Optimizing OMNeT++ Network Simulations with the Open-Source Optimization Framework Simopticon

Burkhard Hensel, Touhid Hossain Pritom, Christoph Sommer
TU Dresden, Faculty of Computer Science, Germany
{ burkhard.hensel, touhid_hossain.pritom, christoph.sommer }@tu-dresden.de

***Abstract*—Network simulations play an important role in many domains. Often it is helpful to optimize simulation parameters automatically based on given simulation output metrics. We present an extension of the open-source optimization framework Simopticon that is able to automatically optimize simulation parameters of OMNeT++ simulations. To demonstrate the optimizer, we apply this to two simulation models: A Veins scenario where 15 wirelessly connected vehicles optimize their routes, and a Simu5G scenario optimizing the gNodeB's transmission power in a mobile network.**

***Index Terms*—Wireless networks, simulation, optimization**

I. Introduction

Simulations are a cornerstone of the development of new products and technologies, particularly in networking. Network simulations typically precede real network implementations, but can also be used to represent existing networks, especially in the context of a network digital twin [1, 2].

Whenever network simulations should match a real network but not all parameters of the real network are known, it is necessary to try and adjust the unknown model parameters so that the simulation outputs match real measurements. This is an optimization task. Similarly, if the goal is not to match a real network but to find the best parameter settings for a given purpose, this too is an optimization task.

OMNeT++ is one of the most well-known network simulation frameworks. It provides basic means for parameter studies by allowing to define ranges of values for parameters, but automatic optimization of model parameters based on simulation outputs is not supported directly.

In this paper, we describe how we filled this gap by extending the open-source software Simopticon for optimizing parameters of OMNeT++ network simulations. We also describe two use cases as demonstration examples, one using Veins to simulate cooperative driving and one using Simu5G to simulate a mobile network.

II. Related Work

Optimization of simulation parameters is a common task, but it is often not the focus of the respective publications or still performed manually. Dräxler et al. [3] published a simulation model designed for energy optimization, but the optimization itself is not part of the paper. Likewise, Davoudzade et al. [4] defined all relevant parts of a network optimization (inputs, outputs, parameters, objective function), but ran all parameter combinations manually instead of using an optimizer.

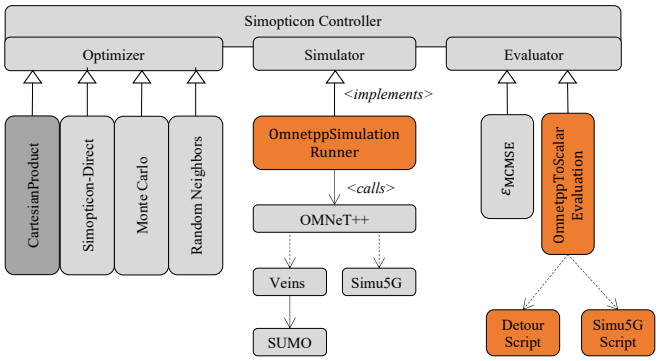


Figure 1. Architecture of Simopticon with extensions for running OMNeT++ simulations and the demonstration scenarios highlighted in orange.

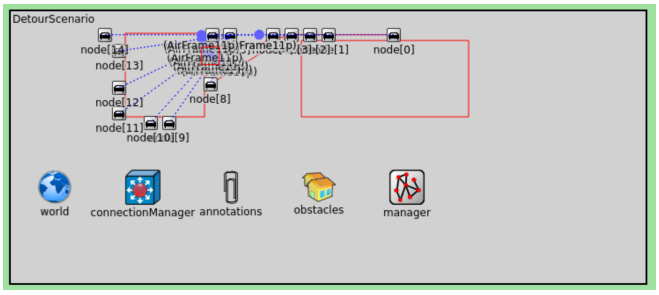
Where optimization is the focus of the publication, it is often performed with commercial software, hindering reproducibility and reusability of the results. Virdis [5] integrated the commercial optimization solver CPLEX into the OMNeT++ runtime to optimize simulations during execution. Chang et al. [6] developed an optimizer for network reliability based on MATLAB. Similarly, Simon et al. [7] presented a MATLAB-based optimization software for network simulation. Both require MATLAB as a platform, which is not available as open-source software.

