# TimeNET Optimization Environment

## Batch simulation and heuristic optimization of SCPNs with TimeNET 4.2

Christoph Bodenstein
System & Software Engineering, Ilmenau
University of Technology, P.O. Box 100 565
D-98684 Ilmenau
Germany
Christoph.Bodenstein@tu-ilmenau.de

Armin Zimmermann
System & Software Engineering, Ilmenau
University of Technology, P.O. Box 100 565
D-98684 Ilmenau
Germany
Armin.Zimmermann@tu-ilmenau.de

## ABSTRACT
In this paper a novel tool for simulation-based optimization and design-space exploration of Stochastic Colored Petri nets (SCPN) is introduced. The working title of this tool is **T**imeNET **O**ptimization **E**nvironment (TOE).

Targeted users of this tool are people modeling complex systems with SCPNs in TimeNET who want to find parameter sets that are optimal for a certain performance measure (fitness function). It allows users to create and simulate sets of SCPNs and to run different optimization algorithms based on parameter variation.

The development of this tool was motivated by the need to automate and speed up tests of heuristic optimization algorithms to be applied for SCPN optimization. A result caching mechanism is used to avoid recalculations.

## Keywords
SCPN, Simulation based optimization, Petri nets

## 1. INTRODUCTION
Simulation-based optimization is a useful technique when the shape of the fitness function to be optimized is unknown or very complex so that gradient-based optimization algorithms cannot be applied [3, 6, 1, 7, 15]. As the modeling of real-life systems often results in complex models, simulation-based optimization has become a popular engineering design method over the last years.

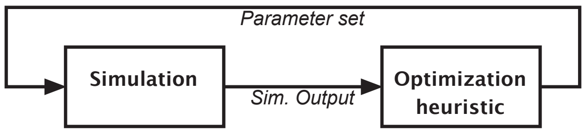


**Figure 1: Common black-box optimization, see [2]**

Carson and Maria [2] published an overview of approaches to optimize these systems. The common part of most methods as shown in Figure 1 is a so-called black box optimization.

At the beginning of an optimization, the heuristic algorithm [9] starts with the value range of parameters and their possible discretization. It generates an input (parameter set), triggers the simulation combined with input data, and uses the generated simulation results to calculate the next input until a stop condition is satisfied.

Stochastic colored Petri nets (SCPNs [12]) are a useful tool to model complex systems in numerous research areas and industrial application fields. For editing and simulating these nets we use TimeNET [13], a software tool developed originally at TU Berlin and now maintained at TU Ilmenau. Our focus for optimization are heuristic methods, a subset of simulation-based optimization methods [2].

The idea of coupling an optimization tool and a simulation tool via defined interfaces is not new, compare for instance [11] and the implementation in the REMO tool. It was written in SUN Pascal and is running on Sparcstations with OS 4.1.3 or Solaris 2.5.X. Some of the features and requirements mentioned therein apply to our use cases and are adopted, while others were not feasible or useful for our application. Restrictions in terms of further development, cross-platform use and required adaptation to TimeNET led us to the development of a new specialized tool.

TOE should help to discover the design space, esp. the fitness function shapes of parameterizable stochastic Petri nets and to find better optimization heuristics (or at least better configurations for existing ones) to be used for simulation based optimization of SCPNs.

## 2. AUTOMATED PARAMETER OPTIMIZATION OF SCPNS
TimeNET is a tool for modeling, simulation and analysis of Stochastic Colored Petri nets and several other model classes. One key feature is the parametrization of nets which means that users can easily change properties of SCPNs by defining and modifying parameters. These parameters can be used in timing functions, measurements, and as place capacities.

DOI 10.4108/icst.valuetools.2014.258193

To simulate not only one SCPN but a set while varying specific parameters can expose interesting dependencies and improve system understanding. One possibility to do several simulations is to use the TimeNET-internal scripting engine. However, this has some disadvantages regarding flexibility, debugging options and usability. Furthermore, some features are not possible to implement with this technique.

From our experiences with TimeNET and the targeted analysis of simulation-based optimization, we identified some key requirements as listed below:

- Batch simulation of SCPNs while iterating model parameters (internal);
- Batch simulation while iterating simulation parameters (external);
- Parsing of simulation results and application of optimization heuristics;
- Modular software architecture to add other heuristics and simulation types later.

