

Combining Response Surface Methodology with Numerical Models for Optimization of Class-Based Queueing Systems*

Peter Kemper, Dennis Müller, and Axel Thümmler

University of Dortmund

Department of Computer Science, 44221 Dortmund, Germany

<http://ls4-www.cs.uni-dortmund.de/MuS>

Abstract

In general, decision support is one of the main purposes of model-based analysis of systems. Response surface methodology (RSM) is an optimization technique that has been applied frequently in practice, but few automated variants are currently available. In this paper, we propose the combination of RSM with numerical analysis methods to solve continuous time Markov chain models of class-based queueing systems (CBQ). We consider first- and second-order models in RSM to identify an optimal parameter configuration for CBQ as part of the differentiated service architecture. Among the many known numerical solution methods for large Markov chains, we consider a Gauss-Seidel solver with relaxation that relies on a hierarchical Kronecker representation as implemented in the APNN Toolbox. To effectively apply the proposed optimization methodology we determine a suitable configuration of RSM and compare the results with previous results for optimizing CBQ.

1. Introduction

In model-based design of computer and communication systems, optimization techniques can help to identify optimal or nearly optimal configurations to support decision-making. From a conceptual point of view, an optimization procedure searches for those parameter settings that maximize or minimize a given objective function f . In many model-based designs f depends on a stochastic model of a discrete-event system and different opportunities for its evaluation exist. In the case of finite state Markov chains, numerical methods are known that give exact results with respect to a transient or steady state distribution. Furthermore, simulation is an approach that is widely applicable due to its relative lack of

constraints. However, simulation of stochastic discrete event systems yields only estimates of the performance measures, typically accompanied by confidence intervals [12].

In this paper, we consider f to be defined on a family of continuous time Markov chains (CTMCs). Numerical analysis of CTMCs has a number of challenges. In particular, the generator matrix \mathbf{Q} of a CTMC is often large and very sparse. That has stimulated a lot of research on data structures that represent \mathbf{Q} in a space-efficient way and iterative solution methods that converge quickly to the resulting distribution. In the current situation, the space used for the resulting distribution is the bottleneck in terms of space if state-of-the-art representations of \mathbf{Q} are employed, i.e., symbolic structures like multi-terminal binary decision diagrams or matrix diagrams and structures based on a matrix algebra like modular and hierarchical Kronecker representations; see [14], [4] for recent overviews. An ample variety of numerical methods exist for steady state analysis (see [18] for a textbook overview); however, no technique is known yet that is clearly superior in general. However, a common property of all techniques is that computations to obtain exact results are relatively costly. That has considerable impact on the selection of an optimization procedure to be applied for objective functions that require numerical analysis of a CTMC.

Optimization is a research area with a long tradition, particularly in the field of operational analysis, which has given rise to a wealth of techniques. In particular, genetic algorithms and evolutionary strategies have gained a lot of attention recently. Despite their impressive performance in many areas, the fact that those techniques tend to require an evaluation of the objective function at many parameter settings has motivated us to focus on a different and also well-established method, namely the response surface methodology (RSM) [15]. It was originally developed for optimization based on real

* This material is based upon work supported by DFG, SFB 559, and DoMuS, University of Dortmund.

experiments, but can be easily extended for use with stochastic models. However, although RSM has been known for a long time and a well-established theory has been developed, the algorithm is usually described in such a way that several steps must be done manually, so that the approach cannot be integrated in an optimization package in which the whole experimentation and optimization approach is done automatically.

Recently, Neddermeijer, Oortmarssen, Piersma, and Dekker proposed a framework for response surface methodology for optimization of simulation models [16]. Their framework shows a possible way to combine the various mathematical and statistical methods that belong to response surface methodology. Nevertheless, their framework is still far from being an automated algorithm. Kleijnen and Sargent proposed a procedure for linear regression metamodeling in random simulation [11]. Their approach distinguishes between fitting and validating a single metamodel with respect to an underlying simulation model. To the best of our knowledge, until now no fully automated realization of the response surface methodology that is tailored to the optimization of computationally expensive numerical models has been available.

In this paper, we present a novel approach for the optimization of numerical models that is based on the response surface methodology. Our approach iteratively uses first- and second-order linear regression metamodels combined with a gradient-based method to find a direction of improvement. Of particular importance is the algorithmic realization of RSM such that it can run with very limited information or support from a user. Since RSM is able to tolerate imprecise evaluation of the objective function to a certain extent, we make use of this effect to adaptively adjust the precision of an iterative procedure being applied to the steady state analysis of a CTMC. Furthermore, we observe and make use of the phenomenon that values of an objective function that result from an iterative solution may converge much faster than the iterative solution itself. That gives rise to a novel, adaptive, and heuristic approach for RSM optimization with numerical solution methods. The approach trades computation time for accuracy. As an application example, we consider the optimization of a class-based queueing system [7], [13]. Class-based queueing (CBQ) is a “per hop” packet-scheduling mechanism that provides differentiated service to traffic flows of different types and is used as part of the differentiated service architecture (DiffServ) [9]. According to [3], we develop a stochastic Petri net model of a CBQ system and apply numerical analysis based on a Gauss-Seidel solver with relaxation for a

hierarchical Kronecker representation of the generator matrix. The APNN toolbox is used for modeling and analysis [2].

The paper is organized as follows. Section 2 develops the fully automated RSM algorithm. Section 3 considers the analysis of stochastic models via numerical methods and develops three strategies for combining RSM with numerical models. In Section 4, a stochastic model of a class-based queueing system is developed, and the optimization task is defined. Section 5 presents experimental results.

2. Response Surface Methodology

From an abstract mathematical point of view, a stochastic model can be represented by a function $\phi(\mathbf{w})$, which maps a vector of input parameters $\mathbf{w} = (w_1, \dots, w_k)$ onto a set of performance measures of interest. Since ϕ represents the stochastic nature of a stochastic model, the output performance measures are random variables, and different characteristics of their distributions such as mean, variance, further moments, or quantiles could be of interest. That relationship is expressed by $M[\phi(\mathbf{w})] = f(\mathbf{w})$, where f is called the *response surface function* and M is a mapping from the random variables onto a real-valued algebraic combination of certain characteristics of their distributions. The most frequently considered characteristic is the expectation of those random variables; however, any other measure can be used instead, given that the analysis method employed to compute $\phi(\mathbf{w})$ is able to support it.

