OPEDo: A Tool Framework for Modeling and Optimization of Stochastic Models

Peter Buchholz
Peter Kemper
Dennis Müller
Axel Thümmler

Informatik IV
Universität Dortmund
D-44221 Dortmund, Germany
opedo@ls4.cs.uni-dortmund.de
http://ls4-www.cs.uni-dortmund.de/Opedo

ABSTRACT

A model-based design of systems requires appropriate tool support in many ways. It requires a modeling notation that suits the application problem, a set of analysis techniques that provide qualitative and/or quantitative results, and finally some optimization methods that help a designer to make appropriate design decisions. The challenge is to integrate those components into a homogenous framework such that a model based design takes advantage from synergy effects that result from a sophisticated combination of modeling formalism, analysis and optimization technique. In this paper, we present OPEDo, a tool framework that integrates modeling tools and analysis engines with state-of-the-art optimization methods. With respect to modeling, it contains the ProC/B editor for specifying open process-oriented simulation models, the APNN Toolbox for modeling with stochastic Petri nets, and OMNet++, for modeling using a simulation language. OPEDo provides analysis techniques for stochastic models based on discrete event simulation, based on queueing network analysis and numerical analysis techniques for continuous time Markov chains with the help of HIT, OMNeT++, and APNN Toolbox. Optimization of stochastic models has particular challenges due to the cost of model evaluation and the precision of results that can be achieved, so OPEDo contains specially adjusted variants of a variety of optimization methods, which includes response surface methodology, evolutionary strategies, genetic algorithms, and Kriging metamodeling techniques.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: PERFORMANCE OF SYSTEMS; G.3 [**Mathematics of Computing**]: PROBABILITY AND STATISTICS; I.6 [**Computing Methodologies**]: SIMULATION AND MODELING

ValueTools'06 , October 14, 2006, Pisa, Italy

General Terms

Algorithms, Performance, Reliability, Measurement

1. INTRODUCTION

Providing adequate tool support for the design of real world systems has been stimulating research in modeling techniques and tools for decades. A multitude of frameworks has been developed, many of those emphasize quantitative system properties. If the purpose of a framework is on decision support, there are three aspects that require consideration. First, a modeling notation that provides means to describe a model as well as to formulate requirements or properties that are of interest for a quantitative evaluation. Second, some analysis techniques that are applied to obtain quantitative values for those properties of interest. Third, a notation and methodology for optimization is necessary. The notation is used to formulate degrees of freedom for a model and to formulate objectives that shall be achieved while the optimization methodology is necessary to identify optimal configurations. The latter is based on some search technique that evaluate an objective function with results for individual models computed with some analysis techniques.

Even if we restrict ourselves to stochastic models of discrete event dynamic systems (DES), each of those three aspects above allows for a variety of choices and most existing tools support a particular combination thereof. Fu et al provide an overview in [8]. Examples for such combinations in commercial simulation tools are Automod with Autostat, which uses evolutionary and genetic algorithms, similarly for Extend with Evolutionary Optimizer, RISK with RISKOptimizer, which uses genetic algorithms, WITNESS with Optimizer, which uses simulated annealing and tabu search. An example for a more rich set of combinations is OptQuest, which can be combined with a variety of simulation engines including Arena, Crystal Ball, and ProModel, and which uses scatter search, tabu search and neural networks. However, choices for optimization methods and analysis methods are more rich than what is currently available and we see a demand for a more flexible and open solution. An optimization engine that supports a multitude of optimization methods can be combined with any analysis engine that is able to evaluate an objective function for a given model configuration. However, analysis of stochastic DES has particular

challenges that an optimization method should reflect to be efficient. Simulation results are not precise, but estimates that come with confidence intervals and small confidence intervals can require substantial computation effort. Exact numerical analysis of Markovian and specific non-Markovian models can be computationally very costly, while approximate techniques usually give results with an unknown approximation error. Most queueing network techniques are computationally inexpensive but restrictions and abstractions used to model a system by a queueing network may imply an approximation error on the modeling side.

In this paper, we describe OPEDo, the Optimization and Performance Evaluation tool from the University of Dortmund, which integrates multiple modeling formalisms and multiple analysis techniques with state-of-the-art optimization methods. In its current version, OPEDO is dedicated to single criterion optimization problems. Parameters of an objective function, resp. factors in operations research terminology, can be taken from both discrete and continuous domains. Corresponding models are parameterized in the sense that fixing values of all parameters yields a stochastic model that can be evaluated by conventional analysis techniques, for instance by a discrete event simulation. We have investigated synergy effects among analysis techniques and optimization methods, e.g., using evolutionary strategies with Ranking and Selection algorithms and simulation [6], using evolutionary strategies and exact and approximate analysis techniques of Markov Chains [4] and using response surface methodology with numerical analysis of Markov chains [11].

The rest of the paper is structured as follows. In Section 2, we describe in further detail which modeling formalisms are currently supported, how other notations can be integrated due to an open interface and how parameters can be specified. In Section 3, we briefly recall common analysis techniques that are supported, namely discrete event simulation, queueing network analysis and numerical analysis techniques for finite Markov chains. We focus in particular on those properties that are relevant for optimization methods. Section 4 briefly summarizes optimization methods that are supported in OPEDo, namely response surface methodology, pattern search, Kriging metamodels and various evolutionary strategies. Section 5 is devoted to the mutual requirements and synergy effects between analysis techniques and optimization methods and how those are addressed in OPEDo. Section 6 illustrates OPEDo's software architecture and interfaces, while Section 7 exercises an application example to shed some light on how to apply OPEDo in practice. We conclude in Section 8.

2. MODELING

The design of a modeling notation for DES is usually influenced by the envisaged application area, analysis technique, and theoretical considerations. OPEDo currently supports the ProC/B notation and stochastic Petri nets as two examples of rather complementary notations. Furthermore OMNet++ as a general simulation is integrated.