Internal parameters are all system parameters which are defined inside the model. Their influence on the simulation result is defined by model architecture, formulas, or measurement expressions. On the other hand, external parameters control the way of simulating a model. These include, among others, the seed for random number generators, the maximum simulation steps to compute, maximum CPU time, and accuracy parameters such as the maximum relative error or confidence interval of measurements.

The first requirement is obvious and could have been realized with a script inside TimeNET. It allows to explore the whole expected design space step by step with a user-defined step size for discretization of continuous parameters. The result data needs to be read and plotted to get an overview of the shape of the fitness function (measurements) from the SCPN afterward.

Some changes were necessary inside TimeNET itself. As normal parameters are stored within the SCPN .xml file, external parameters have to be handed over to TimeNET in another way.

We decided to exchange external parameters via SCPN file names. This is not a perfect way for passing information between processes, but allows easy debugging in case of simulation errors. If any unexpected results occurs during a simulation, the corresponding log file and simulation file have the same name (except file extension) including all necessary simulation control parameters for TimeNET, such that the problematic behavior can be reproduced.

Since the release of the current version 4.2, TimeNET accepts and interprets several command-line parameters regarding SCPN simulation. One of these is the SCPN file name to load at startup. Other parameters let TimeNET automatically start a stationary simulation after the net has been loaded, and close the main program after simulation to facilitate scripting and embedding in tool environments.

The proposed tool extension TOE uses TimeNET just as a command-line tool which is called for every simulation run. The results are read from the created log files afterward. This is the third requirement and necessary for implementing optimization heuristics. The tool parses result log files and stores the measured data as internal objects for later use, or exports them as a summary .csv file.

Parsing the log files and storing them as internal data structures allows the integration of optimization heuristics.

Some further requirements of TOE came up during the development. One is the ability to plot the simulation results: The user can create a script for the statistics tool R [10] and call it from within TOE to create and show simple 2D or 3D scatter-plots of chosen data from any .csv file containing simulation data.

