OPEDo: A Tool for the Optimization of Performance and Dependability Models

Markus Arns Peter Buchholz Dennis Müller
Informatik IV, TU Dortmund, D-44221 Dortmund, Germany
opedo@ls4.cs.uni-dortmund.de
http://ls4-www.cs.uni-dortmund.de/Opedo

ABSTRACT

OPEDo is a software tool for the optimization of discrete event systems according to performance or dependability measures. The tool can be seen as an add on to various tools for performance and dependability analysis. The goal of OPEDo is to provide a wide variety of optimization algorithms for complex black box functions as they are required for the model based optimization of discrete event systems using analytically tractable models or simulation models.

The paper introduces the software architecture of the tool, gives a brief sketch of the integrated optimization algorithms and presents several examples.

1. INTRODUCTION

Discrete event systems (DEDSs) can be found in various application areas ranging from computer and communication systems to manufacturing or logistics systems. The analysis of DEDSs is usually model based and different types of models are available to describe and analyze the quantitative behavior of DEDSs [8]. One can roughly distinguish between analytically or numerically tractable models and simulation models. Usually, models are applied to describe and analyze the behavior of a DEDS but the final goal of modeling is often to find an optimal or at least good configuration of the DEDS by optimizing its structure and parameters. The latter requires the availability of adequate algorithms to find parameter settings resulting in an optimal model configuration which is in general an optimization problem where the goal function is given by the model of the DEDS. OPEDo, the Optimization and Performance Evaluation tool from the TU Dortmund, is a software tool which includes a wide variety of optimization algorithms that can be applied to models of DEDSs. The idea behind OPEDo is to provide a set of optimization algorithms and define an interface that allows one to use models from different modeling tools as goal functions of optimization problems. The goal functions are evaluated during the optimization process by the solvers of the used modeling tools. Since the internal structure of the models is not visible to the optimization algorithms, models have to be taken as black boxes.

Although model based optimization is an important add on to performance and dependability modeling, optimization techniques are usually not part of performance modeling tools and even the number of theoretical and methodological papers in the area is limited to very specific problems. There is a long history in control theory and its application to queuing systems [4]. However, these techniques are restricted to specific types of optimization problems which in particular do not contain discrete optimization problems that naturally arise from optimizing resources like the number of processors or buffer spaces. An active area of research is the optimization of discrete event simulation models [10] which is still seen as a hard problem although recently some progress has been made in developing efficient optimization methods for stochastic simulation problems [11] and also for real systems using measurements [19]. Additionally, some work has been done in optimizing product form models (e.g., [7]) which can be solved analytically such that algorithms from nonlinear optimization may be applied. Some commercial optimization tools are available which are usually add ons to simulation environments [17]. The most widespread tool in this area seems to be OptQest [16] which can be combined with several simulation tools and includes an efficient optimization algorithm. However, all these optimization tools usually include one or very few heuristic optimization algorithms which are hidden from the user. Although the algorithms often work surprisingly good for most examples, one can usually easily find models where the optimization fails. Thus, from a practical perspective it seems to be important to have several optimization algorithms in a tool and to combine the optimization algorithms with different model types to support the whole design process starting with abstract queuing models and possibly ending with a very detailed simulation model. Furthermore, a tool should be open and easily extendable to new model types and new optimization algorithms. The mentioned points motivated the development of OPEDo.

From a mathematical point of view the goal of OPEDo is to find $\min_{\mathbf{x}\in\mathcal{U}}(f(\mathbf{x}))$ where $\mathcal{U} \subseteq \mathbb{R}^n \times \mathbb{N}^m$. Thus, we have n real and m integer parameters. Set $\mathcal{U}$ is defined by intervals $x_i \in [l_i, u_i]$ with $l_i, u_i \in \mathbb{R}$ and possibly q additional linear constraints $\mathbf{Ax} \leq \mathbf{b}$ where $\mathbf{A} \in \mathbb{R}^{q\times(n+m)}$ and $\mathbf{b} \in \mathbb{R}^q$. Parameters are denoted as factors in optimization. For the optimization of system performance or dependability $f(\mathbf{x})$ is given by the model of a DEDS defined in some of the modeling tools supported by OPEDo. We denote the result $f(\mathbf{x})$ as the *response* at $\mathbf{x}$ which could be a random variable, e.g., if $f(\mathbf{x})$ is realized by a stochastic model analyzed via simulation.