2.1 ProC/B notation

The ProC/B modeling formalism [1] is a full-fledged simulation language following the common process interaction approach for simulation modeling. ProC/B models typically describe service networks based on a hierarchy of virtual machines that provide services to their environment. A virtual machine (a functional unit in ProC/B terminology) performs a service with the help of resources it contains. Such resources can be virtual machines or some basic resource like a queueing server or a counter for passive resources with limited capacity. Services of resources are used by entities whose behavior is described with the same means as services and whose generation is described in some dedicated source nodes in case of open models. Closed models with a finite population of entities that show a cyclic behavior can be modeled as well but open models are more to the core application of ProC/B.

The definition of the ProC/B formalism originates from modeling logistic networks, however, in practice, it turns out to be a process-oriented simulation language that has a particular notation of hierarchy based on service calls and inclusion of resources. Hence it also applies to common fields of simulation modeling, like computer and communication networks, manufacturing and production systems. The optimization of a logistics system design within the ProC/B framework is presented in [5], where we have shown that the combination of the ProC/B formalism, simulation and the Repsonse Surface Methodology is beneficial.

A well-defined subset of ProC/B models can be analyzed with analytical techniques for queueing networks; in general, discrete event simulation can be applied for the evaluation of a transient or steady state behavior of a model. Common measures of interest include measures based on individual entities and types of entities like the mean and variance of lead times for particular types of entities and measures based on resources like their utilization, average response times for services of resources, and throughput of service calls and entities.

In the context of optimization, ProC/B models can be parameterized in many ways, e.g., parameters of continuous type that determine the workload intensity and the speed of active resources, and parameters of discrete type that dimension storage capacities or cardinalities of resources.

2.2 Stochastic Petri Nets

Generalized stochastic Petri Nets (GSPNs) are a modeling notation that is rooted in a well-developed theory of concurrent systems. In OPEDo, we support Petri Nets of the type supported by the APNN toolbox [2]. Petri nets in OPEDo can be hierarchically structured based on refinement of places and transitions, nets can have shared places and transitions. Places can have a finite number of colors. Transitions can have a finite number of colors and have an associated priority. In OPEDo, we allow for several types of distributions for firing transitions, which means that simulation is the best choice in the general case. However, GSPNs are included as a particular subset that can be analyzed by the numerical analysis of its associated continuous time Markov chain. E.g., in [11] we have shown how synergy effects between the Response Surface Methodology and numerical analysis can be used to reduce the overall time spend for numerical analysis while we keep the quality of the results of the optimization.

In summary, we provide a notation that is based on an arbitrary but finite number of state variables (places) that can be manipulated by a finite set of state-transformation rules (transitions). Petri nets can serve as a basis for different modeling paradigms, that can be used for modeling open systems where entities flow through a network as well as for networks of communicating automata, where communicating can be of synchronous type (rendez-vous) or asynchronous type (message-passing). Typical measures of interest that are formulated in Petri net models are throughput of certain transitions, mean values or quantils for the values of state variables (number of tokens on a place). Measures are usually defined by rewards based on the time being in some state (rate rewards) or the frequency of performing some transition (impulse reward). Since Petri nets are based on few language constructs, there is a limited number of options for defining parameters, for instance the number of tokens in the initial marking, the numerical constants for incidence functions, the value of a priority or the type of distribution and its parameter settings at some transition.

2.3 The simulation tool OMNeT++

OPEDo also integrates OMNeT++, which is a discrete event simulation environment, that has been mainly used to model communication systems and protocols, but can be applied as a general purpose simulation language. A model in OMNeT++ consists of communicating modules. Modules are hierarchically structured and communicate via exchanging messages. Message transfer can be immediate or delayed to model processing and transfer times. An activity can be implemented in a process-oriented mann or by message-passing. While the former method is easier to implement for small processes, it has serious drawbacks on the scalability of the model. Therefore larger models use the message-handling approach. Due to the hierarchical structure of OMNeT++ basic modules can be reused and several basic models, in particular for communication systems, are already available. For further details about OMNeT++ see [23]. Since OMNeT++ is written in C++ and is open source software, it can be easily modified to be used in OPEDo.

Particular application areas may imply preferences for certain specification techniques, but in any case, for optimization, one has to define parameters of a model. Models may have parameters of continuous or discrete type. Note that supporting parameterized models create an additional challenge, because models may give valid results for certain values of parameters but become faulty for others.

3. EVALUATION

The common set of automated analysis techniques for stochastic models includes analytical techniques for particular classes of queueing networks, numerical analysis techniques for stochastic processes, in particular for the class of Markov processes, and finally discrete event simulation which applies with little constraints. In the following, we briefly recall particularities of those three methodologies.

3.1 Queueing network analysis

Techniques for this class of models are computationally inexpensive and provide either exact results or good approximative results for the class of separable (product-form) queueing networks. The resulting values are mean values of common performance measures like throughput, utilization and response times. In our context, queueing network techniques can be applied for certain ProC/B models.

3.2 Numerical analysis of Markov chains

Markov chain (MC) analysis applies to Markov models with finite state spaces. Techniques take advantage of space efficient symbolic or Kronecker representations of large MCs and minimization techniques based on lumpability. A rich variety of exact and approximate computational methods are at hand. Most are based on fixed point operations such that there is often a trade-off between precision and computation effort that can be exploited in combination with an optimization method. Resulting values are rewards that aggregate the fine-grained solution of a probability distribution. In our context, MC analysis applies to stochastic Petri net models as well as to particular ProC/B models.