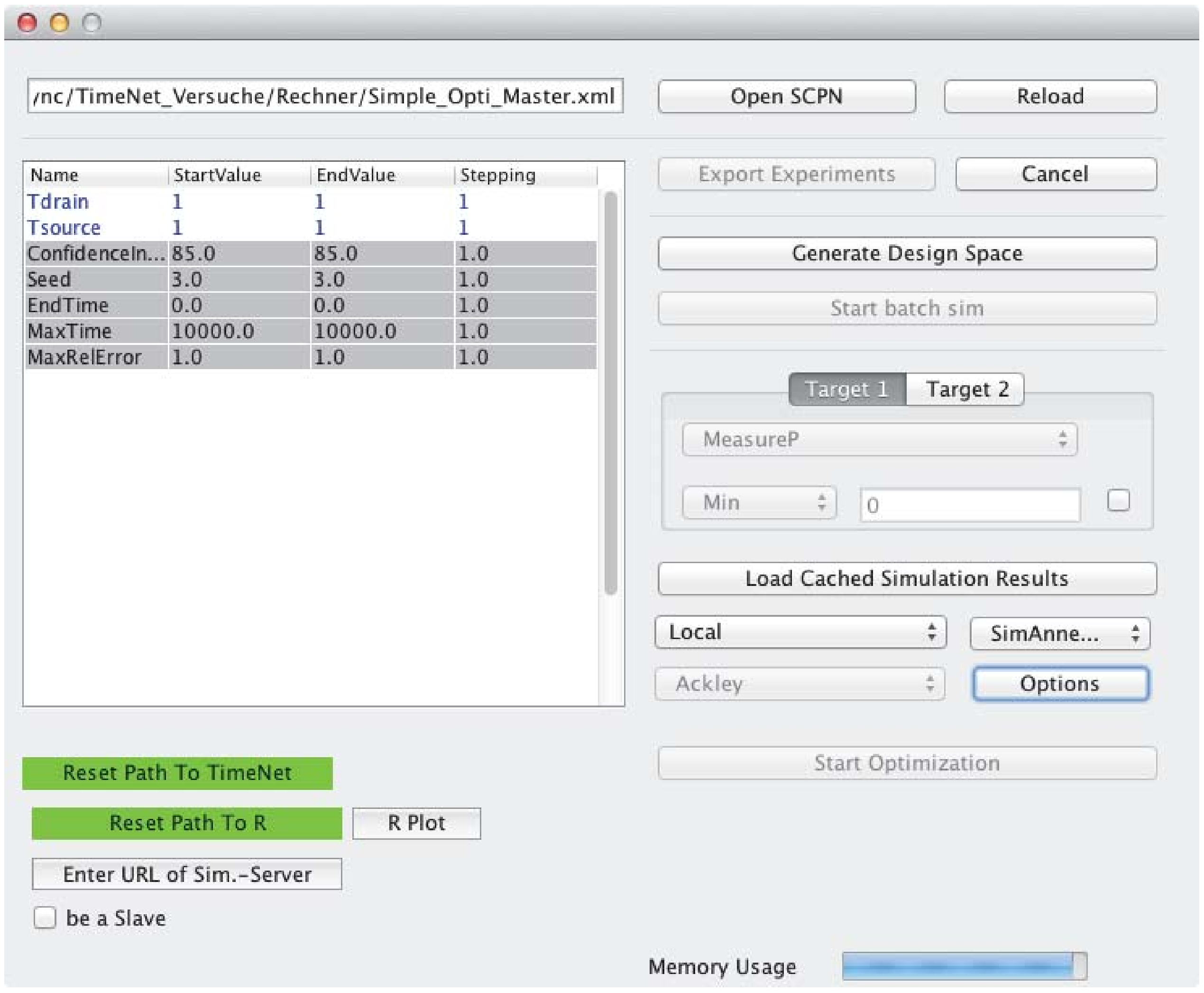
## 2.1 Current Implementation Status

By applying agile software development methodology we extended the functionality of the software over the past year. At the moment the core features are implemented and simple experiments can be run, including

- Explore the coarsely discretized design space of an SCPN;
- Visualize the design space in 2D and 3D scatter plots;
- Test several optimization heuristics online (with real simulation) and offline (with cached result data);
- Test optimization heuristics with different benchmark functions to emulate simulation runs.

To save time while testing the optimization heuristics, the user can load results from former batch simulations and use them as a result database instead of simulating the real SCPN again. For simple heuristic-tests this is very useful, but for a real optimization run this database can also be used as a cache. That means whenever the optimizer creates a new set of parameters, which has already been simulated in a former batch-run, the software will use the results from the cache. This results in a significant speedup of optimization if the used design space discretization is similar to the one used for the loaded result database. Moreover, it makes comparison between different heuristics simpler as the randomness of the underlying simulation is (at least partially) avoided. A similar caching mechanism has been implemented to be used for every single optimization run.

Figure 2 shows the main window of TOE. On the left there is a table to show all possible parameters including start and end value, which are taken automatically from the model file. In combination with the step size, this specifies size and boundaries of the design space to be explored. External parameters are marked with a gray background while internal parameters have a white one. For all parameters start/end values and discretization (stepping) can be set to determine the definition space for optimization. After every change of these table values, the approximate size of definition space is calculated.



| Name | StartValue | EndValue | Stepping |
|---|---|---|---|
| Tdrain | 1 | 1 | 1 |
| Tsource | 1 | 1 | 1 |
| ConfidenceIn... | 85.0 | 85.0 | 1.0 |
| Seed | 3.0 | 3.0 | 1.0 |
| EndTime | 0.0 | 0.0 | 1.0 |
| MaxTime | 10000.0 | 10000.0 | 1.0 |
| MaxRelError | 1.0 | 1.0 | 1.0 |

**Figure 2: Main window of TOE**

## 2.2 Restrictions and Future Work

Despite the implemented features of TOE there are several open points to be improved in the future. Optimizers and simulators have to implement a specific interface to make it easier to integrate other optimization heuristics or further simulator tools.

However, besides this interface, developers usually need to learn a lot about the internal structure before they can integrate other simulators/optimizers. A plugin-based software architecture for simulation-based tools [8] would be very useful especially for the integration of new heuristics.

The external binding to TimeNET is realized via specific file name conventions and start parameters at the moment. An integration of other Petri net types such as extended deterministic and stochastic Petri nets (eDSPNs [4]) would thus require some changes in TimeNET itself. On the other hand, the TOE parser for result log files would need to be adapted for other kinds of simulations and Petri nets.

Other optimization heuristics like adaptive optimization [14] or population-based heuristics [5] are being implemented and planned to be available within the next months.

A major feature to speed up batch simulation and population-based optimization in the future is currently being implemented and already visible in the interface: distributed simulation of SCPNs. TOE can run on several computers as a client. It will receive simulation tasks (SCPNs & simulation parameters) from a coordinating simulation server and return the results via server to one Master instance of the tool. For a rough sight of design space shapes this is a time saving feature. This will also help to improve the runtime of population-based optimization through parallel simulation of multiple parameter sets.

