Optimization of Markov Models with Evolutionary Strategies Based on Exact and Approximate Analysis Techniques *

Peter Buchholz
Informatik IV
Universität Dortmund
D-44221 Dortmund, Germany
peter.buchholz@udo.edu

Peter Kemper
Department of Computer Science
College of William and Mary
Williamsburg, Virginia 23187-8795, USA
kemper@cs.wm.edu

Abstract

Markov models are useful in the performance and dependability assessment of systems to obtain quantitative information that helps in making design decisions. The many known analysis techniques can be partitioned into approximate and exact techniques, where the former can be usually applied with limited effort but unknown precision and the latter give exact results but for the price of a computationally expensive calculation. In this paper, we discuss how an optimization method that is used to find an optimal configuration in a design space can make good use of both approximate and exact techniques for Markovian models. We develop a general approach that is formulated for evolutionary strategies and evaluated with Markov models of two queueing systems, a polling server model with real-valued design parameters and a finite buffer queueing network with discrete parameters.

1 Introduction

A model-based design of systems frequently provides some degrees of freedom that must be carefully used to obtain a satisfying design. This may yield to optimization problems of two kinds, namely static or dynamic ones. A static optimization problem is to find parameter settings that give optimal results for a parameterized model; a more dynamic optimization problem is to make optimal decisions for the dynamic behavior of a model at runtime, e.g., during a simulation run. In this paper, we focus on the former class of problems.

Optimization is a classic research area in its own right, with a long tradition and an ample variety of techniques for different types of optimization problems. Problems can be classified by the types of their parameters, which distinguishes discrete, continuous, and mixed problems, by the number of peaks of the objective function, which distinguishes uni- and multimodal problems, and also by the number of criteria that shall be maximized or minimized, which distinguishes single and multi criteria problems.

In the context of performance and dependability modeling, we observe continuous parameters, e.g., arrival rates, failure rates, speed factors, probabilities, and discrete parameters, e.g., number of resources, number of redundant spare components. Designs for performance or dependability often analyze tradeoffs such that the shape of an objective function will often be unimodal although this not guaranteed in general. Tradeoffs also imply that there are criteria that are in conflict, e.g., the classical situation between waiting time and utilization, which implies that multi-criterial optimization would be natural choice. However, often it is possible to encode multiple criteria and constraints into a single function that evaluates to a single result value for optimization. Mathematically this is often formulated by a weighted summation. The same concept is commonly used to encode constraints such that a violation of a constraint implies a negative effect, a penalty on the objective function.

In summary, we expect both discrete and continuous parameters, sometimes unimodal but often multimodal objective functions, and restrict ourselves to the single criterion case. However, evaluation of performance and dependability models comes with some further challenges.

a) The evaluation of the objective function at a design point may be computationally costly if a result of high precision is required, e.g., in case of a discrete event simulation with estimators of reward values with small confidence intervals or in case of a numerical analysis for the steady state distribution of a CTMC, phenomena like stiffness and rare events relate to models that are particular cumbersome to evaluate.

*This material is based upon work that is partially supported by Deutsche Forschungsgemeinschaft, SFB 559.
b) The evaluation may be performed quickly but only approximately. The approximation error is usually unknown, as it is the case for most approximate queueing network analysis methods or approximate numerical techniques for steady state analysis of Markov chains. The result of a simulation is always approximate, its quality is measured by confidence intervals only in a statistical manner.

So there is an obvious trade-off between computation time and precision of results for the evaluation of a single design point. On the other hand, optimization methods differ in how they identify a global optimum. Some methods require gradient information and follow a steepest ascent. Some methods like RSM approximate a gradient by a first or second order model and search in the direction of steepest ascent. Some methods like evolutionary strategies (ES) and genetic algorithms perform a randomized pointwise evaluation of the design space and only require a reasonably reliable comparison among the results of evaluations at different design points.

In this paper, we investigate how evolutionary strategies can make use of both, approximate and exact analysis techniques for the analysis of Markov models. The rest of the paper is structured as follows. In Section 2, we briefly recall Markov chains and their analysis. Section 3 is devoted to ES and we discuss the combination of Markov chain analysis with ES in Section 4. The whole approach is implemented and we evaluate a polling server model and a finite buffer queueing network in Section 5 as application examples to illustrate properties of the proposed technique.