The general optimization problem discussed in this section is characterized by finding the input parameters that maximize/minimize the response surface function. Since ϕ is only implicitly represented as a stochastic model, or in other words as a *black box*, only those optimization methods that do not exploit the structure of ϕ can be applied. To solve that optimization

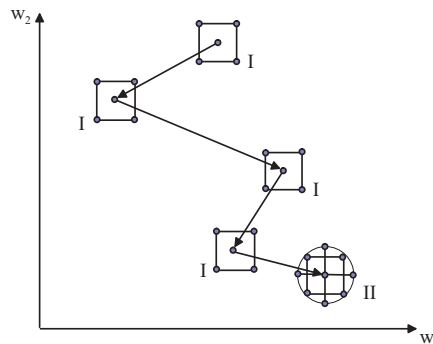


Fig. 1. Illustration of the Response Surface Methodology in two dimensions

```

(1) Transform natural variables into coded variables
(2) Choose initial center point  $\mathbf{c}_{\text{new}} := (0, \dots, 0)$  and half-width of local region  $\omega := 0.2$ 
(3) WHILE  $\omega > \omega_{\text{stop}}$  DO
(4)    $\mathbf{c}_{\text{old}} := \mathbf{c}_{\text{new}}$ 
(5)   Transform local region with center point  $\mathbf{c}_{\text{old}}$  into a  $[-1, 1]^k$  hypercube
(6)   Approximate response surface function in the local region with center point  $\mathbf{c}_{\text{old}}$  and at least  $k$  design points by a first-order linear regression metamodel
(7)   Test the first-order model for goodness-of-fit according to the adjusted coefficient of determination
(8)   IF first-order model is adequate THEN DO
(9)     determine direction of steepest ascent/descent and step size
(10)    REPEAT
(11)      go one step in direction of steepest ascent/descent and determine the response for the new input parameters via a single evaluation of the stochastic model
(12)    UNTIL new response results in no further improvement
(13)     $\mathbf{c}_{\text{new}} :=$  input parameters of last improvement of the response
(14)    IF  $\mathbf{c}_{\text{new}} = \mathbf{c}_{\text{old}}$  THEN set new half-width  $\omega := \omega/2$ 
(15)  ELSE DO
(16)    IF  $\omega < 4 \cdot \omega_{\text{stop}}$  THEN DO
(17)      Approximate the response surface in the local region with center point  $\mathbf{c}_{\text{old}}$  by a second-order linear regression metamodel
(18)      Test the second-order model for goodness-of-fit according to the adjusted coefficient of determination
(19)      IF second-order model is adequate THEN DO
(20)        Determine stationary point of the second-order model
(21)        Determine type of stationary point according to the signs of the eigenvalues of matrix  $\mathbf{B}$ 
(22)        IF type of stationary point conforms with optimization goal THEN  $\mathbf{c}_{\text{new}} =$  stationary point
(23)      OD
(24)    OD
(25)    set new half-width  $\omega := \omega/2$ 
(26)  OD
(27) OD
(28) RETURN optimal solution  $\mathbf{c}_{\text{new}}$ 

```

Fig. 2. Pseudo-code of the RSM optimization algorithm

problem, we use the *Response Surface Methodology (RSM)* [15]. RSM is an optimization method that is able to identify a (local) optimum. In contrast to, for example, evolutionary algorithms [17] and simulated annealing [10], it is a deterministic optimization method. In RSM, local marginal effects of the stochastic model are estimated by regression metamodels to find a direction of improvement. Fig. 1 shows a possible course of the general RSM procedure in a two-dimensional search space. In the local-exploration phase (see phase I in Fig. 1), RSM uses a sequence of first-order regression metamodels, combined with a steepest ascent/descent search. In that phase, four points in a square are simulated, 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 a second-order regression metamodel (see phase II in Fig. 1) to estimate the optimum from the resulting fit.

Fig. 2 shows the proposed RSM algorithm in high-level pseudo-code. It operates in a fully automated algorithmic way. When starting the algorithm, one has to choose the lower and upper limits of each input parameter in order to transform the whole search space into a $[-1, 1]^k$ hypercube, i.e., to transform the natural variables into coded variables (see step (1) in Fig. 2). Furthermore, an initial center point \mathbf{c}_{new} and an initial half-width ω of the local region in the response surface

must be specified. If not explicitly defined differently, we assume $\mathbf{c}_{\text{new}} = (0, \dots, 0)$ as the initial value for the center point and $[-0.2, 0.2]^k$ for the local region. Local regions of the size $[-0.2, 0.2]^k$ up to the size of $[-0.4, 0.4]^k$ are typically good choices. As we will later deal with a function that is only partially defined within the lower and upper bounds of the input parameters, a small local region seems to be appropriate. The main steps of the algorithm are performed in the while-loop from step (3) to step (27) in Fig. 2. In our implementation, we consider two types of stopping criteria, i.e., stop the RSM iteration if (i) the estimated optimal response is no longer improving sufficiently or (ii) the local region becomes too small. Note that the pseudo-code in Fig. 2 implements the second stopping rule, where ω_{stop} is the half-width of the local region when the algorithm should stop. An implementation of the first criterion is quite similar.