Besides this, some user interface improvements are planned to make the tool easier to use and to avoid unexpected behavior.

## 3. TOOL USE

Figure 3 shows the sequence of usual steps with TOE for batch simulation or optimization. We tried to improve usability and force the user to follow this workflow by deactivating all unnecessary buttons each time the software changes its major states. For instance, users cannot start an optimization while the design space is not defined or too small.

After starting the software, the last used SCPN is loaded automatically. If the program was not used before the user has to choose an SCPN file with "open SCPN". TOE will parse the corresponding .xml file and show all found parameters in the table as well as all possible measures which can be chosen as optimization targets.

Now the design space can be defined by setting the start/end value and the discretization value (step size). The program will automatically estimate the corresponding design space size for changed values in the parameter part. The creation of all possible parameter combinations for batch simulation can be started manually ("Generate Design Space").

Instead of defining the design space it is possible to load simulation results from a given .csv file ("Load cached simulation results"). In that case the values for start, stop, and step will be changed to match the loaded (and then cached) data. In addition to that, the option to simulate "cache-only" will be activated. The user can limit the design space by setting other values for that. If these values extend the range of the cached data, the cache-only simulation option will be disabled.

The user can then either start a batch simulation or optimization. For an optimization there is no need to generate the whole design space first, but the measure to be optimized and its target value have to be set. The predefined standard optimization algorithm is hill climbing; other algorithms [9] such as simulated annealing [6] or several population-based algorithms [5] are currently being implemented.

Optimization results and some other information are printed in the log-window and (if activated) saved in the program log file. To get a quick overview of simulation and optimization results, the user can open the R-plugin, choose the appropriate .csv file as well as the data-columns to plot, and start the generation and execution of an R-script to show the results as a scatter plot (Figure 4 shows an example).

### 3.1 An Application Example

One typical use case of our current work is to measure the influence of simulation precision on simulation time and used CPU time. Simulation precision of SCPNs is mostly determined by external accuracy parameters. These are maximum relative error and configured confidence interval. As an experiment, a simple Petri net is chosen and simulated approximately 400 times while varying these accuracy parameters.

Figure 4 shows the results of this experiment. With increasing confidence interval the necessary CPU time per simulation run increases as expected. However, the results also show that the maximum relative error has a much bigger influence on the CPU time. Of course these tradeoffs depend strongly on the specific SCPN model.

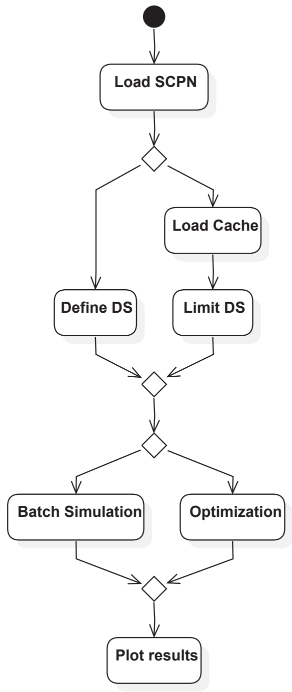


**Figure 3: Use of TOE**

These results will guide further development of the tool and lead to integration of more sophisticated optimization heuristics. Besides analyzing different optimization heuristics, the accuracy of simulations will be controlled by TOE in the future, following the ideas described in [14].

### 3.2 System Requirements

The tool is implemented completely in Java and is executable on most standard systems which support Java 1.6 or later. It has been tested successfully on Windows 7, OSX 10.9.4, and Ubuntu 12.2 so far.

As an obvious prerequisite, TimeNET 4.2[1] [13] has to be installed to handle command line parameters and execute simulation requests. TimeNET itself needs some additional libraries on Unix-like systems and a GNU C compiler (or MinGW on Windows).

To plot the generated simulation results, the R package[2] [10] needs to be installed including the 3D-plot library[3].

At first program start the user should click the buttons for "Path to R" and "Path to TimeNET" and navigate to the according directories. If the chosen directories are correct, the corresponding user interface buttons turn green and TOE is fully operational.