2 Markov Models

Modeling with Markov models usually takes place by describing a model in a high-level formalism like generalized stochastic Petri nets (GSPNs), Markovian process algebras, or stochastic automata networks. Those modeling formalisms are defined in a way, that there is an established procedure to obtain its associated Markov chain. In this paper, we focus on continuous time Markov chains (CTMCs). A CTMC is a state-transition system with a set of states \( S \) where a transition from a state \( s \) to a state \( s' \) takes place with a rate \( r(s, s') \), which is the rate of a negative-exponential distribution. For a finite set \( S \), a CTMC is represented by its generator matrix \( Q \), a real-valued \( S \times S \) matrix, where \( Q(s, s') = r(s, s') \) for \( s \neq s' \) and diagonal entries are defined as \( Q(s, s) = \sum_{s' \in S, s' \neq s} Q(s, s') \). Given an initial probability distribution \( \pi_0 \), the transient distribution \( \pi_t \) at time \( t > 0 \) is given by \( \pi_t = \pi_0 e^{Qt} \). For \( t \rightarrow \infty \), we can compute the steady state solution \( \pi \) from \( \pi Q = 0 \). Both distributions, \( \pi \) and \( \pi_0 \), are the basis to evaluate highly aggregated measures (called rewards) that have an interpretation at the level of the modeling formalism, e.g., in a GSPN, the mean throughput of a transition or the mean number of tokens at a certain place.

Many models imply very large dimensions of \( Q \) (the infamous state-space-explosion) which has stimulated a lot of research to be able to solve the equations for transient and steady state analysis if \( S \) is in the order of \( 10^6 \) and above. Prominent examples are symbolic representations like multi-terminal binary (multi-valued) decision diagrams (MTBDDs, MTMDDs) and matrix diagrams (MxDs), and Kronecker representations, see [15, 4] for recent surveys. A Kronecker representation is based on a decomposition of a model into a set of submodels \( T, |T| = N \), with corresponding matrices \( Q_i \) for submodel \( i \in T \) and activity \( t \in T \), where \( T \) denotes the set of possible action labels. The overall matrix is represented by a sum of Kronecker products of smaller matrices.

\[
Q = \sum_{t \in T} \bigotimes_{i \in T} Q_i^t - D
\]

where \( D \) is a diagonal matrix to assure that row sums are 0. \( D \) can be represented in a compact symbolic form or by a vector of length \( |S| \), [4]. A Kronecker representation can be used as a building block for a block structured matrix representation of \( Q \) where each block \( Q[k, l] \) is represented by a sum of Kronecker products.

\[
Q[k, l] = \sum_{t \in T} \bigotimes_{i \in T} Q^t_i[k, l] - \delta(k = l)D_k
\]

where again \( D_k \) is a diagonal matrix to represent the diagonal elements of \( Q \). The resulting structure is called hierarchical Kronecker representation and automatic procedures exist that generate it from a given Kronecker representation. A hierarchical Kronecker representation has certain advantages, for instance, it contains only reachable states and it supports advanced solution methods like multi-level methods [3].

Exact analysis Although, only a system of linear equations needs to be solved for steady state analysis, numerical stability and the fill-in effect that destroys sparsity and regularities of \( Q \) limit the applicability of direct methods like Gaussian elimination and LU decomposition to CTMCs with very small state spaces. There is a large variety of iterative methods that are successfully employed instead. Simple methods are the power method, the Jacobi, and the Gauss-Seidel method. Projection methods based on a Krylov subspaces require more iteration vectors for the potential of a faster convergence. Iterative Aggregation/Disaggregation and multi-level techniques iterate on a set of equation systems at different levels of aggregation. All iterative methods start with some given initial distribution and terminate when a fixed-point is reached with a final solution \( \pi \). In practice, convergence is identified by a measure that falls below a threshold \( \epsilon \) close to zero, e.g.,
the maximal difference between iteration vectors, the maximum or sum of residuals for $\pi, Q = 0$ and $\pi_i$ being the current iteration vector.