III. Simopticon Extensions

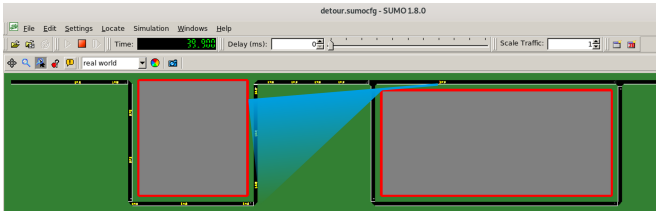
Simopticon [8] is a generic, modular optimization software and is available as open-source software.[1] Although it is designed in flexible modules, it was originally created solely to optimize controller parameters in Plexe, an extension of Veins for simulating platoons of road vehicles, a framework which in turn builds on OMNeT++. Although Plexe uses OMNeT++ for network simulation, some parts of the original Simopticon were so specific that it was not directly possible to optimize just OMNeT++ simulations without Plexe.

To support also other OMNeT++-based simulations, a new *OmnetppSimulationRunner* has been implemented that can be used instead of the *PlexeSimulationRunner* to configure and run OMNeT++ simulations. It also links the necessary libraries and NED directories. The mobile communication library Simu5G [9] and the simulation framework Veins [10] can be loaded simply by setting a boolean parameter and their

[1] https://nsm.inf.tu-dresden.de/research/software/simopticon/



(a) Detour example with Veins, screenshot in OMNeT++.



(b) Detour example with shadowing by buildings, shown in SUMO.

Figure 2. Detour example.

installation path. In addition, a new *OmnetppToScalarEvaluation* module calls a Python script to evaluate simulation results to obtain a metric that is used as the objective function in the optimization. Several Python scripts can coexist, because for each simulation model one might need to adapt the script based on the scalar or vector data to be evaluated. The changes are visualized in Fig. 1.

IV. Demonstration Scenario 1: Detour Scenario with Veins

In this demonstration example, the optimization software is applied to a simulation example that utilizes the Veins framework. Veins relies on OMNeT++ for network simulation and SUMO for road traffic simulation to enable the simulation of vehicles as network nodes, both in terms of their mobility and their communication capabilities. In this case study, we use OMNeT++ 6.3.0, Veins 5.3.1, and SUMO 1.18.0.

For the demonstration, we selected a "detour" scenario. In this scenario, a total of 15 vehicles follow a road winding around buildings with both a shorter route and a longer detour available to reach their goal. Right after a fork in the road, we force the first vehicle to stop, simulating, e.g., an accident (see Fig. 2a and Fig. 2b). The remaining vehicles can take a detour to reach their goal, but will do so only if they receive a wireless broadcast from the first vehicle containing the accident report; otherwise, they will end up in a jam behind the accident. Wireless communications, though, are hindered by buildings.

As a simple demonstration protocol, the original broadcast is thus repeated 4 times with a time period t between two broadcasts; vehicles that received the broadcast will forward it immediately with probability p. All vehicles use the same transmission power P.

In this case study, we optimized these parameters using Simopticon, restricting them to intervals of $t \in [0, 30\,\mathrm{s}]$, $p \in [0, 1]$, and $P \in [0, 30\,\mathrm{mW}]$.

The metric for evaluating the quality of a parameter combination J is a weighted sum of: the number of informed vehicles (which got the accident warning) N_{received}, the number of wireless broadcasts that were sent N_{sent}, and the transmission power P. It is defined as

$$J = J_{\mathrm{basis}} + \alpha \cdot N_{\mathrm{sent}} - \beta \cdot N_{\mathrm{received}} + \gamma \cdot P. \tag{1}$$

J_{basis} is an offset to make the metric positive. This is not necessary for the optimizer and has no influence on the optimization result, but leads to more intuitive values. Its value has been set to 10 000. We chose $\alpha = 1$, $\beta = 3$, and $\gamma = 1/\mathrm{mW}$ as example weighting factors, meaning, e.g., that an additional sent broadcast is evaluated as beneficial if more than three additional vehicles receive it.

We simulated 10 repetitions per parameter combination and took the mean of these solutions. Based on the number of vehicles, repetitions, and parameters of J, the theoretical minimum of J is 9590. It would be reached if, in all 10 repetitions, all 15 vehicles received the broadcast without any forwarded broadcasts and with minimal transmission power.