In each RSM iteration, the current local region is transformed into a $[-1, 1]^k$ hypercube. Then the response surface function f is approximated by a first-order linear regression metamodel (see e.g. [15]). To test whether the estimated regression model adequately describes the behavior of the response in the current region of interest we consider the adjusted coefficient of determination, which is defined as the ratio of the variation explained by the metamodel to the total variation (see also [15]). If the first-order model gives

an adequate approximation, a line-search along the path of steepest ascent/descent is applied to find a point of improved response that serves as a new center point. If the line-search does not yield an improved response the algorithm reduces the half-width of the local region at the last center point and starts a new RSM iteration. That is exactly what is implemented in steps (4) to (14) in Fig. 2. If the first-order model is found to be no adequate approximation of the response surface function, it is likely that the true response surface has significant curvature in the local region and may be better approximated by a second-order quadratic metamodel. Nevertheless, we recommend the approximation of a second-order model only in the final steps of the optimization procedure (i.e., if ω is less than 4 times the stop-width ω_{stop}), since a second-order model requires much more evaluations of the stochastic model than a first-order model does. Thus, a better strategy is to decrease the local region in order to better approximate a first-order model (see steps (14) and (25) in Fig. 2).

In the final optimization phase, it may be reasonable to use a second-order model. As for the first-order model, a goodness-of-fit test is performed. If the approximated model shows no significant lack-of-fit, a point of improved response is predicted with a canonical analysis, i.e., the stationary point is derived from the first derivative of the regression metamodel (see steps (19) to (23) in Fig. 2). The nature of the stationary point, i.e., whether it is a maximum, a minimum, or a saddle point, can be determined by inspecting the signs of eigenvalues of a certain symmetric matrix \mathbf{B} that is composed of the regression coefficients (see [15] for further details). At the end of all RSM iterations, the algorithm returns the center point of the local region, after transformation from coded to natural variables, as the optimal/best solution it could find.

3. Optimizing Numerical Models

The RSM approach of the foregoing section requires a method to evaluate the responses y_i for any given set of values of input parameters. In this section, we discuss how iterative numerical procedures of steady state analysis for CTMCs can be combined with RSM. Numerical analysis of a CTMC with generator matrix \mathbf{Q} computes values for a set R of rate or impulse rewards by computation of a steady state distribution $\boldsymbol{\pi}$. Distribution $\boldsymbol{\pi}$ is a solution of $\boldsymbol{\pi}\mathbf{Q} = 0$. Due to the large dimensions of \mathbf{Q} , i.e., the large cardinality of the set of states S , and sparsity of \mathbf{Q} , it is common practice to employ iterative fix point

algorithms to solve $\boldsymbol{\pi}\mathbf{Q} = 0$; see [18] for a comprehensive textbook explanation. Simple methods for stationary analysis are the power method, the method of Jacobi, and the method of Gauss-Seidel. More involved methods include projection methods like GMRES and the method of Arnoldi. Decompositional methods like iterative Aggregation / Disaggregation methods and recent Multi-Level methods aim to solve equations at different levels of granularity. Selection criteria for application on a particular CTMC are speed of convergence, computation time, and memory requirements (plus availability in a tool). Since there is no clear best choice known for speed of convergence, required CPU time, in general, we focus on memory requirements.

Generator matrices \mathbf{Q} are usually automatically generated from some modeling formalism and typically provide some structure such that symbolic representations like MTBDDs, MxDs or Kronecker representations are particularly space-efficient, which moves the bottleneck in terms of space to the iteration vectors; see [14], [4] for a recent overview. Those data structures are based on a divide and conquer approach that makes effective use of repeated information in \mathbf{Q} , such that it is represented in space just once but potentially used more often. Numerical methods differ in the number of vectors they require; for instance; Gauss-Seidel can be implemented with a single iteration vector, while projection methods need a significant number of vectors to represent a Krylov subspace. Therefore, we focus on Gauss-Seidel with relaxation (SOR) in the following. For any given iterative solution method, we need to decide on the initial distribution $\boldsymbol{\pi}_0$ and the required precision ϵ .

Selection of initial distribution. RSM repeatedly solves CTMCs of the same model but for some modified input parameter values \mathbf{w} , i.e., a set W of configurations is considered with corresponding generator matrices \mathbf{Q}_w for $\mathbf{w} \in W$. A modification of parameters may have different effects on \mathbf{Q}_w .

a) Dimensions and non-zero structure of all \mathbf{Q}_w in $\{\mathbf{Q}_w \mid \mathbf{w} \in W\}$ are the same, only numerical values at certain positions change. Those changes can be significant for the performance of numerical solution methods, since one may modify rates to introduce or remove different time scales in the model. However, evaluation of a set of matrices may imply that \mathbf{Q}_w can be derived from the generator matrix $\mathbf{Q}_{w'}$ of a previous configuration \mathbf{w}' without a costly generation from the model description.

b) Dimensions and structure of the CTMC are changed. In that case, \mathbf{Q}_w has to be generated from the model and solution has to start from the beginning.

In the following, we focus on case (a). It is well-known that selection of $\boldsymbol{\pi}_0$ has an impact on the number of iterations needed to compute solution $\boldsymbol{\pi}$. The repetitive application of a numerical solver to CTMCs of the same dimension but for different entries allows us to start a subsequent SOR solution with some previously computed $\boldsymbol{\pi}$. Clearly, the more similar those CTMCs (as seen by the distance in the Euclidean space in Fig. 1) are, the more likely it is that a previously computed $\boldsymbol{\pi}$ for one CTMC will give a good initial distribution for another. Note that design points of the first experimental design are rather far apart but that RSM reduces the width of the local region, i.e., the distance between design points, the closer it gets to its termination. That can imply similarity of CTMCs and their solutions. In consequence, we formulate a heuristic strategy for selection of $\boldsymbol{\pi}_0$.

Strategy 1: For the initial distribution $\boldsymbol{\pi}_0$ of the currently considered configuration \mathbf{w} , use a previously computed solution $\boldsymbol{\pi}'$ of a configuration \mathbf{w}' that is close to \mathbf{w} .

In RSM, we evaluate design points of first- or second-order models and design points during the line-search. In a first-order model, the center point is close to the design points in the Euclidean space. In a second-order model with a cyclic evaluation of design points and during the line-search, the configuration considered directly before the current one is the closest. Thus, the selection has the additional positive effect that it is not necessary to store many previously computed solutions.