For transient analysis, the solution of $\pi_i = \pi_0 e^{Qt}$ is usually obtained by randomization (uniformization, Jensen’s method), which is also an iterative procedure based on matrix-vector multiplications involving $Q$. Formally, it follows

$$\pi_i = \pi_0 \sum_{i=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^i}{i!} P^i$$

where $P = I + Q/\lambda$ with identity matrix $I$ and $\lambda \geq \max_i(|Q(i,i)|)$. Although randomization truncates an infinite summation, the truncation error can be controlled, such that we consider the method as exact.

The bottlenecks of exact analysis techniques are space and time. Symbolic and Kronecker representations move the space limitation from the representation of $Q$ to the representation of iteration vectors, which are of length $S$. However, that limitation is still severe. Iteration methods differ in the number of vectors that are required, SOR with a single vector is the most efficient in terms of space. Krylov subspace methods are very demanding to achieve a good rate of convergence. In any case, computation times to obtain solutions for large state spaces and small $\epsilon$ are usually significant.

**Approximate analysis** The motivation for approximate analysis techniques is to obtain results rather quickly and for a much larger class of models. A number of such techniques have been developed which we briefly recall in this section.

For steady state analysis, so-called fixed-point iteration approaches based on a decomposition of a model into submodels are known. Those submodels are solved individually and in isolation. Computed results of one submodel are used as input parameter for others, such that one iterates over the set of submodels until results, resp. parameter settings stabilize. Those approaches have no complete description of the overall CTMC. Given that a symbolic or Kronecker representation is so space efficient, there are some recent approaches that work with a complete description of the CTMC and approximate $\pi$. In [14], Ciardo et al describe an iterative solution method based on aggregated vectors. In [2], Buchholz proposes a representation of iteration vectors by a sum of Kronecker products of smaller vectors, which yields an approximation of $\pi$.

For transient analysis, the variety of approaches is more limited. Approximations are based on the aggregation of unimportant states [1], or apply to a particular class of models of a specific structure [6]. Approximation of $\pi$ by a sum of Kronecker products can be applied as well. Buchholz and Sanders describe appropriate enhancements in [5]. In addition, the finite time range motivates to truncate the dynamic behavior of a model after a certain number of state changes, which yields to path-based approaches that perform a path-truncation, see [12] for details.

### 3 Evolutionary Strategies

We consider the optimization (minimization/maximization) of some function $f : \times_{i=1}^n [l_i, u_i] \rightarrow \mathbb{R}$ with $l_i < u_i$. Usually we assume real valued parameters, but the approach may as well be applied for integer parameters as outlined below. In the cases considered here, the evaluation of $f$ requires that one computes the stationary or transient analysis of a CTMC where dimension and entries of $Q$ depend on values $x_i \in [l_i, u_i]$. Furthermore, we assume that another function $g : \times_{i=1}^n [l_i, u_i] \rightarrow \mathbb{R}$ exists which approximates $f$ and is much more efficient to evaluate. In our context, $g$ is an approximate analysis technique for CTMC analysis. As it is often the case, we assume that the difference $f(x) - g(x)$ for $x = (x_1, \ldots, x_n)$ and $l_i \leq x_i \leq u_i$ is unknown. Since $f$ is not available in closed form, only very limited information is available, we can only derive results by evaluation at certain points in the parameter space and derivatives are unknown. The structure of $f$ is in general unknown, but for most practical problems it is often smooth on wide parts of the parameter space, but it is often multimodal and might have 5 to 10 parameters.

Most commonly used optimization methods cannot be applied for the optimization of $f$ due to the lack of structural information. A class of methods that can be applied to this kind of models are evolutionary algorithms (EA) which are metaheuristics and became very popular in the recent decade. Among the different classes of EA we consider here evolution strategies (ES) but our approach could as well be combined with other variants like genetic algorithms in a straightforward manner. We first give a brief overview of ES; for further details about ES and other EA, we refer to the literature [16]. Afterwards, we identify specific requirements for the optimization of CTMCs and we propose a refined ES which exploits the availability of two functions, the exact function $f$ and the approximation $g$.