3.3 Discrete event simulation

Simulation of stochastic models is a technique that broadly applies with little constraints. Estimates of performance measures can be obtained for the steady state as well as for the transient case. Steady state solutions can be either obtained from a single, long simulation run with the help of the batch means method as well as with replicated, independent simulation runs [14]. For transient analysis only replicated runs apply. Confidence intervals help to estimate quality of results and an optimization method may take the width of confidence intervals into account in addition to estimates of performance measures. Simulation reaches its limits in case of models with different time scales (rare event simulation). In our context, simulation is applied to all supported modeling formalisms.

4. OPTIMIZATION

Much research has gone into the development of optimization methods and there is an ample variety known in the literature. However, according to the "no-free-lunch" theorems [24], there cannot be a methodology that is superior in general. In practice and even worse, optimization methods are typically sensitive to parameter settings of their own. Hence, our approach on that side is to provide a certain subset of optimization methods and a set of sophisticated guesses for default method configurations that are based on empirical evaluations we performed. OPEDo currently provides four different classes of optimization techniques (see Fig. 3), namely the response surface methodology (RSM) [17], pattern search [22], Kriging metamodels [9] and a family of evolutionary algorithms (EA) [20]. We give here a brief overview of the used optimization methods and refer for details to the cited literature.

4.1 Response surface methodology

RSM is a deterministic optimization method that is able to identify a (local) optimum of a response surface function. In RSM, the black-box model is evaluated for certain parameter settings that give design points in the response surface. A regression model is adjusted to match the response surface at those points and the gradient of the regression model is used to direct a stepwise search procedure towards an extreme point. Hence the initial setting determines at

which (local) optimum the procedure will terminate. Figure 1 shows a possible course of the general RSM procedure in a two dimensional search space starting in the middle of the right most square. Note that the gray lines in Figure 1 are contour lines of the real response surface, which is not known in general and shown here for the sole purpose of illustration. The graph at the bottom of Figure 1 shows the response value of the best configuration after each iteration. The course of RSM is indicated by the black line starting at the middle of the right most square. RSM begins its search with a sequence of first-order regression metamodels, combined with a steepest ascent/descent search. In that phase, four corner points of a square are evaluated, and a first-order regression model is approximated to characterize the response surface around the current center point. In the final optimization phase, RSM uses second second-order regression metamodels to estimate the response surface more accurately and search for the optimum from the resulting fit. The final part of the optimization course is not clearly visible in Figure 1. RSM can be applied for models with continuous parameters. However, some extensions exist such that also specific cases with discrete parameters can be handled.

4.2 Pattern search

Pattern search (PS) is a local search method, which belongs to the group of direct search strategies. A lot research has been done in this area, leading to different versions like Generalized Pattern Search. The version, that we are working with, has been in described in [20].

The general approach is similar to the one used by RSM though PS does not use metamodels. PS starts at a given point and searches for better points in a given distance, which decreases during the iterative optimization. Each iteration of PS can be divided into two steps, the search and the poll step. During the search step PS tests for each dimension if the value of this dimension should be increased or decreased by evaluating accordingly modified points. If neither increasing nor decreasing the value provides a better response, the value of this dimension remains unchanged. If the search step is successful it achieved a new point with an improved response. The poll step starts with the calculation of a vector of differences between the new point and the starting point of this iteration. New points are calculated in the direction of that vector and tested for an improved response until no further improvement can be determined. Like RSM, PS has been developed for continuous parameters.

4.3 Evolution strategies

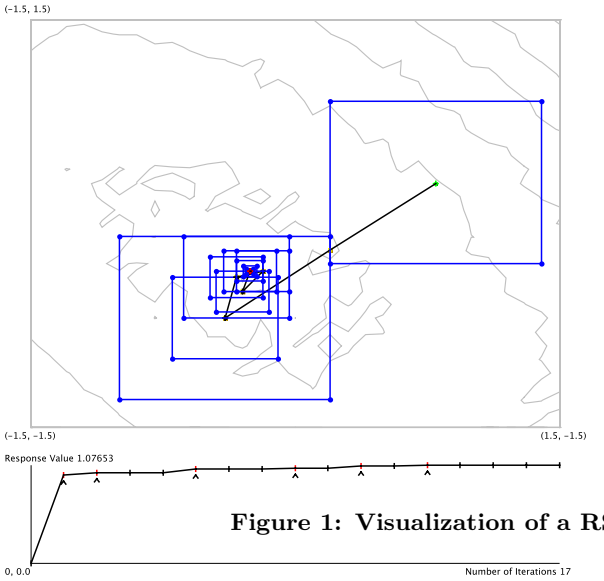
Evolution strategies (ES) [20] are a class of optimization algorithms that mimic strategies observed in the evolution of biological systems. The algorithms use a set of candidate solutions called population. A population consists of a fixed number of individuals and each individual is described by a parameter vector and a vector of strategy variables. In ES, a new generation of size λ is generated from a parent population of size μ. A new offspring results from a mutation of an element of the parent generation. The mutation operation is a randomized modification, e.g., a random value is added that is sampled from a normal distribution $N(0, \sigma)$ where the selection of σ depends on the values of strategy variables of the parent individual. In addition, some variants of ES employ recombination to obtain new individuals. Recombination uses values of two parent individuals, e.g., each parent contributes half its parameter values to the parameter vector of the new individual.

ES creates a sequence of generations. Let $\mathcal{P}^{(t)}$ be the parent population of the t-th generation. One can distinguish between $(\mu + \lambda)$-ES (plus-ES) and (μ, λ)-ES (comma-ES). In the former case the new parents are selected from the old parents and the offspring. In the latter case, new parents are selected only from the offspring. It depends on the optimization problem which of both strategies gives better results. In every generation t, a set $\mathcal{Q}^{(t)}$ of λ new candidate solutions is created from $\mathcal{P}^{(t)}$ with the help of recombination and mutation operations. The elements of the next parent population $\mathcal{P}^{(t+1)}$ are selected from $\mathcal{P}^{(t)} \cup \mathcal{Q}^{(t)}$, resp. from $\mathcal{Q}^{(t)}$, with respect to their fitness, i.e., with respect to their values of the objective function. The selection rule is deterministic and select the best individuals. ES can be applied to problems with continuous and discrete parameters.