Selection of precision. It is common practice to use several measures of distance to decide convergence; for example, they might include the maximum difference between consecutive iteration vectors $d_1 = \max_{s \in S} \{|\boldsymbol{\pi}_i(s) - \boldsymbol{\pi}_{i+1}(s)|\}$, the maximum residual $d_2 = \max_{s \in S} \{|\mathbf{v}(s)|\}$, and the sum of residuals $d_3 = \sum_{s \in S} |\mathbf{v}(s)|$ where $\mathbf{v} = \boldsymbol{\pi}_i \mathbf{Q}$. Convergence is identified if $d_k \leq \varepsilon$ for $k = 1, 2, 3$ for some user-given threshold value ε . Note that all three measures are approximations. The response function makes use of $\boldsymbol{\pi}$ with the help of rate or impulse rewards. Since it is common practice to encode impulse rewards in the state space [6], we can consider a set X of rewards r_x with reward vectors \mathbf{r}_x of non-negative real values and dimension $|S|$. The reward evaluation results in computation of weighted sums

$$\mathbf{r}_x = \sum_{s \in S} r_x(s) \boldsymbol{\pi}(s) \text{ for } x \in X. \quad (1)$$

Note that associated reward values often do come from a finite set of real values of small cardinality. Let R_x denote the set of such values; then

$$\mathbf{r}_x = \sum_{r \in R_x} r \sum_{s \in S, r_x(s)=r} \boldsymbol{\pi}(s). \quad (2)$$

Hence, if iteration vectors differ only in such a way that the inner sum in Eq. (2) remains mainly unchanged, the reward value will remain the same. Therefore, for iteration vectors of steady state solvers, a sequence r_x based on intermediate vectors $\boldsymbol{\pi}_i$ will converge due to convergence of $\boldsymbol{\pi}_i$, but the convergence rates may differ. Since RSM relies on a sufficiently accurate evaluation of the response function f , it may be possible to stop the iteration procedure for rather large values of ε if the evaluation of the response function has converged, i.e., we can define $d_4 = |f(\mathbf{w}, \boldsymbol{\pi}_i) - f(\mathbf{w}, \boldsymbol{\pi}_{i+1})|$ as a further measure for convergence where $f(\mathbf{w}, \boldsymbol{\pi}_i)$ gives the response for an intermediate vector $\boldsymbol{\pi}_i$ and a model configuration \mathbf{w} . Since in general, the convergence of the response function or its reward functions is as unknown as the convergence of $\boldsymbol{\pi}_i$ vectors, it either requires intermediate evaluations or is only known a posteriori once the numerical procedure has converged. The evaluation of d_1, \dots, d_4 implies different efforts. While d_1 is computationally inexpensive, d_2 and d_3 are inexpensive for the Power method and Jacobi but require an additional matrix-vector multiplication for SOR. d_4 requires the evaluation of a set of rewards X and the subsequent evaluation of the algebraic expression of f . Hence, we look for ways to avoid the computation of solutions that are unnecessarily precise as well as frequent evaluation of d_4 .

For certain models, the assumption that all matrices $\{\mathbf{Q}_w \mid \mathbf{w} \in W\}$ have a similar correspondence between the rate of convergence measured by d_1, \dots, d_3 and d_4 is a heuristic that can be used to adapt parameters for a numerical solver, particularly the required accuracy ε . We propose the following heuristic strategy for RSM:

Strategy 2:

1. **Inspection step:** Evaluate the points of the experimental design of the first local region that RSM considers with the procedure presented in Fig. 3 in order to determine for which $\varepsilon = \varepsilon_0$ one does not observe a significant change of the response.
2. **Main step:** Evaluate the remaining points during the RSM run just with precision ε_0 to save iterations at the price of a potential lack of accuracy.

Note that the inspection step implies that the procedure adapts itself at the beginning of the RSM optimization process. Clearly, there is a lot of room for variations based on the decision at which design points ε_0 is determined and whether it should be determined once, occasionally, or frequently during the RSM algorithm. The inspection step itself is divided into two parts. In steps (2) to (6) in Fig. 3 the Gauss-Seidel solver is forced to continue iterating until the relative difference between consecutive response values is

- (1) $\boldsymbol{\pi}^{(0)} := \boldsymbol{\pi}_0$; $y^{(0)} := f(\boldsymbol{\pi}_0)$; $i := 0$
- (2) **REPEAT**
- (3) $i := i + 1$
- (4) start Gauss-Seidel solver with $\boldsymbol{\pi}_0 := \boldsymbol{\pi}^{(i-1)}$ until maximum residual d_2 is below $\varepsilon = 10^{-i}$
- (5) compute reward measures and evaluate response function $y^{(i)} := f(\mathbf{w}, \boldsymbol{\pi}^{(i)})$
- (6) **UNTIL** $|(y^{(i-1)} - y^{(i)}) / y^{(i-1)}| < 10^{-2}$
- (7) $k := i$
- (8) **REPEAT**
- (9) $i := i + 1$
- (10) start Gauss-Seidel solver $\boldsymbol{\pi}_0 := \boldsymbol{\pi}^{(i-1)}$ until maximum residual d_2 is below $\varepsilon = 10^{-(k + (i-k)/10)}$
- (11) compute reward measures and evaluate response function $y^{(i)} := f(\mathbf{w}, \boldsymbol{\pi}^{(i)})$
- (12) **UNTIL** $|(y^{(i-1)} - y^{(i)}) / y^{(i-1)}| < 10^{-4}$
- (13) **RETURN** $\varepsilon_0 := \varepsilon$

Fig. 3. Inspection step of RSM strategy 2

below a threshold 10^{-2} where response values $y^{(i)}$, $i=1,2,\dots$, are computed from intermediate distributions $\boldsymbol{\pi}^{(i)}$ only if $d_2 < 10^{-i}$, i.e., if the accuracy of the intermediate distribution is increased by one order of magnitude. In the second part of the inspection step (see steps (8) to (12) in Fig. 3), the response function is evaluated more frequently in order to determine the value of ε_0 more precisely such that the relative difference between the responses is below 10^{-4} . The reason for not applying the inspection step for every evaluation of the stochastic model is simply to avoid the effort of a multiple computation of the response function during the iterations of the equation solver in the main step of the RSM optimization process.