ES are population based and inspired by the evolution of species in nature. We consider a sequence $l = 1, 2, \ldots$ of generations with a population $\mathcal{P}$, $|\mathcal{P}| = \lambda^l$. An individual $p \in \mathcal{P}$ is characterized by a tuple of two vectors $(x, s)$ of size $n$. Vector $x$ with $l_i \leq x_i \leq u_i$ describes the parameter values of the individual and vector $s$ with $s_i > 0$ is used for strategy parameters as described below. We number individuals in the population consecutively and denote by $p^{(l)} = (x^{(l)}, s^{(l)})$ the $l$th individual ($1 \leq l \leq \lambda$). The

\footnote{The size of the population is usually denoted as $\lambda$ in ES. In conjunction with CTMCs, this might be misleading since $\lambda$ is there used for arrival rates. However, we decided to use $\lambda$ in this paper for the population size. Similarly, we use $\mu$ for the number of parents as common in ES.}
Evolution strategies have been applied to a large number of problems. However, the majority of problems are characterized by a high dimensional parameter space and a function $f$ that is computationally inexpensive to evaluate. This implies that many function evaluations can be made for optimization. However, if the evaluation is costly, then the situation changes. Some examples exist where ES has been applied to continuous simulation models which are costly to evaluate [16]. To the best of our knowledge ES has only rarely been applied for the analysis of discrete event systems and we are not aware of any other approach using it in combination with the numerical analysis of CTMCs. The practical problem is that the number of function evaluation can be enormous such that the approach becomes infeasible, if a single function evaluation takes too long.

If that happens to be the case for $f$, the question arises to what extent we can substitute evaluations of $f$ by computationally inexpensive evaluations of $g$ without reducing the ability of ES to identify an optimum. Combinations of approximate and exact analysis techniques have already been investigated for evolutionary algorithms [11, 8], but those

---

**Figure 1. Scheme of a $(\lambda, \mu)$ ES**

ES Scheme

1. create initial population $P$ with $|P| = \lambda$
2. evaluate population $P$
3. while(termination condition = false)
   4. select the $\mu$ best individuals from $P$ as parents
   5. for ($l = 1$ to $\lambda$)
      6. if ($\text{ran}(f) < p_{\text{recomb}}$)
         7. select two parents for recombination
         8. recombine the selected parents to yield $(x^{(l)}, s^{(l)})$
      9. else
         10. select one parent for mutation
         11. mutate the selected parent to yield $(x^{(l)}, s^{(l)})$
      12. add $(x^{(l)}, s^{(l)})$ to $P'$
   13. evaluate individuals in $P'$ and set $P = P'$, $P' = \emptyset$
14. return best value of all generations

Fitness of individual $l$ is given by $f(x^{(l)})$. The basic idea of ES is to choose $\mu$ ($< \lambda$) individuals with the highest fitness from $P$ and use them as parents for the next generation. A commonly used guideline is to choose $\mu/\lambda \approx 1/7$. From the parents new offsprings are generated by applying some perturbation operations denoted as mutation or sometimes also recombination. The operations are described in more detail below. Two variants of ES exist; the difference is whether elements of a parent generation can be selected for the next generation or not. The former is denoted as plus selection ($\lambda + \mu$), the latter as comma selection $(\lambda, \mu)$. A high-level pseudo description of the comma selection ES scheme is given in Fig. 1. The algorithm for plus selection is similar.