[1]http://www.tu-ilmenau.de/timenet
[2]http://www.r-project.org/
[3]http://cran.r-project.org/web/packages/plot3D

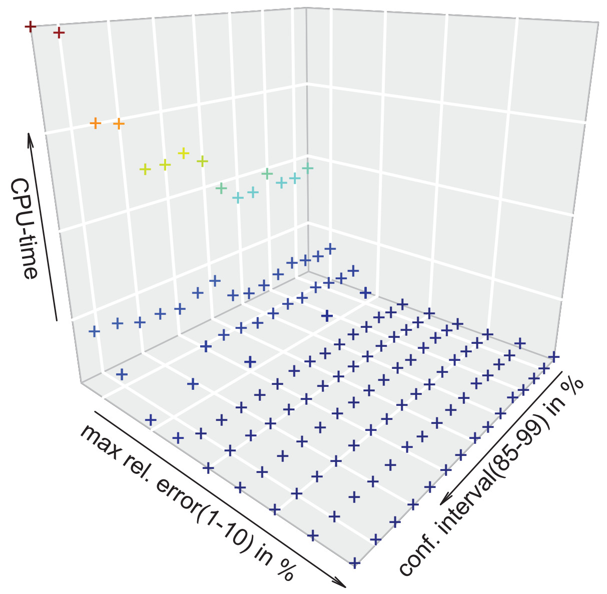


**Figure 4: Maximum relative error and confidence interval in relation to used CPU time**

## 4. CONCLUSIONS

The paper presents an extension tool for TimeNET supporting optimization heuristics and design space exploration. The tool allows to test implemented algorithms based on newly computed or cached simulation results, mixes of both, as well as benchmark function results.

The tool is written completely in Java with an extendable architecture. The current version allows basic analysis of SCPN design spaces and optimization. Other features such as distributed simulation, more heuristics and other Petri net types will be available with the next major releases. It is planned to be available via the TimeNET download website at www.tu-ilmenau.de/timenet by the end of 2014.

One of the final goals of this development is to implement accuracy-adaptive optimization heuristics.

## 5. REFERENCES

[1] J. Biel, E. Macias, and M. Perez de la Parte. Simulation-based optimization for the design of discrete event systems modeled by parametric Petri nets. In *Computer Modeling and Simulation (EMS), 2011 Fifth UKSim European Symposium on*, pages 150–155, 2011.

[2] Y. Carson and A. Maria. Simulation optimization: Methods and applications. In *Proceedings of the 29th Conference on Winter Simulation*, WSC '97, pages 118–126, 1997.

[3] M. C. Fu. A tutorial overview of optimization via discrete-event simulation. In G. Cohen and J.-P. Quadrat, editors, *11th Int. Conf. on Analysis and Optimization of Systems*, volume 199 of *Lecture Notes in Control and Information Sciences*, pages 409–418, Sophia-Antipolis, 1994. Springer-Verlag.

[4] R. German. *Performance Analysis of Communication Systems, Modeling with Non-Markovian Stochastic Petri Nets*. John Wiley and Sons, 2000.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[6] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[7] S. Künzli. *Efficient Design Space Exploration for Embedded Systems*. Phd thesis, ETH Zurich, Apr. 2006.

[8] R. Maschotta, S. Jager, T. Jungebloud, and A. Zimmermann. A framework for agile development of simulation-based system design tools. In *Systems Conference (SysCon), 2013 IEEE Int.*, pages 861–866, April 2013.

[9] C. L. Reeves. *Modern Heuristic techniques for Combinatorial Problems*. Wiley, 1993.

[10] S. Stowell. *Using R for Statistics*. Apress, 2014.

[11] M. Syrjakow, E. Syrjakow, and H. Szczerbicka. Tool support for performance modeling and optimization. *International Journal of Enterprise Information Systems*, 2005.

[12] A. Zimmermann. *Stochastic Discrete Event Systems*. Modeling, Evaluation, Applications. Springer-Verlag New York Incorporated, Nov. 2007.

[13] A. Zimmermann. Modeling and evaluation of stochastic Petri nets with TimeNET 4.1. In *Performance Evaluation Methodologies and Tools (VALUETOOLS), 2012 6th Int. Conf. on*, pages 54–63, Oct 2012.

[14] A. Zimmermann and C. Bodenstein. Towards accuracy-adaptive simulation for efficient design-space optimization. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 1230 –1237, Oct. 2011.

[15] A. Zimmermann, D. Rodriguez, and M. Silva. A two-phase optimisation method for Petri net models of manufacturing systems. *Journal of Intelligent Manufacturing*, 12(5/6):409–420, Oct. 2001. Special issue "Global Optimization Meta-Heuristics for Industrial Systems Design and Management".