Recall that RSM estimates a direction of steepest ascent/descent with the help of a first-order model that only approximates the real response surface. Hence, some lack of precision from the estimation would add only to the imprecision of the gradient estimation and need not be critical. A determination of threshold values for that trade-off is clearly model-dependent. Nevertheless, the following assumption seems natural. Since RSM initially considers rather large regions and it is likely to be far from the optimal solution, a very moderate precision is supposed to be sufficient to identify a promising direction. In a later phase and closer to an optimum, a higher accuracy is necessary to identify an optimum or to get close to it. That results in the third heuristic strategy that we propose.

Strategy 3: Start with an initial precision of ε_0 and increase the accuracy if RSM is forced to reduce the local region of a first-order model significantly.

In strategy 3, the initial precision ε_0 results from the inspection step of strategy 2. For our implementation, we considered a reduction of the half-width ω of the local region below 0.06 (in coded variables) to be significant.

4. Optimizing Class-Based Queuing

In a class-based queuing system (CBQ), the requirement for link-sharing is to share bandwidth on a network link between multiple traffic classes, where each traffic class wants to receive a guaranteed share of the link bandwidth during congestion, but where bandwidth that is not being used by one traffic class should be available to other traffic classes sharing the link. In CBQ, a conceptual distinction is made between a *general scheduler* and a *link-sharing scheduler* [7]. The general scheduler is usually a weighted Round Robin scheduler (WRR) that is controlled by the link-sharing scheduler (see Fig. 4). Incoming traffic is classified into the appropriate queue according to a set of filtering rules. The general scheduler selects packets from the queues in a way that guarantees for each traffic class at least a predefined bandwidth portion of the overall output link bandwidth. A traffic estimator, located at the output link, measures the departure time between successive packets of each class and characterizes each class as over-limit, under-limit, or at-limit. A class is called over-limit if it has recently used more than its guaranteed bandwidth, under-limit if it has used less than its guaranteed bandwidth, and at-limit otherwise. If a queue of a certain traffic class is empty, the link-sharing scheduler distributes the excess bandwidth among the remaining classes. Furthermore, if congestion occurs, the link-sharing scheduler makes over-limit classes inactive so that the general scheduler does not serve them until their suspension period ends.

For modeling the class-based queuing system we used the generalized stochastic Petri nets (GSPN) notation [1]. According to [3], the model is arranged in a hierarchical structure such that efficient solution methods based on Kronecker representation can be applied. In particular, we consider three traffic classes, each of which comprises one sub-model. Furthermore, the arrival process of data packets is contained in a fourth sub-model. For traffic modeling we considered a 5-state phase-type distribution, which was fitted to the LBL-TCP-3 trace from in the Internet Traffic Archive [8]. For distribution fitting, we used the tool G-FIT [5]. The GSPN model, as well as a detailed description of its behavior, is available for download on the Web page of the APNN toolbox [2].

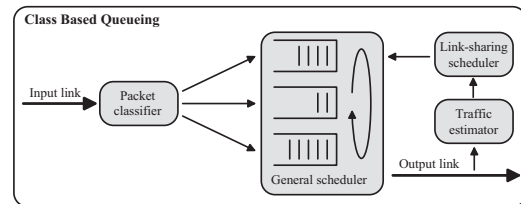


Fig. 4. Class-based queuing scheduler

In the following optimization problem, the weights of WRR should be configured such that the sum of the average delays experienced by packets of each traffic class is minimized under a given delay constraint for each class. We assume that the QoS requirements of traffic class i are such that the average delay for each class should be less than a predefined class-specific bound D_i^* , $i = 1, 2, 3$.

From the stationary solution of the CBQ model, performance measures of interest can be computed. We consider performance measures that depend on the setting of WRR weights $\mathbf{w} = (w_1, w_2, w_3)$. Denote the mean queue length of class i by $MQL_i(\mathbf{w})$ and the throughput of class i by $THU_i(\mathbf{w})$. Applying Little's law, the delay of class i is given by $D_i(\mathbf{w}) = MQL_i(\mathbf{w}) / THU_i(\mathbf{w})$. With that definition, the formal optimization problem is:

$$\sum_{i=1}^3 D_i(\mathbf{w}) \rightarrow \min \text{ with constraints} \quad (3)$$

$$D_i(\mathbf{w}) \leq D_i^*, i = 1, 2, 3, \text{ and } \sum_{i=1}^3 w_i = 1.$$

The constrained optimization problem (3) can be transformed into the unconstrained form with the help of a penalty function

$$f(\mathbf{w}) = \sum_{i=1}^3 D_i(\mathbf{w}) + \beta \sum_{i=1}^3 \max\{0, D_i(\mathbf{w}) - D_i^*\}, \quad (4)$$

where $f(\mathbf{w})$ is the response function that shall be minimized and β is a penalty coefficient. Furthermore, we assume that the weights are normalized before we evaluate $f(\mathbf{w})$, so we can omit the second constraint in (3). Note that the penalty coefficient β is a fixed value, so the response function (4) remains the same during the optimization process. This is different in [3] where β is changed during the optimization procedure.

5. Experimental Results

In this section, we evaluate the CBQ model for a particular configuration and workload with the goal of optimizing the setting of WRR weights \mathbf{w} . The unconstrained objective function Eq. (4) is analyzed with penalty coefficient $\beta = 1$. In the CBQ model, we assume that the packet classifier assigns the same share of arriving packets to each queue and let 20 packets arrive per time unit in each queue on average. The overall bandwidth of the network link is set to a transfer rate of 100 packets per time unit. Let $D^* = (0.025, 0.25, 1)$ be a vector with predefined delay constraints. For a consistency check, we calculate the minimal affordable delay for each class, e.g., for class 1, the minimal affordable delay is 0.01216, which is achieved for weights $\mathbf{w} = (1, 0, 0)$. By analogous calculations for classes 2 and 3, we check that the

model is able to comply with D^* at least for individual classes.