4.4 Kriging metamodels

Kriging metamodels are a mathematical model that has been originally developed by a South African mining engineer to find gold. Afterwards they have been applied for the optimization of deterministic functions [9, 19] and more recently also for the optimization of stochastic simulation models [12]. Kriging uses a correlation model for the response surface. The model is fitted at some points in the search space where the model has been analyzed. The resulting Kriging metamodel gives, in contrast to the polynomial metamodel used in RSM, exact results at those points where experiments have been performed. For the remaining non-explored points in the search space the approach comes up with estimates for response values and variance. A fundamental but reasonable assumption of the approach is that the variance of an estimate for the response value at a non-explored point increases with an increasing distance from other explored points. So the metamodel automatically takes into account that more information is available for points in the search space that are in the neighborhood of some evaluated points than for points without evaluated points in their neighborhood. Kriging metamodels can serve as global metamodels which approximate the whole response surface and can therefore be used for global optimization by finding the point in the search space with the largest expected improvement which takes into account the expected response value and the variance.

Our optimization algorithm tries to improve the model iteratively. An experimental design, which is typically a latin hypercube sampling, is generated at the beginning. Based on that design, the first Kriging metamodel is generated. The model is optimized with respect to a given likelihood function. The algorithm iteratively identifies a point of maximal expected improvement, evaluates that point and adds it to the Kriging metamodel. The optimization iterates until a given number of evaluations has been reached. Observe that searching for the point of maximum expected improvement is a non trivial optimization problem of its own. Conceptually, the approach is described in [9]. However, the practical realization bears additional problems. In particular, the method becomes inefficient and numerically unstable if



Symbol	Meaning
Dot	Evaluated Point
Black Line	Path between Centerpoints or edge of local region
Gray line	Contour line

Figure 1: Visualization of a RSM-Optimization

the number of evaluated points becomes too large and the maximization of expected improvement is costly. In contrast to [9], we use EA for the maximization of expected improvement. This increases the quality of the approximation in some situations. To further improve the approximation we use penalized likelihood functions as described in [15].

5. COMBINATION

The main issue in the optimization of an objective function over a family of stochastic models is to find a balance between the precision of results that is delivered by an analysis technique for a given model and the precision that is required by an optimization method to find an optimum. The trade-off is between saving computation time on the side of the analyzer for the price that results are not as precise as possible and saving computation time on the side of the optimizer that performs faster on exact results. We worked on that trade-off in several combinations of analysis techniques and optimization methods and recognized that this is a valuable source of efficiency. Thus, the combination of optimization and analysis is definitely more than the simple connection of an analysis algorithm and an optimization algorithm, it has to take into account specific aspects of the used algorithms into account. Here we describe some of the combinations that have been realized in OPEDo.

5.1 Optimization with the Response Surface Methodology

RSM tends to require a modest number of function evaluations to identify a local optimum This makes it attractive for evaluations that are computationally costly, for instance, simulation as well as numerical analysis of Markov chains can require a substantial amount of computation time. RSM itself is not a specific algorithm but rather a methodology with many degrees of freedom that must be determined to achieve an automatic procedure. In [5], we propose a variant of RSM that is designed for simulation. We conducted a number of empirical evaluations to achieve a configuration that is robust and efficient. In particular, we recognized that first order models are usually sufficient and more efficient to use, such that we apply second order models only during the last phase of a search in order to get as close as possible to a local optimum. Other issues that need to be resolved are the selection of a stepsize in a line search, a lack-of-fit test to recognize if the first-order model does not represent the real surface well enough, scaling factors for the reduction of the first order model, and many more, for more details on the fully automatic algorithm for RSM we refer to [5].

The resulting RSM approach for simulation models requires some adjustments if it is applied for the numerical analysis of continuous time Markov chains (CTMCs). In [11], we discuss how RSM can be combined with steady analysis of CTMCs. We recognize that a different measure for the lack-of-fit test is necessary and suggest to use the adjusted coefficient of determination. Furthermore, we develop three heuristic strategies that reduce the computational effort for the numerical steady state analysis of the set of related CTMCs that is selected by RSM. The first strategy is to use an already computed solution of a CTMC (the steady state distribution) as an initial distribution for a CTMC whose parameters are similar (the differences are small in value). Since the solution methods are iterative fixed point methods, the heuristic can be applied whenever the set of states remains the same and only transition rates depend on parameter settings. For the second and third heuristics, we recognize that convergence of the fixed point iteration is typically measured at a fine-grain level as the residuals of the solution of a homogeneous linear equation system and that this measure may converge much slower than the value of the objective function if evaluated with intermediate results. The objective function seems less sensitive since it is usually an aggregated result and fine-grain errors rather average out than build up. We can make use of this effect and determine from some initial evaluation the relationship between convergence measures for a detailed solution and aggregated results. This relationship may lower the requirements for precision of the detailed solution significantly. Finally, this approach can be made adaptive by reapplying it whenever RSM experiences difficulties in the identification of a direction for improvement. Those three heuristics can reduce computation times for RSM with steady state analysis of CTMCs significantly (see [11] for additional details.

Both approaches, RSM for simulation models and RSM for CTMC analysis, are fully integrated in OPEDo and available for applications.