The current paper is structured as follows. In the next section we present the structure and architecture of the tool. Afterwards, section 3 gives a very brief overview of the integrated optimization algorithms. Then we present three example models. The paper ends with the conclusions and an outlook on future extensions of the tool.

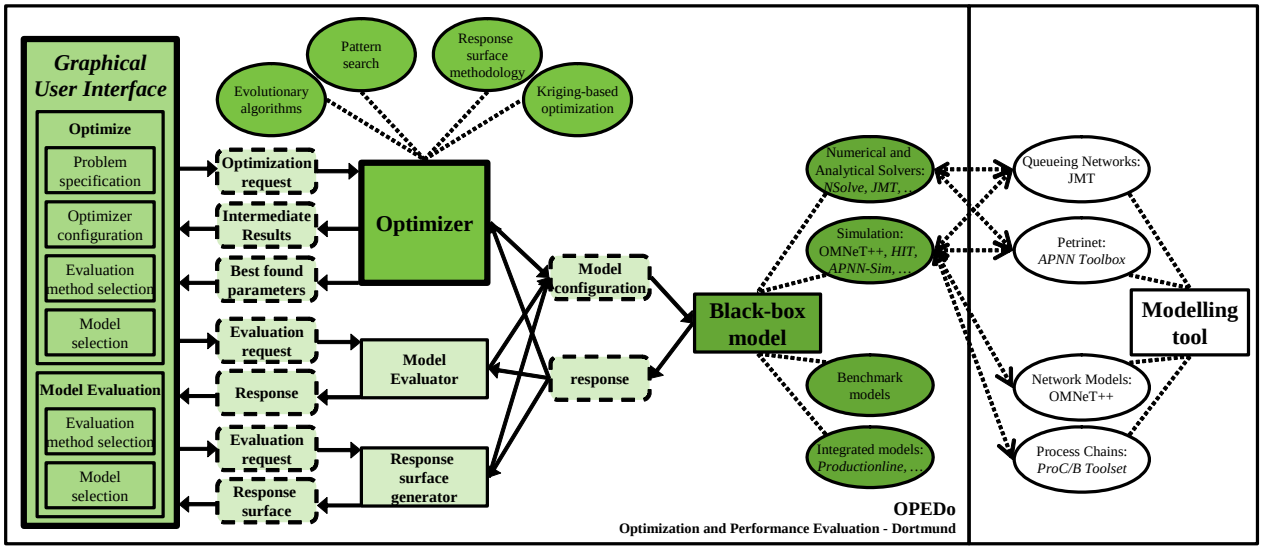

Figure 1: Overview of OPEDo

2. TOOL ARCHITECTURE

The overall architecture of the tool and connected tools are presented in Figure 1. OPEDo consists of three main parts. The corresponding modules can be identified as the darker green boxes of the figure where each part has its own shade of green. The modules communicate via files, pipes or other interfaces presented as dashed boxes. The left column contains the graphical user interface (GUI) which allows the user to manage the optimization process and which also presents the intermediate and final results. The right column describes the black box model. In black box evaluation the evaluator is only aware of an interface to define parameters and to read result values. In case of a stochastic models, additionally the seed value of the random number generator has to be set to run several replications at one point.

In the middle we have modules to manage the evaluation of the black box model. The major goal is optimization which is realized by the optimizer including several optimization algorithms that are presented in section 3. OPEDo allows optimization according to a single result measure. Factors are real values and integers, qualitative factors like scheduling strategies may be coded as integers which, however, might result in a bad performance of the optimization algorithm. For each factor an interval has to be defined, linear constraints may be defined as well. The optimization algorithm tries to find an optimal value with all parameters in the predefined intervals and all constraints satisfied. Since black box optimization algorithms compute results by an intelligent search strategy, all algorithms compute intermediate results which may be presented in the GUI.