For any fixed \mathbf{w} , we obtain a CTMC with 1,474,704 states and 10,151,988 non-zero entries. Different settings of \mathbf{w} do not change the dimension or non-zero structure of \mathbf{Q} ; only individual non-zero entries depend on \mathbf{w} . All rates are in a range of $[3.64, 10^4]$. The large differences in rates result from control tokens that circulate rather quickly relative to the service delays at a queue. We focus on iterations of the numerical solver as a hardware-independent measure. It took about 10 minutes to compute 1000 iterations not taking into account the transformation of the model into a CTMC and the evaluation of the results.

The numerical solution shows a remarkable behavior with respect to convergence of residuals, rewards, and response function f . Figs. 5 and 6 show a sequence of values for residuals, throughputs, queue lengths, and response function over the number of iterations performed by the numerical solver. Individual measures are taken every 100 iteration steps of an SOR solver, with a relaxation of 0.95 that starts with a uniform distribution for π_0 . In this experiment, we consider fixed weights $\mathbf{w} = (0.715, 0.207, 0.078)$. We observe a very slow rate of convergence for residuals (maximum and sum); after 6000 steps, the maximum residual is $d_2 < 10^{-7}$. In practice, solutions of $d_2 < 10^{-8}$ to $d_2 < 10^{-12}$ would be considered reasonably accurate. The values of throughputs and queue lengths

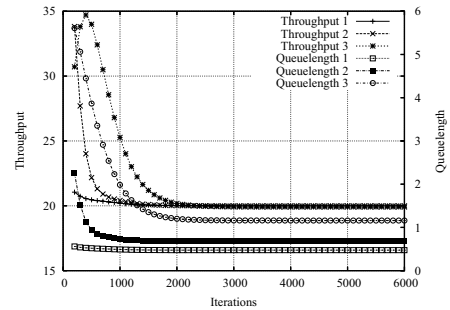


Fig. 5. Convergence behavior of SOR solver: Throughput and queue length (scenario 1)

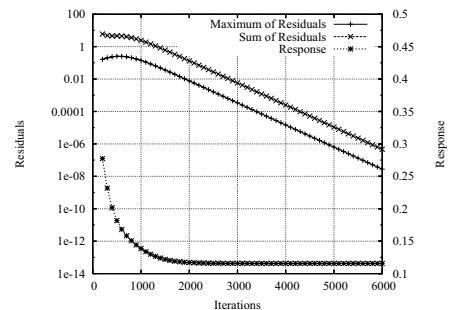


Fig. 6. Convergence behavior of SOR solver: Residuals and response (scenario 1)

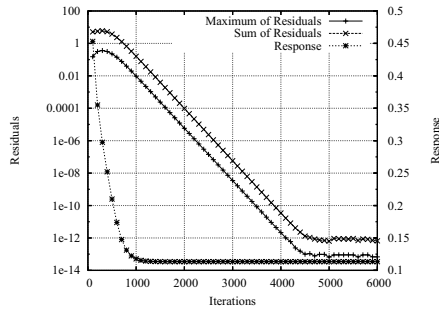


Fig. 7. Convergence behavior of SOR solver: Residuals and response (scenario 2)

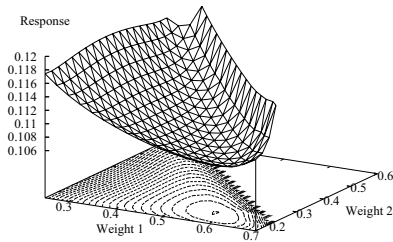


Fig. 8. Response surface for the class-based queueing system

vary significantly within the first 1500 steps and stabilize for more iteration steps, so convergence is quicker than for residuals. The same holds true for the curve of response function f in Fig. 6. We exercised different values for relaxation and observed an increased convergence rate for increasing values of the relaxation factor up to 1.15; however, at 1.2, we observed divergence. Hence, for the experiments described in this section, we decided on a defensive selection of the relaxation factor as 0.95. Since the CTMC varies according to values of \mathbf{w} , curves of the kind shown in Figs. 5 and 6 are expected to vary as well. Fig. 7 gives analogous curves for a CTMC with $\mathbf{w} = (0.33, 0.33, 0.34)$. Convergence for residuals is better; we achieve $d_2 < 10^{-5}$ in 2000 iterations, and $d_2 < 10^{-8}$ in 3000 iterations. Both examples confirm our assumption that an approximate evaluation of f based on numerical solutions of low accuracy can give reasonably good approximations in significantly less computation time.

We computed a grid over the response surface with precisions $d_2 < 10^{-4}$ and $d_2 < 10^{-8}$ and found that the numerical differences between the response values are insignificant, i.e., the maximum difference was below 0.004. The entire response surface for which we do not provide a diagram for lack of space, has very steep areas near the edges and a very flat area around the optimum; the best configuration we could identify is at $\mathbf{w} = (0.5774, 0.2142, 0.2084)$. Fig. 8 presents a region of the response surface that contains the optimum.

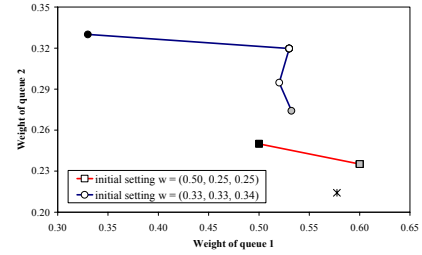


Fig. 9. Courses of RSM runs with $\epsilon = 10^{-2}$

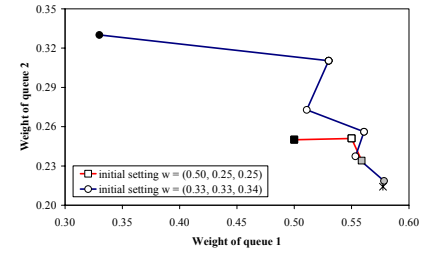


Fig. 10. Courses of RSM runs with $\epsilon = 10^{-3}$

Note that $w_3 = 1 - w_1 - w_2$ and w_3 is thus implicitly represented in the figure as well. Clearly, the computation of a grid over the entire surface is only used for illustration and validation purposes. The RSM algorithm only evaluates relatively few spots on the response surface.