5.2 Evolution strategies

A strength of evolution strategies (ES) is their flexibility and robustness to overcome wrong decisions during the optimization process and therefore also their robustness according to functions with noise. This makes ES suitable for optimization of stochastic simulation models. However, a weakness of ES is that often a huge computational effort is required, since a large number of function evaluations are necessary to reach the optimum. This indicates the major challenge for applying ES in combination with simulation modeling in practice: The combination must be carefully designed with respect to efficency of the overall approach. Note that ES mainly uses simulation results for decision making, i.e., it distinguishes among individuals with superior or inferior values of the objective function, but absolute or precise values are not necessarily required (in most cases). This implies that simulation runs have to be just long enough to classify and rank individuals in a population and even a reasonably small percentage of false decisions is tolerable for ES.

In ES the selection rule for parents in each generation is deterministic; individuals with best response values are selected. However, when optimizing a stochastic simulation model it is not obvious which individual is the best since one can only compare the mean values of a certain number of samples. Thus, ES should select the best individuals from a population with a high probability and with low effort. In fact, this is a stochastic ranking and selection problem, which is commonly known in discrete event simulation and a large number of approaches exists [21].

In OPEDo, we include an evolution strategy that incorporates statistical procedures for the selection of best individuals. In particular, the procedures of Boesel, Nelson, and Kim [3], Chen and Kelton [7], and Buchholz and Thümmler [6] as well as traditional procedures of Rinott [18] and Koenig and Law [13] are included. In the following we consider the plus-strategy for ES but comma-strategies can be applied as well. Two different phases where statistical selection occurs in the optimization process are distinguished. With S_1 we denote the selection strategy. S_1 is used for the survivor selection during the evolutionary process, which repeatedly selects μ individuals from the $\mu + \lambda$ parents and offspring. A second statistical selection procedure, denoted as S_2, is applied for the final selection at the end of the evolutionary process, which selects the best individual from a population of τ candidate individuals, denoted the *elite population*.

OPEDo's enhanced $(\mu + \lambda)$-ES works as follows. First, the parameters of the ES are initialized and a feasible search-space is predefined and the parent population $\mathcal{P}$ and the elite population $\mathcal{B}$ are initialized. After initialization, the evolutionary process continues until a certain termination condition holds. The most common termination condition is to stop when a predefined number of generations has been passed. Other termination criteria can be to stop if the progress gets sufficiently small, i.e., if the individual of the elite population with the largest sample mean does not change for at least 10 consecutive generations.

In each generation of the evolutionary process, λ individuals are selected randomly from the parent population. Then variational operators are applied to these individuals to create the offspring. Note that the mutation operator can produce an individual which is outside the search-space. In this case, mutation is repeated until an offspring is generated that lies in the search-space. After mutation, individuals are evaluated according to selection strategy S_1. Note that the selection strategies only define the number of evaluations required for each individual. The selection itself is then performed by choosing the individuals with the largest mean values for the parent population of the next generation. Finally, when the ES stops the best individual is determined from the elite population according to selection strategy S_2.

Apart from simulation, ES can also be combined with other solution techniques. It is especially possible to use different solution techniques such that fast approximate and slow exact techniques can be combined for a very efficient optimization process as recently shown in [4].

6. ARCHITECTURE

OPEDos architecture is designed to improve the combination of state of the art analysis and optimization techniques. This should be achieved by an extensible multi-paradigm, multi-solution, multi-optimization framework that is beneficial to researchers as well as practitioners. Figure 2 illustrates OPEDos compositional architecture and its interaction with external tools. The left part, which has a light gray background, illustrates external tools that can be used to create models for OPEDo. The main part of Figure 2 illustrates the components of OPEDo and their interaction. The main components of OPEDo are the specification tool, the optimizer and the black-box model.

The specification tool is the graphical user interface (UI) of OPEDo (see Figure 3). The UI is divided into multiple areas, which are called "Global", "Model", "Optimizer", "Solve" and "Utilities". The "Global" area allows to specify the optimizer, the number of continuous and discrete parameters with their ranges of values and a set of linear constraints the parameters have to fulfill. The "Model" area provides functionality to configure the model, which includes selection of parameters and responses and definition of the response function. The "Optimizer" area always contains the label of the selected optimizer and provides access to all parameters of the optimizer. In the "Solve" area optimizations can be started and managed. Once an optimization is started, the algorithm is fully automated. The "Utilites" area provides functionality for an in depth analysis of the configured model. For instance, it is possible to evaluate a the model for a particular parameter configuration to compare the obtained results with existing data for the validation of a model.

The second component of OPEDo is the optimizer. The optimizer manages the execution of the selected optimizer implementation. Therefore it configures the environment, initializes the selected analysis method and provides functionality to store statistics. Adding a new optimizer is as simple as implementing an interface and making the optimizer known to the program. For example it took only a few lines of code to fully integrate an optimizer provided by

