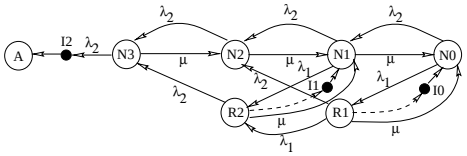


Figure 6: Dynamic control strategies as a function of time and of arrival rates.

ulated for a fixed time horizon. The simulation component enhances flexibility of the modeling tool and provides means for verification of the numerical results.

3.2.2 Minimization of Losses

The optimal strategy, which minimizes the mean number of lost class one packets, will be investigated. In order to do that, we have to modify the model from Fig. 1(right) slightly. In Fig. 7 *branching states* I_0, I_1, and I_2 were added. The new resulting reward structure is summarized in table which is attached to Fig. 7. We assign the reward (cost) rate λ_1 to states in which arriving class one packets get lost, that is, if there are at least two messages ahead, or if the absorbing state has already been reached. Furthermore, if a reconfiguration is



state	reward rate
N_i	0
N_j	λ_1
R_1	0
R_2	λ_1
I_i	1
I_2	$2p_1p_2 + p_1^2$
A	λ_1

$i \in \{0,1\}$, $j \in \{2,3\}$,
$p_1 = \frac{\lambda_1}{\lambda_1+\lambda_2}$, $p_2 = 1 - p_1 = \frac{\lambda_2}{\lambda_1+\lambda_2}$

Figure 7: The modified EMRM model.

executed from state R_i to state I_{i-1}, $i \in \{1,2\}$, one packet of the first class

is dropped. This is captured by pulse rewards 1, that are assigned to branching states I_0 and I_1. The pulse reward assigned to state I_2 corresponds to the expected number of lost class one packets, being in the system at the moment of service disruption. Note that a minimization of $E[Y_i(t)]$ can be realized in PENELOPE by specifying negative reward rates.

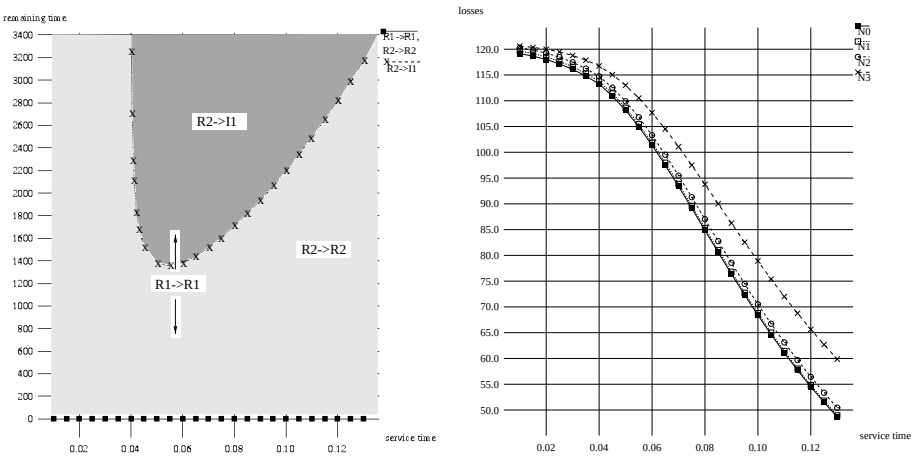


Figure 8: OTS for minimization of lost packets (left) and mean total loss under this OTS (right).

The OTS is depicted in the left part of Fig. 8. If the system is in the state R_1 then the strategy is the same as the one of the throughput model: Packets of class one are accepted for the whole time horizon and for all values of the considered service rate parameter interval. If the system is in the state R_2, however, two overlapping effects are observed.

If the service rate is relatively small, or equivalently, the load relatively high, it can be observed that the smaller the service rates the less it pays off to discard a packet. This is due to the fact that the absorbing state will be reached very quickly under that load condition. For service rates $\mu < 0.04$, packets will therefore never be discarded if the total loss is to be minimized.

A reverse effect dominates for larger values of μ: The higher the service rate, the less the expensive discarding is performed. The risk of reaching the absorbing state decreases with an increasing service rate. It becomes more and more likely that the queue will shrink again before next arrival of a class two message. The overlapping of reverse effects results in a minimum of the switching curve $R_2 \rightarrow I_1$, which partitions the decision space into regions, where alternative strategies apply, at $\mu \approx 0.055$.

The right part of Fig. 8 shows the total loss in time horizon $[0, 6000)$ as a function of the service rate for the initial states N_0, N_1, N_2, and N_3. With increasing service rate the $E[Y_i(t)]$ decreases. It is also evident, that the less packets are initially in the queue, the less packets get lost.

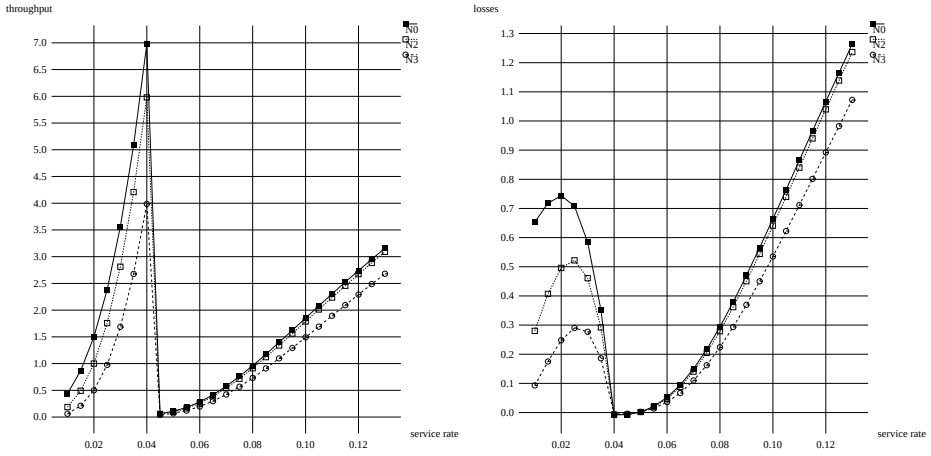


Figure 9: Difference graph of throughput and losses computed under different strategies.