Two of the input parameters of the RSM algorithm are the precision of the numerical solution, for instance, measured by the maximum residual d_2 , and the initial values of \mathbf{w} , whose impact we evaluate for the given model. We conduct experiments for $d_2 < \epsilon$ with $\epsilon = 10^{-i}$, $i=2, \dots, 6$, and two initial settings for \mathbf{w} , namely $\mathbf{w}_1 = (0.33, 0.33, 0.34)$ and $\mathbf{w}_2 = (0.5, 0.25, 0.25)$. Fig. 9 shows center points on a course of the RSM algorithm for \mathbf{w}_1 , \mathbf{w}_2 , and $d_2 < 10^{-2}$. A filled black symbol indicates the initial configuration, a filled gray symbol the final result, and an asterisk the optimal solution from the grid evaluation of Fig. 8. For RSM we considered stopping rule (ii) introduced in Section 2.2, i.e., RSM stops if the half-width of the local region becomes smaller than $\omega_{\text{stop}} = 0.01$. In Fig. 9, both computations get in the region of the optimum; however, the quality of the results suffers from lack of precision. Fig. 10 gives results for a precision of $d_2 < 10^{-3}$, where RSM gets close to the optimum. A similar behavior is observed for a better precision, i.e., $\epsilon = 10^{-4}$ and $\epsilon = 10^{-6}$. We use the results as an indication that the RSM algorithm is robust with respect to the selection of an initial configuration. In the following, we study the three different strategies for RSM introduced in Section 3.

In Fig. 11, we perform RSM with different levels of precision and consider the quality of the results whose values are given in Tab. 1. RSM(i) denotes an RSM

run with precision $d_2 < 10^{-1}$. The column “Weights” gives the final value of \mathbf{w} of the RSM algorithm, and the column “Response” gives the exact value $f(\mathbf{w})$, i.e., computed from an additional numerical solution with $d_2 < 10^{-10}$. The results show that RSM with a precision higher than 10^{-4} does not yield better results. It is not worthwhile. A precision of 10^{-2} gives results whose response function differs from the response for precision 10^{-4} by less than 1%. We focus on $f(\mathbf{w})$ rather than the Euclidian distance of \mathbf{w} from the optimal solution; the best value of f that we obtained is $f(0.5774, 0.2142, 0.2084) = 0.106959$. Recall that the SOR precision that is required in order to compute the response function with sufficient accuracy can also be determined by the algorithm presented in Fig. 3 of Section 3. In fact, running that algorithm for the five design points of the experimental design that RSM uses for the first local region yields a required precision of $\varepsilon_0 = 10^{-4.1}$ when the computed precisions for the design points are averaged. Thus, strategy 2 can safely be applied in this case. The following experimental results indicate that for this example, RSM succeeds even for a lower precision. However, we believe that for strategy 2 we chose constants that will perform reasonably well for a range of models. We do not want to calibrate the algorithm in Fig. 3 too much to our specific example.

To reveal the impact of strategy 1, we experimented with two cases: a) a uniform initial distribution and b) an initial distribution according to strategy 1, i.e., the initial distribution is determined from a previously computed distribution $\boldsymbol{\pi}$ of a configuration that is close to \mathbf{w} . For strategy 3, note that evaluation of all points of a single first- or second-order model should take place with the same precision, since Figs. 6 and 7 indicate that results obtained at different levels of

Tab. 1. Settings of the weights found by RSM

	Weights	Response
RSM(2)	(0.600, 0.235, 0.165)	0.107293
RSM(3)	(0.559, 0.234, 0.207)	0.107010
RSM(4)	(0.577, 0.214, 0.209)	0.106959
RSM(5)	(0.577, 0.213, 0.210)	0.106959

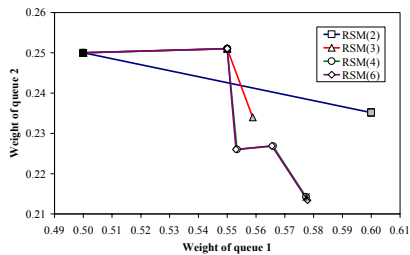


Fig. 11. Courses of RSM runs with same starting point but different SOR precisions

Tab. 2. RSM runs with different strategies

	$\boldsymbol{\pi}_0$	ε_0	ε_1	#Iterations	Response
RSM(2)	uniform	10^{-2}		39100	0.107293
RSM(3)	uniform	10^{-3}		44000	0.107010
RSM(4)	uniform	10^{-4}		87600	0.106959
RSM(2,3)	previous	10^{-2}	10^{-3}	14750	0.113713
RSM(2,4)	previous	10^{-2}	10^{-4}	31500	0.106993
RSM(3,4)	previous	10^{-3}	10^{-4}	32900	0.106994

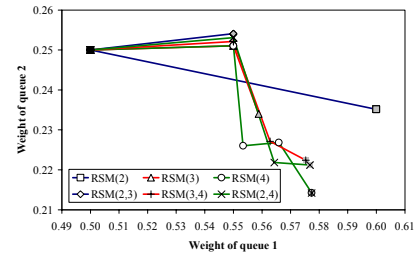


Fig. 12. Effect of different strategies

precision differ significantly in the steep initial part of the curves of response function f .