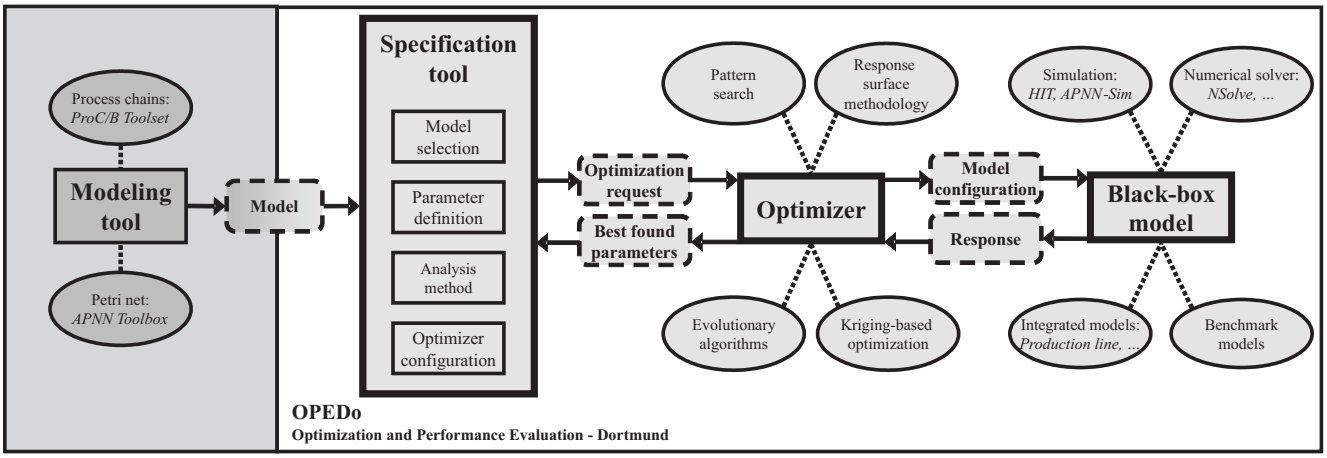


Figure 2: Architecture of OPEDo

the GNU scientific library (GSL) [10]. Currently we support RSM, EA, PS and Kriging based optimization, while having a few others implemented as test cases. A good demonstration of the power of our approach is the fact, that for the optimization of the Kriging model's parameters any of the available optimizers can be used by simply changing the selection in the UI.

The third component of OPEDo is the Black-Box model, which uniformly wraps the access to an analysis method. The Black-Box model uses the performance figures that are computed by the selected analysis method and determines a response value, which is then returned to the optimizer. The Black-Box model component provides a simple interface for analysis tools. Note that the integration of an analysis technique can be implemented in different degrees. The basic implementation provides an optimizer access to the analysis tool, calculates a response out of a set of parameter values and forwards the response to the optimizer. This implementation is a true Black-Box which, as already mentioned, often does not yield an efficient optimization approach. Thus, the interface has been extended to provide information about the progress and the state of the optimization to the analysis tool and vice versa. As we have shown in [11] this extension can be successfully used to reduce computation time, while maintaining the quality of the optimization. The resulting combination can be named as Grey Box combinations.

During an optimization the components interact as follows. The specification tool sends the selected options as an optimization request to the optimizer, which chooses and initializes the appropriate model and optimization technique. While the optimizer iteratively improves the model configuration, it sends parameter configurations to the model, which then returns a response value, which is calculated from the results of the analysis tool. Specification-tool and optimizer co-operate to keep the user informed on the current state of the optimization process. Figure 1 illustrates this for the case of RSM. The last part of the optimization is not clearly visible in the figure and OPEDo provides some zoom-functionality to overcome that difficulty. Figure 1 mainly serves for illustration purposes; the precise values are collected and accessible in different tables and graphics for subsequent text processing and documentation.

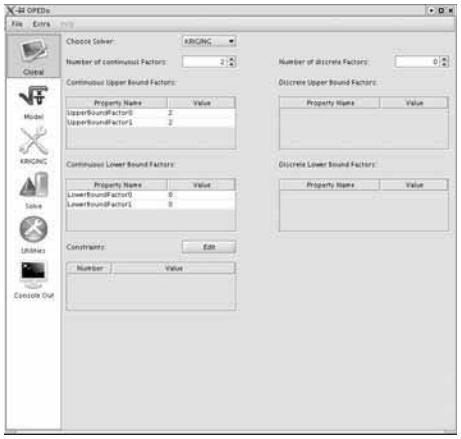


Figure 3: Specification tool

7. EXAMPLE

In this chapter we will evaluate the functionality of OPEDo. It is organized as follows. First we describe the model and the evaluation method, that we are using. Second we introduce the parameters of the model, the performance measures and the response function. Afterwards we give information about the experimental setup. In the last paragraph we evaluate the results from our experiments.

7.1 Description of the model

We will use a practical example, which is based on "Case Study II: A Web Server" from [16]. In this example, a new webserver is planned. More precisely, it is a fileserver, which has to fulfill requests for two different types of files: zip- and pdf-files. While the zip-files have a mean size of 1155.6 kilobyte, the pdf-files have a mean-size of 377.6 kilobyte. Furthermore about 60% of the requested files are zip-files.

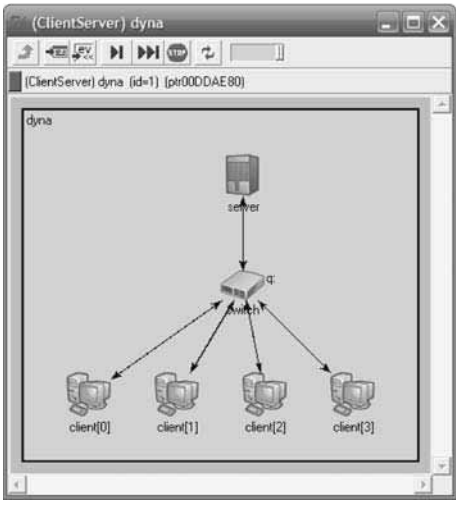


Figure 4: OMNeT++: Dyna

The server includes one cpu and six hard-disks. The hard-disks are separated into two groups. The first four hard-disks store the zip-files and the other two hard-disks store the pdf-files. This partition is motivated by the fact that the mean file size of zip-files is larger than that of pdf-files and the probability for a download request is higher than for pdf-files. Using a benchmark it has been determined that the reference cpu required about 0.1 milliseconds per processed kilobyte and that the reference harddisc requires about 0.4 milliseconds per kilobyte.

The files are transferred to the clients through a 100MBit network with a central switch (see Figure 4). The network communication is packet oriented, where request- and status-messages have a size of 64 byte and data-messages can have a size up to 16 kilobyte. Since the connection is via a local area network we consider a timeout value of 2 seconds. 60 clients are connected to the network. At peak times each client produces a request two seconds after the last request has been finished or timed out.