It is also interesting to investigate the impact of the OTS, minimizing the total loss, on the performance in terms of mean accumulated throughput. We refer back to the parameter set summarized in Tab. 1 and select again the time horizon $[0, 6000)$ for transient analysis. With the throughput model (Fig. 1 right) and the attached reward structure from Fig. 2, a transient analysis under the strategy from Fig. 8(left), which minimizes losses, is performed. Since this strategy is not optimal for the throughput model, the resulting performance is poorer than the one in Fig. 5, gained under the optimal strategy. The difference graph is depicted in the left part of Fig. 9. Each point is a result of a subtraction $a - b$, where a is the throughput computed under the transient strategy, which is optimal for the throughput model (Fig. 4 left), and b is the throughput computed under the transient strategy, which is optimal for the loss model (Fig. 8 left).

The right part of Fig. 9 shows results from a reverse experiment. The baseline model is the model from Fig. 7, with the attached reward structure. a is the mean accumulated loss computed under the transient strategy, which is optimal for the loss model (Fig. 8 left), and b is the mean accumulated loss computed under the transient strategy optimal for the throughput model (Fig. 4 left).

4 Conclusions

We have presented a new modeling tool which can be used for the combined optimization and evaluation of reconfigurable systems. The important features of PENELOPE have been introduced by means of example. In particular, it was demonstrated how transient optimization techniques and optimization until absorption can be applied for congestion control problems by providing selective packet discarding strategies. It was shown how model structures and optimization strategies could be easily combined with each other in order to compare

the impact of different control strategies on resulting performance or performability functions. While series of experiments are flexibly accomplished with a minimum of user interference, complex control strategies can be presented in an abstract way directly relating to a series of input models. The specification of control options is based on the paradigm of reconfigurability. A knowledge of details of the underlying algorithms is therefore not necessary in order to apply optimization techniques.

For the sake of completeness we may mention that the largest optimization models which were investigated with our tool had a size in the order of some $100,000$ states.

Continuing effort is made to improve the interface further and to extend the class of implemented numerical optimization algorithms.

References

[1] Béounes, C., Aguéra, M., Arlat, J., Bachmann, S., Bourdeau, C., Doucet, J.-E., Kanoun, K., Laprie, J.-C., Metge, S., Moreira de Souza, J., Powell, D., Spiesser, P.: "Surf-2: A program for dependability evaluation of complex hardware and software systems"; *Proc. FTCS 23, IEEE Computer Soc. Press.*, 1993, pp. 668-673.

[2] Ciardo, G., Muppala, J., Trivedi, K.S.: "SPNP: Stochastic Petri net package"; *Proc. PNPM'89, IEEE Computer Soc. Press*, 1989, pp. 142-151.

[3] Couvillion, J.A., Freire, R., Johnson, R., Obal II, W.D., Qureshi, A., Rai, M., Sanders, W.H., Tvedt, J.E.: "Performability modelling with UltraSAN"; *IEEE Software*, September 1991, 69-80.

[4] Furchtgott, D.G., Meyer, J.F.: "A performability solution method for degradable nonrepairable systems"; *IEEE Trans. Comput.* 33(6), 1984, 550-554.

[5] Haverkort, B.R., Niemegeers, I.G.: "Performability Modelling Tools and Techniques"; *Performance Evaluation*, Vol. 25, No. 1, March 1996.

[6] Lee, Y.H., Shin, K.G.: "Optimal Reconfiguration Strategy for a Degradable Multimodule Computing System", *Journal of the ACM*, Vol. 34, No. 2, 1987.

[7] Lindemann, C., German, R.: "DSPNexpress: A software package for efficiently solving deterministic and stochastic Petri nets"; *Performance Tools 1992*, Edinburgh University Press, Edinburgh 1993.

[8] de Meer, H.: "Transiente Leistungsbewertung und Optimierung rekonfigurierbarer fehlertoleranter Rechensysteme"; *Arbeitsberichte des IMMD der Universität Erlangen-Nürnberg*, Vol. 25, No. 10, October 1992.

[9] de Meer, H., Trivedi, K.S., Mario Dal Cin: "Guarded Repair of Dependable Systems"; *Theoretical Computer Science, Special Issue on Dependable Parallel Computing*, Vol. 129, July 1994.

[10] de Meer, H., Ševčíková, H.: "XPenelope User Guide, Version 3.1"; Technical Report, FBI-HH-M-265/96, University of Hamburg, November 1996.

[11] Schulzrinne, H., Kurose, J.F., Towsley, D.: "Congestion Control for Real-Time Traffic in High-Speed Networks"; *IEEE INFOCOM'90*, 1990.

[12] Stewart, W.J.: *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, 1994.

[13] Tijms, Henk C.: *Stochastic modelling and analysis : a computational approach*; John Wiley, 1986.

[14] "Special Issue on Performability"; *Performance Evaluation*, February 1992.