In Fig. 12, we consider the effect of using strategies 1 and 3 for the RSM algorithm in combination. RSM(i,j) denotes an RSM run that starts with precision 10^{-i} and switches over to precision 10^{-j} if the half-width ω of the local region that RSM considers decreases to below 0.06 (see strategy 3 in Section 3). As in the previous figures, a filled gray symbol denotes a final result. Tab. 2 gives detailed information on the applied variant of RSM, i.e., it shows whether the selected initial distribution was uniform or a previously computed one (column $\boldsymbol{\pi}_0$), it shows the selected precision (column ε_0), and it shows the observed total number of iterations of the numerical solver (column #Iterations) that are summed over all evaluations of design points. If there is an entry in column ε_1 , it indicates that the algorithm initially starts with the precision given in column ε_0 and then switches to a numerical solution with higher precision ε_1 . Note that differences in Tab. 2 result not only from different precisions but also from the fact that RSM performs a different total number of steps and visits different design points. RSM(2,4) and RSM(3,4) yield results of similar quality, but RSM(2,4) saves about 1400 iterations. RSM(2,3) saves a lot on iterations because of two effects. First, low precision reduces the number of iterations per design point, but more importantly, the RSM algorithm terminates after a few steps being misled by the imprecise values of the numerical solution. For low precisions, we do not achieve a better result with RSM(i,j) than by using RSM with no strategy. However, that means only that a change for the maximum residuals from 10^{-2} to 10^{-3} is not enough to find an appropriate maximum. If we compare results of RSM(3,4) and RSM(2,4), we gain a significantly

better result than for the fixed precision variants RSM(2) and RSM(3) and only a slightly worse result than RSM(4). Considering the computational effort, the savings achieved by strategies 1 and 3 are significant; for instance, RSM(2,4) reduces the number of necessary iterations from 87600 to 31500. Note that we get even significantly better results with RSM(2,4) or RSM(3,4) that need about a fourth less iterations than RSM(3).

In [3], a more simple variant of the CBQ model with a Poisson arrival stream is considered with a similar objective function. In [3], the minimal sum of delays for a comparable configuration of the model is found at $\mathbf{w} = (0.7157, 0.02768, 0.09141)$ with delays $D = (0.02367, 0.03373, 0.05860)$. In our CBQ model, that configuration has a response of 0.11601. Running the RSM algorithm with a penalty coefficient $\beta = 1$, the best solution found is $\mathbf{w} = (0.57737, 0.21425, 0.20838)$ with corresponding delays $D = (0.02614, 0.03958, 0.0401)$ and a response of 0.10696. In this solution, the delay constraint for the first traffic class is slightly violated. Increasing the impact of the penalty coefficient to $\beta = 5$, the best solution found is $\mathbf{w} = (0.64438, 0.18170, 0.17392)$ with corresponding delays $D = (0.02497, 0.04073, 0.04176)$. With $\beta = 5$, the constraints are fulfilled, and the sum of delays is lower than in the configuration found in [3]. Thus, we can conclude that the presented RSM algorithm for optimizing numerical models is a quite versatile approach that outperforms a previous approach in terms of quality of the solution.

6. Conclusions

We presented an approach for the optimization of stochastic models that is based on the response surface methodology. In contrast to previous work, our approach works in a fully automated way and is tailored to the optimization of computationally expensive numerical models. During the optimization process, we iteratively use first- and second-order linear regression metamodels combined with a gradient-based method to find a direction of improvement.

The numerical analysis is based on a Gauss-Seidel solver with relaxation that uses a hierarchical Kronecker representation for a given continuous time Markov chain. Since RSM implies a repeated solution of related CTMCs, we propose three strategies to reduce the amount of computation time per evaluation of the response function. The overall approach is evaluated with the help of an application example of a class-based queueing system. We determined a

configuration of the queueing weights such that the sum of delays is minimized under given delay constraints.

References

- [1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Francheschinis, *Modelling with Generalized Stochastic Petri Nets*, John Wiley & Sons, 1995.
- [2] F. Bause, P. Buchholz, and P. Kemper, A Toolbox for Functional and Quantitative Analysis of DEDS, *Proc. Tools'98, LNCS 1469*, 356-359, Springer, 1998. www4.cs.uni-dortmund.de/APNN-TOOLBOX/
- [3] P. Buchholz and A. Panchenko, Numerical Analysis and Optimisation of Class Based Queueing, *Proc. 16th Europ. Simulation Multiconference*, 543-547, SCS Publishing, 2002.
- [4] P. Buchholz and P. Kemper, Kronecker Based Matrix Representations for Large Markov Models, *Validation of Stochastic Systems, LNCS 2925*, 256-295, Springer, 2004.
- [5] P. Buchholz, M. Telek, and A. Thümmler, A Novel Approach for Fitting Probability Distributions to Real Trace Data with the EM Algorithm, *Proc. Int. Conf. of Dependable Systems and Networks (DSN), Yokohama, Japan, 2005*.
- [6] D.D. Deavours, et al., The Möbius Framework and Its Implementation, *IEEE TSE 28*, 956-969, 2002.
- [7] S. Floyd and V. Jacobson, Link-sharing and Resource Management Models for Packet Networks, *IEEE/ACM Transactions on Networking 3*, 365-386, 1995.
- [8] Int. Traffic Archive <ita.ee.lbl.gov/index.html>
- [9] V. Jacobson, K. Nichols, and L. Zhang, A Two-bit Differentiated Service Architecture for the Internet, *Request for Comments 2638, Internet Engineering Task Force*, 1999.
- [10] S. Kirkpatrick, C. D. Gelatt, and M.P. Vecchi, Optimization by Simulated Annealing, *Science 220*, 1983.
- [11] J.P.C. Kleijnen and R.G. Sargent, A Methodology for Fitting and Validating Metamodels in Simulation, *European Journal of Operational Research 120*, 14-29, 2000.
- [12] A.M. Law and W.D. Kelton, *Simulation Modeling and Analysis*, 3rd Edition, McGraw-Hill, 2000.
- [13] A. Michalas, et al., Proportional Delay Differentiation Provision by Bandwidth Adaptation of Class-based Queue Scheduling, *Int. J of Communication Systems 17*, 2004.
- [14] A. Miner and D. Parker, Symbolic Representations and Analysis of Large Probabilistic Systems, *Validation of Stochastic Systems, LNCS 2925*, 296-338, Springer, 2004.
- [15] D.C. Montgomery and R.H. Myers, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 2nd Edition, John Wiley, 2002.
- [16] H.G. Neddermeijer, et al, A Framework for Response Surface Methodology for Simulation Optimization, *Proc. Winter Simulation Conference, USA*, 129-136, 2000.
- [17] H.P. Schwefel, *Evolution and Optimum Seeking*, John Wiley, 1995.
- [18] W.J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, 1994.