We optimized the parameters using Simopticon with the three currently available optimizers of Simopticon: the *Direct*, *Monte Carlo*, and *Random Neighbors* optimizers. We stopped each optimizer after testing 500 parameter combination. The progress of the optimization and a scatter plot of the simulation results are shown in Fig. 3. A Python script for generating such diagrams has been added to Simopticon. The different characteristics are clearly visible from the scatter plots: While the *Direct* optimizer searches systematically by dividing the search space into rectangles of decreasing size and the *Monte Carlo* optimizer selects random parameter combinations within the whole parameter space, the *Random Neighbors* optimizer extends the random approach by preferably searching in the neighborhood of the currently best found solution.

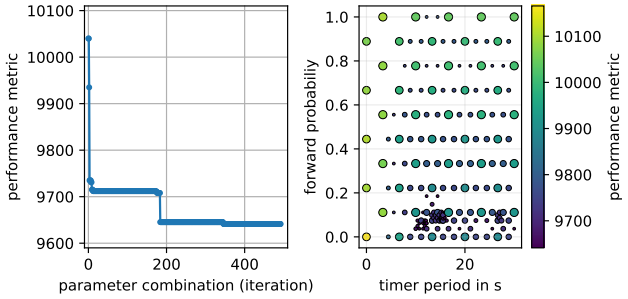
The best solution within this study has been found with the *Random Neighbors* optimizer, lying at $t = 24.7659\,\mathrm{s}$, $p = 0.137196$, and $P = 0.325\,909\,\mathrm{mW}$ with $J = 9609.26$.

V. Demo Scenario 2: Simu5G

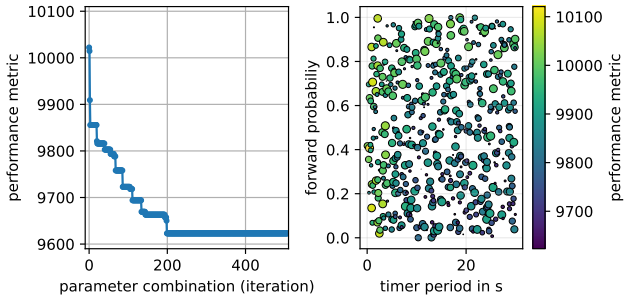
The second demonstration use case leverages Simu5G to simulate a mobile network. We use OMNeT++ 6.3.0, Simu5G 1.4.3, and INET 4.6.0. The "Multiple-UEs" configuration of the "tutorial" scenario provided with Simu5G simulates several UEs (user equipments) which communicate with a server via a gNodeB (base station) and a few intermediate nodes. Transmission occurs only from the gNodeB to the UEs, not vice versa. Several parameters can be adjusted in the simulation model, we search for the best transmission power of the gNodeB, called *eNodeBTxPower* in the simulation model, to optimize the throughput, called *cbrReceivedThroughput*.

We define the optimized metric as

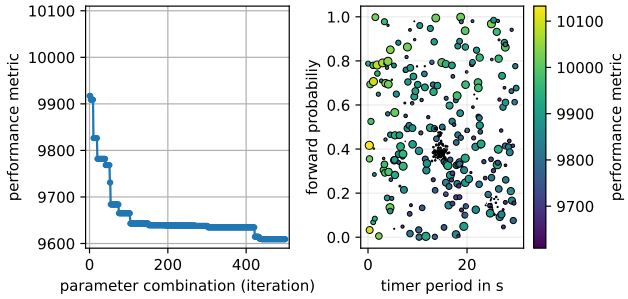
$$J = J_{\mathrm{basis}} - T + \delta \cdot P \tag{2}$$



(a) The *Direct* optimization algorithm, systematically dividing the search space into rectangles of decreasing size.



(b) The *Monte Carlo* optimization algorithm, always selecting random parameter combinations within the search space.



(c) The *Random Neighbors* optimization algorithm, searching with a probability of 50% in a neighborhood of $\pm 10\%$ around the current optimum, and with a probability of 50% in the entire search space.