Apart from optimization, OPEDo allows the evaluation of user defined configurations and the generation of response surfaces. If the user has some information about the possible location of the optimum, he or she may evaluate the model at this point and compare the response with other responses found by one of the optimization algorithms. Additionally, good responses can be used as initial points for one of the optimization algorithms. Furthermore the response surface can be computed as a multi dimensional grid. The user defines for each parameters an interval and a step length and the response surface generator evaluates the model at every point.

OPEDo uses functions from the GNU Scientific Library [13] for statistical evaluations and random number generation. Additional libraries are used for GUI and graphics generation. A complete list can be found on the OPEDo homepage.

2.1 Black Box Models

The goal of OPEDo is twofold. First, it is an optimization environment for performance and dependability models. Second, it is an environment for new algorithms for the optimization of black box functions. Since optimization of black box functions is mainly done by heuristic methods, a lot of experimental evaluation is necessary to find efficient and reliable algorithms. In particular, new algorithms have to be compared with available methods. It is impractical to do such comparisons by means of complex simulation models which require a huge evaluation effort. Therefore, benchmark functions have been defined in the optimization literature [9]. These functions, which have been used in many studies on the comparison of optimization algorithms, are also built in OPEDo. Since a major goal of OPEDo is the optimization of stochastic functions and the benchmark functions are deterministic, it is possible to add a $N(0, \sigma)$ distributed random variable to the function value such that the resulting optimization problem becomes a stochastic problem. Apart from the benchmark functions some basic models of DEDSs are also included, namely a (s, S) inventory system [17] and a simple tandem queuing network with Poisson arrivals and exponential service times.

To use models from some modeling tool in OPEDo, one has to define the interface between the optimizer and the modeling tool. Different levels of integration are supported. The minimal requirements to integrate a tool are the possibility to define factor values and necessary parameters for

the solver and to start an analysis run and to read the result. A loose connection between OPEDo and a modeling tool is realized by a Perl script. The parameters are identified by name and are substituted by the Perl script in the model file. Of course, this implies that the model contains parameters with the right name and of the right type. In a similar way, the seed of the random number generator is set and the result is identified by name in the output file of the modeling tool. Basic Perl scripts are available for the connection to the tools HIT [3] and OMNeT++ [12]. It is relatively straightforward to integrate other tools in a loose form by modifying the available generic Perl scripts. However, for a loosely coupled modeling tool the user has to build his or her model carefully since OPEDo cannot interpret the model structure.

The optimization process is much more convenient for models defined in tightly coupled tools. In this case, OPEDo parses the model file and presents the model parameters and possible values to the user. From the GUI the user can choose the model parameters that are used as factors in the optimization and he or she can also define the result measure to be optimized. The result measure is defined using an arithmetic expression over the results of the model. Currently, three modeling tools are tightly coupled to OPEDo, namely the APNN toolbox [2], the ProC/B toolset [1] and JMVA from the Java Modeling Tools (see M. Bertoli, G. Casale and G. Serazzi, this issue).

The APNN toolbox is a Petri net tool which allows the definition of colored generalized stochastic Petri nets. The Models can be analyzed via discrete event simulation and exact or approximate numerical analysis of the underlying Markov chain if the state space is finite and not too large. Possible parameters for the optimization are initial markings of places and firing rates or weights of transitions. The response is defined by a combination of the token distribution at the places and the firing frequency of transitions.

ProC/B is a toolset to describe hierarchical process chain models as they are particularly applied in logistics. Models are usually analyzed via simulation by mapping the process chain model onto a simulation model in one of the tools OMNeT++ or HIT, analyzing the simulation model and mapping the results to ProC/B. Factors for optimization are selected from the parameters of the ProC/B model that specify the number of resources, the speed of resources and branching probabilities. ProC/B allows the definition of very sophisticated measurement streams which can be accessed from OPEDo such that the mean values of those streams can be used to define the response.