We use OMNet++ as analysis and simulation method. The dyna example (see Figure 4), which comes with the OMNeT++ distribution, serves as a starting point for our implementation. We keep the structure of the model, but make several changes to increase the scalability of the model.

7.2 Optimization properties

Several parameters of the model can be changed for optimization. As we are focusing on an evaluation of our optimization techniques and not on an in depth optimization of the model, we will list the properties that may be changed and then focus on two properties, that will be used during our evaluation. The parameters that may be modified are

- the speed of the cpu,
- the number of cpus,
- the speed of each hard-disk (or the speed of the hard-disks grouped by filetype),
- the number of hard-disks per filetype and
- the speed and the buffersize of the switch.

The configuration of the server will be modifiable in two aspects: The speed of the cpu and the speed of the hard-disks. Both parts can be exchanged with components that have up to ten times the speed of the reference component. We assume that the costs for the components raise quadratically with the speed. We will denote the configuration of the server as vector $\mathbf{c} = (c_0, c_1)$, where c_0 is the speed of the used cpu relative to the reference cpu and c_1 is the speed of the hard-disks relative to the reference hard-disk.

The requirements for the server are as follows. The first requirement is that during peak times the ratio of unserved to served requests should not exceed 1%. The second requirement is to minimize the cost of the server. We will denote the number of successfully requests as $s(\mathbf{c})$ and the number of unfulfilled requests as $u(\mathbf{c})$.

With this definition, the formal optimization problem is:

$$c_0^2 + c_1^2 \rightarrow min\ with\ constraint\ \frac{u(\mathbf{c})}{s(\mathbf{c})} < 0.01 \qquad (1)$$

The constrained optimization problem (Figure 1) can be transformed into the unconstrained form with the help of a penalty function

$$f(\mathbf{c}) = c_0^2 + c_1^2 + \beta \cdot \max\left\{0, \frac{u(\mathbf{c})}{s(\mathbf{c})} - 0.01\right\} \qquad (2)$$

where β is the penalty coefficient. A penalty coefficient is required to weight the penalty function to equation 1. To ensure that the penalty function is respected, we chose $\beta = 100000$.

7.3 Experimental Setup

Before we start the evaluation of the optimization techniques, we are going to evaluate the response function. Since the analysis of this model is not very time consuming, we can afford to evaluate the model on a grid over the parameters. The resulting three dimensional plot can be seen in Figure 5. As the response surface is very steep at the lower bounds of the parameters, we used a logarithmic scale on the response axis.

We are going to compare four optimization algorithms: RSM, PS, Kriging and EA. For the first three algorithms two different configurations are used, which only differ in the number of replicated model evaluations. Since our EA implementation selects the number of replication dynamically, we will use only one configuration. RSM will always analyze design-points once but center-points will be evaluated in the first configuration once and three times in the second configuration. PS and the Kriging are configured like RSM, but they only use center-points. It can easily be seen, that the response function has a global minimum near $c = (1, 2)$ and no local minima. In these situations RSM and PS normally perform better than EA. Our Kriging approach typically performs about the same as RSM and PS in these situations. But it can also be seen, that the main parts of the response

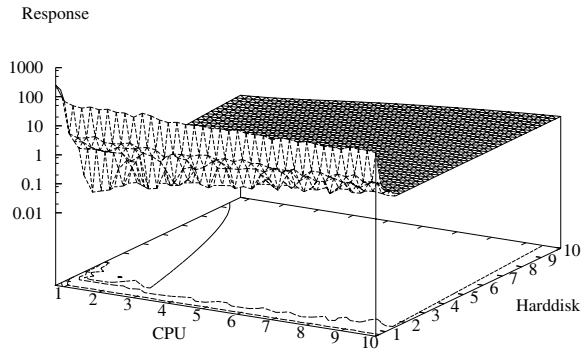


Figure 5: Response surface

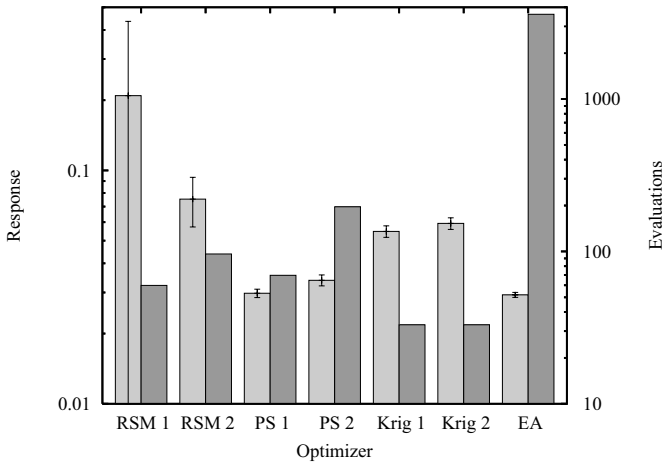


Figure 6: Results

surface are quite flat. This is typically a situation which is bad for RSM and PS. RSM and PS are able to detect low increases of the response function in the search area, but in combination with simulation errors this becomes more difficult. EA and Kriging can usually handle these situations much better. Therefore we expect the following: All optimization techniques will perform about the same, when we focus on the average result of the optimization. EA will have a tighter 95% confidence interval but also a much higher number of evaluations of the model. The configurations of the optimizers with replicated model evaluations will provide better results buts need more evaluations.

Since the model is stochastic, the behavior of all optimization methods depends on the generated stream of random numbers. Thus, replicated runs are necessary to obtain statistical significant results about the optimization quality and speed. Each algorithm has been run 121 times. Furthermore, the local search strategies RSM and PS are started from 121 different points.