Figure 3. Progress of the performance metric J in the detour demonstration example and scatter plot of the simulation results. The third parameter (transmission power P) is represented by the marker size – the higher the transmission power, the bigger the marker.

with an arbitrary offset $J_{\text{basis}} = 100000$, the transmission power P (in dBm), the throughput T (in kbps), and a weighting factor $\delta = 1$. The term $\delta \cdot P$ is very small compared to the throughput values and mainly added to select the smallest transmission power if several allow the same (maximum) throughput. Each parameter combination has been repeated 3 times and the mean of the results has been used for each parameter combination.

Selecting the the *Monte Carlo* optimizer of Simopticon and testing 500 parameter values in the interval $[-40\,\text{dBm}, 40\,\text{dBm}]$, the optimum is found at $P = 18.6727\,\text{dBm}$ resulting in $J = -55.093$. The convergence of the performance and the simulation results are presented in Fig. 4, demonstrating the capabilities of Simopticon for optimizing OMNeT++ simulations.

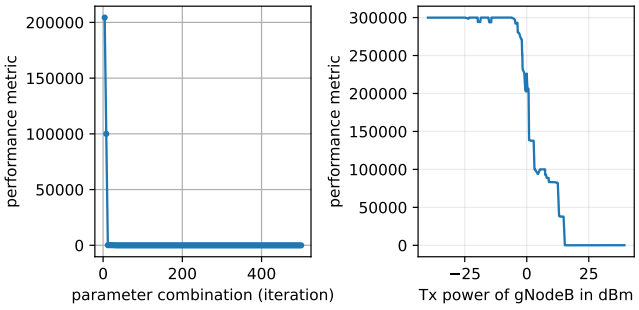


Figure 4. Progress of the performance metric J in the Simu5G demonstration example and plot of the simulation results.

VI. Conclusions

In this paper, we present an extension of the open-source simulation framework Simopticon, designed to optimize parameters of OMNeT++ simulation models. Future work will focus on broadening the range of supported simulation software and enhancing the suite of tools for simulation model analysis and optimization.

Acknowledgments

The 6G-TWIN project has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation program under Grant Agreement No 101136314.

References

[1] A. Zaki-Hindi et al., "A Reference Functional Architecture for Network Digital Twins in 6G Systems," *IEEE Open Journal of the Communications Society*, vol. 7, pp. 2068–2101, Feb. 2026.

[2] S. Si-Mohammed, A. Bardou, T. Begin, I. Guérin Lassous, and P. Vicat-Blanc, "NS+NDT: Smart integration of Network Simulation in Network Digital Twin, application to IoT networks," *Elsevier Future Generation Computer Systems*, vol. 157, pp. 124–144, Aug. 2024.

[3] M. Dräxler, F. Beister, S. Kruska, J. Aelken, and H. Karl, "Using OMNeT++ for Energy Optimization Simulations in Mobile Core Networks," in *5th International Conference on Simulation Tools and Techniques (SIMUTOOLS'12)*, ACM, 2012, pp. 157–165.

[4] M. Davoudzade, C. Barber, B. Esfandiari, and T. Kunz, "Enhancing Network Simulation Accuracy through Calibration and Model Optimization," in *17th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling (PoEM 2024)*, Stockholm, Sweden: CEUR-WS, 2024.

[5] A. Virdis, *Optimization in the Loop: Implementing and Testing Scheduling Algorithms with SimuLTE*, 2015. arXiv: 1509.03562 [cs.NI].

[6] P.-C. Chang, C.-T. Yeh, V. F. Yu, and S.-F. Chiu, "Maximizing network reliability through simulation optimization: a component configuration strategy," *Springer Annals of Operations Research*, Apr. 2025.

[7] G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi, "Simulation-based optimization of communication protocols for large-scale wireless sensor networks," in *2003 IEEE Aerospace Conference Proceedings (Cat. No. 03TH8652)*, vol. 3, IEEE, 2003, 3_1339–3_1346.

[8] P. Natzschka, B. Hensel, and C. Sommer, "Simopticon: Automated Optimization of Vehicular Platooning Controllers," *Elsevier Ad Hoc Networks*, vol. 170, p. 103 781, Apr. 2025.

[9] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5G – An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks," *IEEE Access*, vol. 8, pp. 181 176–181 191, 2020.

[10] C. Sommer et al., "Veins – the open source vehicular network simulation framework," in *Recent Advances in Network Simulation*, A. Virdis and M. Kirsche, Eds., 1st ed., Springer, 2019.