JMVA is an analysis tool for simple multi-class queuing networks which are solved with mean value analysis. The parameters are the service rates and the number of servers. For definition of the response the standard result measures of JMVA, namely throughputs, mean populations or mean sojourn times at class or station level can be combined in an arithmetic expression.

2.2 Graphical User Interface

Optimization runs are defined in OPEDo using four steps in the GUI. The interface is shown in Figure 2. The left column contains the buttons to select different steps to set up optimization runs. According to the selection in the left column, the menu in the right column is selected. In a first step the number and types of parameters and the linear

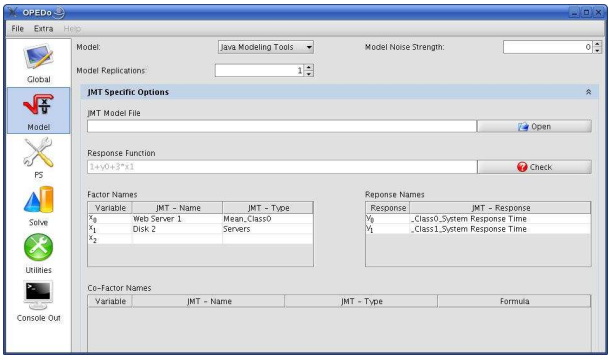

Figure 2: Model parameter specification

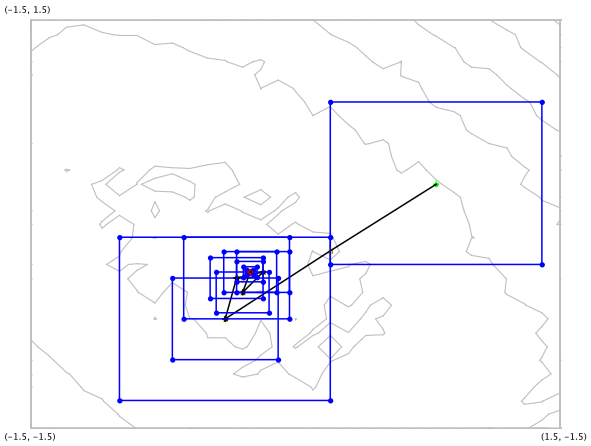

Figure 3: Search path of an optimization algorithm

constraints are defined. Figure 2 shows the screen for the parameter definition of a JMVA based optimization. The user has to relate the factors for optimization to parameters of the JMVA model and the response has to be defined in terms of the result values of the JMVA model. Furthermore, the number of replications may be defined. For an analytical solver like JMVA the number of replications is set to 1. For simulation models the number of replications should be set to some value > 1. Optimization methods which treat stochastic models like deterministic models perform for each point the selected number of replications and use the mean value of the response in the optimization algorithm. Optimization algorithms that are specifically developed for stochastic models determine the number of replications dynamically during optimization such that the fixed value is not used (see section 3).

In the third step the parameters of the optimization algorithm are selected and in the fourth step the optimization is started. Since OPEDo is intended to compare different optimization algorithms, it is possible to replicate the optimization process several times with different seeds for the models and for the optimization algorithm, if random variables are part of the model and algorithm, respectively.

The fifth selection *Utilities* allows one to start the response surface generator or to evaluate the model at a user defined point. At the last point *Console out*, the output of the op-

timization run can be observed. The value of the currently best solution is presented and for problems with only two factors, additionally, the path of the algorithm in the search space is plotted. Figure 3 shows an example using RSM but similar plots are presented for all implemented algorithms. The green dot is the starting point, the red dot the final result and black dots are intermediate results of the algorithm. The blue boxes mark the local regions during each step of the algorithm.