We now describe the different steps of the algorithm in some more detail. In a first step an initial population is generated randomly. Thus, $\lambda$ individuals are generated by setting $x_i = \text{ran}(f) \cdot (u_i - l_i)$ where $\text{ran}(f)$ generates a $[0, 1]$ uniformly distributed random variable. The values $s_i$ are usually set to fixed values, e.g., $s_i = (u_i - l_i)/3$. In the second step, the fitness of the individuals is evaluated which usually means that $f$ is evaluated. We come back to this point below. Then the main loop of the algorithm is iterated until the termination criterion is met. As termination criterion we may use a maximal number of generations or the convergence of the algorithm which is assumed if the fitness of the best individual in the population or the average fitness of the population does not improve for several generations. Selection of the best individuals is done by ordering the individuals according to their fitness. In the loop from line 5 through 12 new individuals are generated. Originally only mutation was used in ES for this purpose. However, contemporary variants also use recombination [16]. The recombination probability is set to a small value, e.g., $p_{\text{recomb}} = 0.05$. For recombination, we use a simple crossover that can also be applied for integer parameters. Recombination means that for each parameter and strategy value the value of one parent is chosen randomly. With simple interval bounds on the parameters as introduced above, the resulting child observes $l_i \leq x_i \leq u_i$. For more complex restrictions like $x_i + x_j \leq u_{ij}$, a recombination may yield a non valid individual. In that case, the procedure is repeated until a valid individual has been produced. The major step to generate new individuals is the mutation. Let $(y, r)$ be the parent individual and $(x, s)$ the offspring. First, let $\sigma_1 = N(0, 1)/\sqrt{\pi}$ where $N(0, 1)$ is drawn from a normal distribution with mean 0 and standard deviation 1. Then $x_i = y_i + N(0, r_i)$ and $s_i = r_i \cdot \exp(\sigma_1 + N(0, 1)/\sqrt{2 \cdot \sqrt{\pi}})$. The parameters of the new individual are given by the parameters of the parent plus a normally distributed perturbation with mean 0 and a standard deviation given by the strategy parameter. If one of the parameters $x_i$ falls out of the interval $[l_i, u_i]$, the procedure is repeated until the value falls into the interval. The strategy parameter is also modified using a random perturbation. Observe that this perturbation will result in positive strategy parameter as required. The use and modification of strategy parameters has been developed in the EA community over many years and with a lot of empirical analysis. The approach applies also to integer parameters. Assume that $x_i$ is an integer parameter with $x_i \in \{l_i, \ldots, u_i\}$. The recombination will also yield an integer value, if the parents have integer parameters. If we apply mutation, then the result is not necessarily integer. Assume that $z_i$ is the new value after mutation, if $l_i - 0.5 \leq z_i < u_i + 0.5$ we round $z_i$ to its nearest integer value to obtain the new parameter value, otherwise we repeat the mutation step.
approaches usually use some form of a metamodel like a polynomial function or a neural network [7] that is generated from some points that are evaluated exactly. Optimization methods that make use of a metamodel have also been applied with discrete event simulation [13]. However, those techniques lack robustness. They may work sometimes quite well but often they fail since the metamodel is only a rough approximation of the original model especially if the dimension of the problem is high. Our approach is different since we evaluate the original model but use an approximate evaluation approach. In the following section, we develop an approach that exploits the availability of \( g \) and \( f \) for a more efficient realization of ES.

4 Advanced Strategies for ES and CTMC analysis

If an optimization method evaluates \( f(x) \) and \( g(x) \) at the same point \( x \), we can measure the approximation error \( e(x) = f(x) - g(x) \) at that point. Let \( \mathcal{E} \) denote the set of design points, where we know \( e(x) \). Usually \( e(x) \) depends on \( x \) and may differ significantly for different points in the parameter space. Based on \( \mathcal{E} \), let us define the maximal error we observed \( e_{\text{max}} = \max_{x \in \mathcal{E}} e(x) \), mean error \( e_{\text{mean}} = \frac{1}{|\mathcal{E}|} \sum_{x \in \mathcal{E}} e(x) \), variance \( \sigma^2_e = \frac{1}{|\mathcal{E}|} \sum_{x \in \mathcal{E}} (e(x) - e_{\text{mean}})^2 \), and standard deviation \( \sigma_e = \sqrt{\sigma^2_e} \). Note that we make use of common formulas for estimators for mean and variance of a random variable. Thus, for random sampling of points in the parameters space, we will get true estimates of the values over the complete parameter space (i.e. \( x_{\text{best}} = \{l_i, u_i\} \)). However, our selection of \( \mathcal{E} \) is driven by the ES as shown below and therefore not random, so the following procedure is a heuristic and not an unbiased estimator for the values on the complete parameter space. On the other hand, we are interested in \( e(x) \) for some \( x \) near the optimum and ES, hopefully, evaluates points around the optimum. We will come back to this point below.