7.4 Experimental Results

The results of our experiments are shown in Figure 6. For every optimization technique and configuration two bars are presented in the Figure. From left to right these are RSM without replicated evaluations (RSM 1), RSM with replicated evaluations (RSM 2), PS without replicated evaluations (PS 1), PS with replicated evaluations (PS 2), Kriging without replicated evaluations (Krig 1), Kriging with replicated evaluations (Krig 2) and EA (EA). The left bars represent the mean response with a 95% confidence interval and are related to the left vertical axis. The right bars represent the mean number of model evaluations per optimization and are related to the right vertical axis. We use a logarithmic scale because of the significant differences between some of the results on the one hand and the similarity of other results on the other hand.

Approximately the optimum has the coordinates (0.95, 1.6) and a response of 0.025. Focusing on the responses it is clear that PS and EA perform best on this model with average responses of 0.0330 and 0.0296. EA provides slightly better results but for the cost of a huge amount of evaluations. It is an interesting situation that PS without and with replicated model evaluations perform about the same. Normally this means that the results of different simulation runs do not differ significantly. In most situations RSM provides sligthly better results than PS while needing slightly more evaluations. However PS is more sensitive to local changes of the response which pays off in situations where almost no improvement is possible. Therefore RSM has problems with the calculated responses which not only results in poorer performance but also in a large 95% confidence interval. This means that RSM determines some pseudo local optima due to the error in the model and the small differences in the responses in wide areas of the parameter space. Nevertheless replicated model evaluations increase the quality of the results significantly and also reduces the confidence interval. The Kriging approach performs better than RSM but not as good as PS and EA, but it can also be seen that the confidence intervals are rather small and also the number of model evaluations is less than half of the number of model evaluations required by PS without replicated model evaluations.

8. CONCLUSION

OPEDo is a general and open framework for the optimization of stochastic models of discrete event systems. It supports different modeling formalisms, namely, hierarchical stochastic Petri nets, stochastic automata networks, process chains and simulation models. A model that is considered for optimization provides a set of parameters that can be modified by the optimization method in a predefined discrete or continuous range respecting linear constraints. For any fixed setting of parameters, we obtain a model configuration that can be evaluated. OPEDo supports a variety of analysis methods to compute performance and dependability measures of a given model configuration. In the current version, this set of methods includes analytical techniques for queuing networks, numerical techniques for Markov models, approximate decomposition techniques and discrete event simulation. Note that the use of a specific technique may be model dependent. Modeling formalisms and analysis techniques are implemented with the help of the APNN toolbox [2], the ProC/B Toolset [1] and OMNeT++ [23]. In addition, OPEDo provides an open inter-

face to combine its optimization methods with other modeling formalisms and tools as well as to combine the integrated modelling formalisms and evaluation methods with additional optimization methods. In this regard it is a platform for research as wells as applications in the optimization of complex models of discrete event systems.

9. 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 R. Puigjaner, N. Savino, and B. Serra, editors, *10th Int. Conf. on Computer Performance Evaluation - Modelling Techniques and Tools (TOOLS 1998)*, pages 356–359, Palma de Mallorca, Spain, 1998. Springer. LNCS 1469.

[3] J. Boesel, B. Nelson, and S. Kim. Using ranking and selection to clean up after simulation optimization. *Operations Research*, 51:814–825, 2005.

[4] P. Buchholz and P. Kemper. Optimization of Markov models with evolutionary strategies based on exact and approximate analysis techniques. Technical report, Submitted for publication, 2006.

[5] P. Buchholz, D. Müller, and A. Thümmler. Optimization of process chain models with Response Surface Methodology and the ProC/B toolset. In H. Günther, D. Mattfeld, and L. Suhl, editors, *Entscheidungsunterstützende Systeme in Supply Chain Management und Logistik*, pages 551–573. Physica-Verlag, 2005.

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

[7] E. Chen and W. Kelton. An enhanced two-stage selection procedure. In *Proc. Winter Simulation Conference*, pages 727–735, Orlando, FL, 2000.

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

[9] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[10] G. Jungman and D. B. Gough. GSL homepage, http://www.gnu.org/software/gsl/.

[11] P. Kemper, D. Müller, and A. Thümmler. Combining Response Surface Methodology with numerical models for optimization of class based queueing systems. In *Int. Conf. on Dependable Systems and Networks (DSN)*, pages 550–559, Yokohama, Japan, 2005.

[12] J. P. C. Kleijnen and W. C. M. van Beers. Kriging interpolation in simulation: a survey. In *Proc. Winter Simulation Conference*, pages 113–121. IEEE, 2004.

[13] L. Koenig and A. Law. A procedure for selecting a subset of size m containing the l best of k independent normal populations, with applications to simulation. *Communications in Statistics Simulation and Computation*, 14:719–734, 1985.

[14] A. M. Law and W. D. Kelton. *Simulation modeling and analysis, 3rd edition*. Wiley, 1999.

[15] R. Li and A. Sudjianto. Analysis of computer experiments using penalized likelihood in Gaussian Kriging models.

[16] D. Menasce, V. Almeida, and L. Dowdy. *Performance by Design*. Pearson Education, Inc., 2004.

[17] R. H. Myers and D. C. Montgomery. *Response Surface Methodology*. Wiley, 2002.

[18] Y. Rinott. On two-stage selection procedures and related probability-inequalities. In *Communications in Statistics Theory and Methods A7*, pages 799–811, 1978.

[19] T. J. Santner, B. J. Williams, and W. Notz. *Design and Analysis of Computer Experiments*. Springer, 2003.

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

[21] Swisher, S. Jacobson, and E. Yücesan. Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey. In *ACM Transactions on Modeling and Computer Simulation 13*, pages 134–154, 2003.

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

[23] A. Varga. OMNeT++, http://www.omnetpp.org.

[24] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.