3. OPTIMIZATION ALGORITHMS

Since OPEDo does not have any knowledge about the goal function $f(\mathbf{x})$, the optimization methods are restricted to black box optimization methods which often means some sort of intelligent search heuristics. Search heuristics can be subdivided into global and local techniques that in general differ with respect to the quality of solutions and computation time. While global optimization techniques are supposed to find global optima they often need a large number of model evaluations. On the other hand, local techniques require less evaluations but tend to stop in local optima. The characteristics of global (resp. local techniques) make them especially applicable to multi-modal (resp. to uni-modal) objective functions. Of course, also this characterization is in case of black-box models not obvious in advance.

OPEDo contains as local search algorithms the well known Nelder-Mead Simplex algorithm (NM) [18], Pattern Search (PS) [21] and the Response Surface Method (RSM) [15]. They have been implemented such that they can deal with stochastic models. As global optimization methods, a random search method, Kriging-models (KM) [15], and Evolutionary strategies (ES) [20] have been integrated.

Although the integrated algorithms are rather general, they usually have been developed for problems where the evaluation of $f(\mathbf{x})$ is cheap or for problems where $f(\mathbf{x})$ is determined by physical experiments which implies that it is much more costly to perform an experiment with new factor values than to repeat the experiment at the same point. The situation is somehow different when DEDS are analyzed. At least for simulation or for the numerical analysis of Markov chains, the effort to evaluate $f(\mathbf{x})$ is high but it does not matter whether an experiment is made at the same or at another point.

It turns out that the basic algorithms are often not efficient enough for the optimization of DEDSs with many factors or a complex response surface. Therefore, one of our goals is to improve optimization algorithms for the analysis of DEDSs. For the lack of space we can only highlight some of the ideas resulting in improved algorithms.

The first idea is to combine a global and a local search heuristic. In OPEDo we use a combination of KM and PS which is denoted as KMPS. The hybrid KMPS basically works like the standard KM (see [15]) but dynamically adds a PS step to find local and global optima. The introduction of PS steps usually reduces the number of model evaluations such that the approach becomes useful whenever model evaluation is expensive which is the case for simulation or numerical analysis of Markov chains. A second approach defines some new version of Kriging models which are denoted as hierarchical Kriging (HKM). The detailed description of the corresponding algorithm is currently underway.

For Markov models that are analyzed numerically, a new version of RSM has been proposed in [14]. The idea of the approach is to use imprecise results at the beginning of the optimization process and to start the solution process at one point with the solution of some point in the neighborhood, if available. It has been shown that with these improvements the effort can be significantly reduced and similar optima are computed compared with the standard RSM approach.

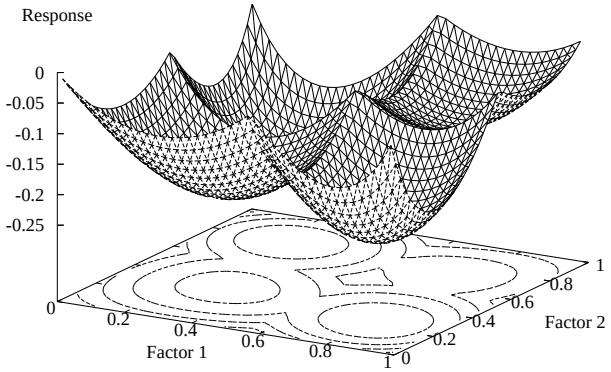

Figure 4: Response surface of the JMVA-Model

Often several alternatives exist to analyze DEDSs. Approximation methods yield approximate results in an efficient way, whereas exact methods require much more effort but compute the precise result. Sometimes approximate and exact analysis are performed on the same model, sometimes two different models have to be defined, an abstract one for approximate analysis and a detailed one for exact analysis. If approximate and exact analysis can be used, this can be exploited in optimization algorithms. In [5] a variant of ES is described that uses both solution methods and tries to reduce the number of exact evaluations as much as possible. The quality of the approximate solution algorithm is evaluated during the optimization process and depending on the quality, it is determined whether a point is evaluated exactly or approximately. Several examples show that the use of two methods reduces the solution time drastically and results in similar optima.