4.1 Optimization based on approximate analysis with exact post selection

The first approach that we propose is to perform an ES optimization based on \( g() \) and store all considered design points \( x \), i.e., after \( m \) generations we obtain the set \( \mathcal{P} = \cup_{i=1}^{m} \mathcal{P}_i \) of points. Let \( \mathcal{P} \) be ordered by decreasing values of \( g() \), i.e., \( g(x^{(i)}) \geq g(x^{(j)}) \) for \( 1 \leq i < j \leq |\mathcal{P}| \). A first and basic approach would be to evaluate all individuals using \( g() \) rather than \( f() \) in the ES. This yields an efficient approach, but the results will be wrong if the optimum of \( g() \) is at a different point than the optimum of \( f() \). Observe that if the problem is not the approximation error, it is the location of the optimum. However, without any further exact analysis nothing can be said about the quality of the approach. Thus, we combine exact and approximate analysis to obtain information about the effect of the approximation error.

The goal is to identify the best result in \( \mathcal{P} \) with respect to \( f() \) but without evaluating those elements of \( \mathcal{P} \) that are (almost surely) inferior. The issue is to give a mathematically formulation of “almost surely inferior”. For that purpose, we propose the following heuristics for a posteriori selection. We start an exact evaluation on the most promising points \( x^{(1)}, x^{(2)}, \ldots \) of \( \mathcal{P} \) and in that way increase \( \mathcal{E} \) and obtain values for \( e_{\text{max}}, e_{\text{mean}}, \) and \( \sigma_e \). Let \( x^{\text{best}} \) denote that point of all those that have been evaluated exactly that achieves the best (maximum) value of \( f() \). We stop the sequence of exact evaluations if the distance between \( g(x^{(i)}) \) and \( f(x^{\text{best}}) \) exceeds a threshold \( \delta = e_{\text{mean}} + \alpha \beta \sigma_e \) with given constants \( \alpha \) and \( \beta \). Alternatively, one can use \( e_{\text{max}} \) instead of \( e_{\text{mean}} \) for a more conservative approach. However, the use of \( e_{\text{mean}} \) was sufficient in the examples we checked so far. The constant values \( \alpha > 0 \) and \( 0 \leq \beta \leq 1 \) are scaling factors that shall ensure that \( \delta \) is large enough in the beginning to ensure that good candidates are exactly evaluated and that in the long run the value of \( \delta \) decreases. In our examples, we choose \( \alpha = 2 \) and \( \beta = 0.99 \). Observe that it is hard to derive properties of the estimators \( e_{\text{mean}} \) and \( e_{\text{max}} \) theoretically because they do not result from independent sampling. However, the estimators are based on sampling of points in the region of interest and therefore give good estimates of the error in the region of interest. Note that the post selection approach is adaptive since the number of exact evaluations depends on the estimated approximation error and the differences among approximated fitness values. Figure 2 describes the postselection procedure in pseudo code.

4.2 Threshold analysis

The post selection presented above has the disadvantage that the search process of the ES is only based on approximate analysis and not on exact values. In extreme cases,
where the approximation is very bad and the location of the optimum of $g()$ and $f()$ differs significantly, the final population $P$ will not include the interesting values. To guide the ES to the optimum of $f()$, exact and approximate values have to be mixed during the search process. The basic decision is which individuals to select as parents for the next generation. Let $P_i$ be the $i$th population of the ES and assume that all members of the population have been evaluated according to $g()$ and are sorted in decreasing fitness order. Now consider some individual $x^{(k)} \in P_i$ with $k \leq \mu$ and some other individual $x^{(l)} \in P_i$ with $l > \mu$, then $x^{(l)}$ instead of $x^{(k)}$ should belong to the next parent generation, if $g(x^{(k)}) + \sigma e < g(x^{(l)}) + \sigma e$. Unfortunately, the value of $e()$ is only known after both values have been evaluated exactly. Thus, we use our estimates for a decision which individuals to evaluate exactly. It can be observed that the decision depends on the variance of $e()$ rather than on the absolute value because a large but identical value of $e()$ at both points will not alter the decision. We now develop a selection procedure that incorporates those observations.

We perform an $(\lambda, \mu)$ ES and use both exact and approximate analysis for the selection of $P'$ in step 4 of the algorithm in Fig. 1. An evaluation in step 13 takes place by approximate analysis, hence we evaluate $g()$. For step 4, we order elements of $P$ in decreasing order of $g()$ and selectively evaluate $f(x)$ for $x \in P$ and subsequently reorder $P$ based on $f(x)$ instead of $g(x)$ wherever available. An individual $x \in P$ needs no exact evaluation, if $g(x)$ is very small such that it is very unlikely that it is among the $\mu$ best individuals and it also does not need an exact evaluation, if $g(x)$ is very large such that it is very unlikely that $x$ does not belong to the $\mu$ best individuals of the population. In order to obtain a formal decision rule, we use the interval $g(x) \pm \gamma \cdot \sigma e$ and define $\gamma = 1$ for the examples considered below. We come back to the case with $x^{(k)} (k \leq \mu)$ and $x^{(l)} (l > \mu)$ considered before and we evaluate one of the two points exactly if $g(x^{(k)}) - \sigma e < g(x^{(l)}) + \sigma e$. If an exact value is already available for $x^{(k)}$ or $x^{(l)}$, then we evaluate the other one exactly if $f(x^{(k)}) - \sigma e < g(x^{(l)}) + \sigma e$ or $g(x^{(k)}) - \sigma e < f(x^{(l)}) - \sigma e$. Observe that the values of $\sigma e$ and $\sigma e$ are updated with each exact evaluation. The procedure to perform the selection using exact and approximate analysis is shown in Fig. 3. $i$ and $j$ are indices of elements in the ordered set $P$ and $\mu$ is the cardinality of the set we want obtain, i.e., we want to select the best $\mu$ elements out of $P$. Sets $E$ and $A = P \setminus E$ make use of $f()$ resp $g()$ for comparisons. Note the lines 5, 8, 11, and 14 handle separate cases that all result to the decision to evaluate $f(x^{(i)})$. The order is only used to distinguish a subset of $\mu$ best elements in $P$ by their index. The algorithm terminates if either all elements have been evaluated exactly or if we considered all elements of $A$ but no element remaining in $A$ required an evaluation. In the latter case, the situation is stable and we can stop. In the worst case, all elements are evaluated exactly if their approximate fitness values are similar and $\sigma e$ is large. In this way, we obtain an efficient approach where simple decisions are made on the basis of approximate results and critical borderline decisions are substantiated by exact results.

### 5 Examples

We consider two different examples. The first one is a polling system with up to 4 continuous variables and the second one is a queueing system with blocking with 8 discrete parameters.

#### 5.1 A Polling System

We consider a polling server system, where a single server visits four queues in a cyclic manner. Each queue has a workload that is generated by a finite population of processes, it is a single class queue with random scheduling and no preemption. A process performs a cyclic sequence of 3 activities, namely thinking, waiting for a server, getting service. The server performs an infinite loop moving from queue $i$ to $i + 1 \mod 4$, its activities are moving to the next queue, if a process is waiting, it starts serving one process,
once the service is completed and there is another process waiting for service at that queue, the server chooses with probability $p$ to stay and continue servicing processes and with probability $1 - p$ to move to the next queue. Figure 4 shows a generalized stochastic Petri net that models the behavior at queue $i$. Places $p_{1i}$, $p_{2i}$, and $p_{3i}$ correspond to states of processes, places $s_{i1}$, $s_{i2}$, $s_{i3}$, and $s_{i4}$ to states of a server at queue $i$.

Table 1 relates activities with Petri net transitions (Column ID) and gives rates of negative exponential distributions for timing information or probabilities in case of immediate transitions in Column weight. Note that each queue $i$ has a single parameter $x_i$, namely the weight of transition $t_{6i} = p$ that gives the probability for the server to remain at a queue and to start serving another customer.

The purpose of this modeling exercise is to maximize the mean throughput $E[T] = \sum_{i=1}^{4} E[T_i]$ over all queues relative to the mean number of waiting processes $E[N] = \sum_{i=1}^{4} E[N_i]$ over all queues in the long run, i.e. we consider the steady state, formally