A last observation which can be exploited to improve optimization algorithms is that the computation of precise results via simulation is very costly since the reduction of confidence intervals by a factor of 2 requires roughly 4 times more replications [17]. Thus, it should be avoided to compute precise results where it is not necessary. In [6] a variant of ES is developed which uses a two phase selection process and statistical ranking and selection procedures to find optimal solutions. Ranking and selection assures that enough but not too many replications are performed to select the best solutions from a set of solutions according to a given significance probability. The two phase approach avoids the decision between points with a similar response at the beginning since later much better configurations may be found.

4. EXAMPLES

We briefly present three small examples using OPEDo in combination with the modeling tools JMVA, APNN toolbox and ProC/B toolset. The first example is a simple tandem queuing network with two types/classes of customers. The service demands of class i customers at station k are exponentially distributed with mean $m_{i,k}$ as $m_{i,1} = (x_i)^{-1}$ and

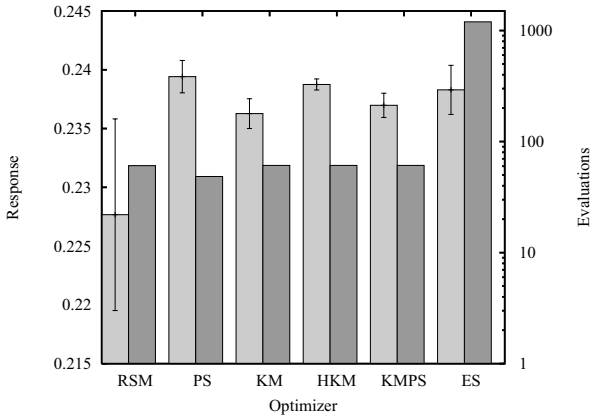

Figure 5: Results for JMVA

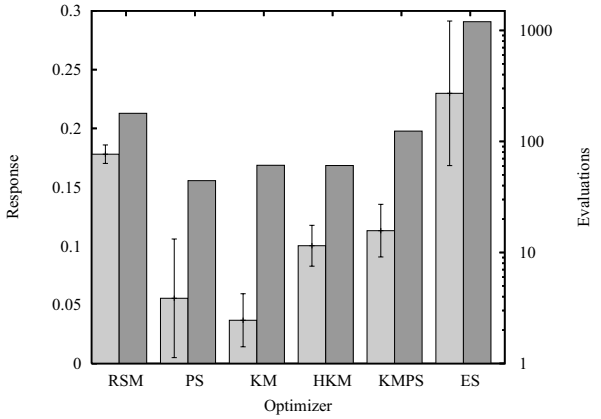

Figure 6: Results for APNN

$m_{i,2} = (1-x_i)^{-1}$ with $i = 1, 2$. The subject of the optimization problem is to determine parameters $x_1, x_2 \in [0.01, 0.99]$ such that they maximize the following objective function

$$Z_1(x_1, x_2) = (2t_1(x_1, x_2) + 2.2t_2(x_1, x_2)) \\ -\tfrac{1}{2}\left(min(x_1, 1.3(1-x_1)) + min(x_2, 1.3(1-x_2))\right)$$

where t_i is the throughput of class i customers and depends on x_1 and x_2. The model can be analyzed efficiently with JMVA. Since the evaluation is fast, a detailed response surface can be computed which is shown in Figure 4. Figure 5 shows the results of six different optimization algorithms applied to the model. Observes that the number of model evaluations which determine the runtime of the algorithms is presented on a logarithmic scale. The light gray bars indicate the optimal solution with a 90% confidence interval according to 20 replications of the optimization runs and the dark gray bars describe the mean number of model evaluations to find the optima. We configured RSM, PS and KM to require about the same number of evaluations. Due to the nature of ES more evaluations are allowed for this algorithm. The results show that RSM should not be used for multi modal functions. PS which is also a local search technique behaves much better since it seems to take advantage of the big search steps at the beginning and its greedy nature. Thus, PS identifies the region of the global optimum in most cases. KM provides good results which can be slightly improved using KMPS. Due to the conception of KMPS it typically requires more evaluations than KM. HKM outperforms KM and KMPS in this example and it is comparable to PS with respect to the quality of results and the number of evaluations.

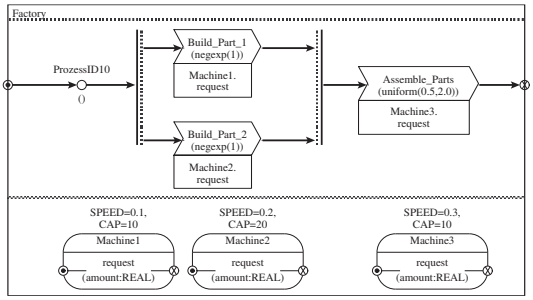

Figure 7: ProC/B Model

In the second example we consider the application of OPEDo to the optimization of a Petri Net model. It represents two machines in series whereby for each machine i ($i = 1, 2$) the buffer space b_i and the number of servers n_i ($n_i \leq b_i$) is configurable. Our goal is to maximize the throughput t (that depends on n_1, n_2, b_1 and b_2) with respect to the costs for servers and buffers as follows:

$$Z_2(n_1, n_2, b_1, b_2) = \quad t(n_1, n_2, b_1, b_2) \\ -\tfrac{1}{4}(n_1 + n_2) - \tfrac{1}{10}(b_1 + b_2)$$

The response surface of Z_2 is unimodal and therefore the local algorithms RSM and PS are expected to find results close to the optimum with a reasonable number of evaluations. As presented in Figure 6 RSM yields higher accuracy but needs more evaluations compared to PS. KM provides a lower accuracy but as in the previous model the results can be improved by using HKM or KMPS. In this example ES yields the best results but the number of evaluations exceeds that of the other algorithms by an order of magnitude.

The ProC/B-Model in Figure 7 represents a simple process chain that in a first step produces two different parts within two different stations concurrently and finally assembles two siblings in a third station. Process instances are generated according to negative exponentially distributed time intervals. The number of machines n_i, $i = 1, 2$ available in the stations 1 and 2 satisfies $n_2 = 30 - n_1$, $n_1 \in \{10, \ldots, 25\}$. Machine 3 in station 3 assembles up to ten siblings in parallel and its speed $x \in [0.1, 0.5]$ is configurable. Our goal is to minimize the turnaround time tt (that depends on n_1 resp. n_2 and x) of the production processes taking into account costs of machine 3. Thus, we have a mixed integer optimization problem that is reflected by the following goal function:

$$Z_3(n_1, x) = tt(n_1, x) + (10x)^2$$

The results of OPEDo are displayed in Figure 8. Since Z_3 is a unimodal function almost all heuristics provide results that are very close to the optimum. Only RSM differs due to fact that the response surface of Z_3 in a region close to the optimum cannot adequately be approximated by a first order linear regression model. Along with the basic optimization algorithms considered so far OPEDo also incorpo-

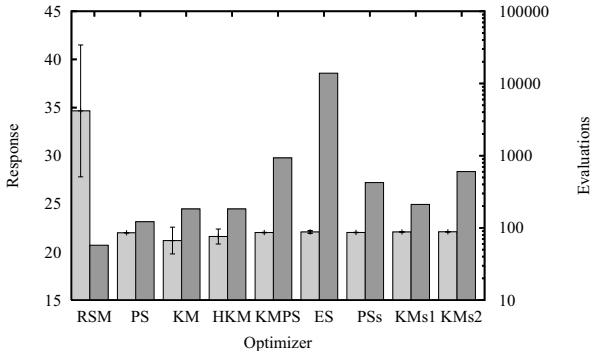

Figure 8: Results for ProC/B

rates variants that apply statistical ranking and selection methods mentioned in Sect. 3 and described in more detail in [6]. Currently, PS and KM adopt this methods and the corresponding results (PSs, KMs1 and KMs2) are also displayed in Figure 8. KMs1 differs from KMs2 that it applies statistical ranking only in the final step and KMs2 applies statistical ranking after each Kriging step. It can be noticed that by the introduction of ranking and selection methods the variance of the results is reduced.

5. CONCLUSIONS

The paper gives a brief overview of OPEDo, an open tool including optimization algorithms for models of DEDS. OPEDo contains a large number of different optimization algorithms and can be connected to different analysis tools.

OPEDo is part of our ongoing research work on optimization algorithms for DEDSs. It will be further developed and extended by integration of new optimization algorithms and possibly also by combining it with further analysis tools. Additional information about the tool is available on the Web-page `http://ls4-www.cs.uni-dortmund.de/Opedo`. The tool is available for research purposes upon request.

6. REFERENCES

[1] F. Bause, H. Beilner, M. Fischer, P. Kemper, and M. Völker. The ProC/B toolset for the modelling and analysis of process chains. In *12th Int. Conf. Modelling Tools and Techniques for Computer and Communication System Performance Evaluation*, pages 51–70, London, UK, 2002. Springer. LNCS 2324.

[2] F. Bause, P. Buchholz, and P. Kemper. A toolbox for functional and quantitative analysis of DEDS. In *Proc. 10th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation, Lecture Notes in Computer Science 1469*, pages 356–359. Springer, 1998.

[3] H. Beilner, J. Mäter, and N. Weissenberg. Towards a performance modelling environment: news on HIT. In R. Puijanger, editor, *Proc. 4th Int. Conf. on Modelling Tools and Techniques for Computer Performance Evaluation*. Plenum Publishers, 1988.

[4] D. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 2007.

[5] P. Buchholz and P. Kemper. Optimization of Markov models with evolutionary strategies based on exact and approximate analysis techniques. In *QEST*, pages 233–242. IEEE Computer Society, 2006.

[6] P. Buchholz and A. Thümmler. Enhancing evolutionary algorithms with statistical selection procedures for simulation optimization. In *ACM Winter Simulation Conference*, pages 842–852, Orlando, FL, USA, 2005.

[7] G. Casale, R. R. Muntz, and G. Serazzi. Geometric bounds: A noniterative analysis technique for closed queueing networks. *IEEE Trans. Computers*, 57(6):780–794, 2008.

[8] C. G. Cassandras and C. G. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2008.

[9] A. E. Eiben and T. Bäck. Empirical investigation of multiparent recombination operators in evolution strategies. *Evolutionary Computation*, 5(3):347–365, 1997.

[10] M. C. Fu, F. W. Glover, and J. April. Simulation optimization: a review, new developments, and applications. In *Winter Simulation Conference*, pages 83–95. ACM, 2005.

[11] L. J. Hong and B. L. Nelson. Discrete optimization via simulation using compass. *Operations Research*, 54(1):115–129, 2006.

[12] R. Hornig and A. Varga. An overview of the omnet++ simulation environment. In *Proc. of 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools)*, 2008.

[13] G. Jungman and D. B. Gough. GSL homepage.

[14] P. Kemper, D. Müller, and A. Thümmler. Combining response surface methodology with numerical methods for optimization of markovian models. *IEEE Trans. Dependable Sec. Comput.*, 3(3):259–269, 2006.

[15] J. P. C. Kleijnen. *Design and Analysis of Simulation Experiments*. International Series in Operations Research and Management Science. Springer, 2008.

[16] J. P. C. Kleijnen and J. Wan. Optimization of simulated systems: Optqest and alternatives. *Simulation Modeling Practice and Experience*, 15:354–362, 2007.

[17] A. M. Law and W. D. Kelton. *Simulation modeling and analysis*. Wiley, 2000.

[18] A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[19] T. Osogami and S. Kato. Optimizing system configurations quickly by guessing at the performance. In L. Golubchik, M. H. Ammar, and M. Harchol-Balter, editors, *SIGMETRICS*, pages 145–156. ACM, 2007.

[20] H.-P. Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, Inc., 1995.

[21] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.