$$\max f(x_1, x_2, x_3, x_4) = E[T]/E[N]$$

Note that $E[T_i]$ is measured by the throughput of transition $t_{15}$ and $E[N_i]$ by the mean number of tokens at place $p_{1i}$. The number of processes visiting each queue is set to 12 and there is only 1 server polling the 4 queues. For any given $x_i \in [0, 1]$ we obtain a CTMC with 219,700 states. Q is represented by a hierarchical Kronecker representation. For an exact solution, we apply a Multi-Level method (ML) with a Gauss-Seidel solver applied at each level [3] and experienced computations times of $70 - 120$ s wall clock time for various parameter settings of $x_1, x_2$. The method terminates if the residual $\epsilon$ is below $10^{-8}$. For an approximate solution, we apply a Power iteration that represents an iteration vector by a sum of Kronecker products of small vectors, [2]. The iteration terminates if the maximum difference between consecutive sets of vectors is below $10^{-8}$. Computation times for an approximate solution is in the range of 1.0 s wall clock time and thus about 2 orders of magnitude smaller than for the exact solution.
of the optimal points derived with the different approaches. It can be noticed that the points vary in the different solution runs, but a clear separation between the points determined using only approximate analysis (denoted as \textit{approx}) and the points for the remaining approaches. Results of the exact analysis are denoted by \textit{exact}. The approach with ES optimization based on \( q(f) \) with an exact post selection based on \( f() \) described in Section 4.1 is named \textit{app-exa}. The approach of Section 4.2 which adaptively makes use of approximate and exact analysis in the selection step and finally uses exact analysis for post selection is called \textit{hybrid}. Fig. 8 shows the estimated mean values for the optimum and the corresponding confidence intervals. Only \textit{approx} differs from the remaining strategies and has also the largest confidence intervals. The remaining three strategies show no significant differences. In Fig. 9 the number of exact evaluations is shown. Since the solution time of an exact and an approximate evaluation differ by two orders of magnitude, the number of exact evaluations determines the overall optimization time. The extreme cases are \textit{approx} with no exact evaluations and \textit{exact} where all points are evaluated exactly. The other two strategies are in between. The results indicate that \textit{hybrid} uses about half of the number of exact evaluations of \textit{exact} and \textit{app-exa} uses about 10\%. Consequently, for this example optimization with approximate evaluation and an exact post selection would be the best strategy.

In the second version of the example we change the rate of transitions \( t_{i1} \) to 0.1, 0.3, 0.2 and 0.4 for the queues 1 through 4, respectively. This destroys the symmetry of the problem and the weight of the transitions follows by \textit{app} function values. Again harder to optimize. Fig. 10 shows the estimates for the best function values. Again \textit{approx} yields the worst results followed by \textit{app-exa}. Between \textit{exact} and \textit{hybrid} no significant difference can be detected. Fig. 11 includes the number of steps required for the optimization. Again there is a significant difference between \textit{app-exa} and \textit{hybrid} but in this example \textit{hybrid} yields better results and is still faster than \textit{exact}.

5.2 A Queueing System with Blocking

Now we consider a tandem queueing network with four queues. Each queue is characterized by a buffer of size \( b_i \) and \( s_i \) servers \( (i = 1, \ldots, 4) \). Customers arrive according to a Poisson process with rate 3.0 to the first queue, as long as space is available. All service times are exponentially distributed with mean 1.0. If a buffer is full, the servers at the previous queue block and stop their service. Let

\[
C = E[T] - 0.2 \cdot \sum_{i=1}^{4} b_i - 0.5 \cdot \sum_{i=1}^{4} s_i
\]

be the cost function we want to consider where \( E[T] \) is the throughput of the system. We assume \( b_i \in \{1, \ldots, 10\} \), \( s_i \in \{1, \ldots, 6\} \) and \( b_i \geq s_i \). This model has 8 parameters which are all discrete such that the parameter space contains 4 100 625 different configurations. We apply the \((\mu, \lambda)\) ES with \( \mu = 5, \lambda = 35 \) and use rounding to generate integer parameters. For approximate analysis, each queue is one submodel for a Kronecker representation. For this example the size of the state space depends on the parameters and is not fixed. However, the approximate analysis is still significantly faster than exact analysis in almost all cases. It should be mentioned that the approximation may yield large errors for the example since the queues are coupled (i.e., the second and third queue have no internal transitions).

For the optimization only 735 configurations are evaluated which are less than 0.02\% of the possible configurations. It should be further mentioned that the model is not well suited for optimization by EA since it contains many local optima and usually only a simultaneous change (increase or decrease) of several parameters will improve the solution. Thus the probability of getting stuck in a local optimum is high. Fig. 12 shows the optimal values for the different approaches. Again the results for \textit{approx} do not reach the values of the other three strategies. Between the other three strategies no significant difference can be observed. Fig. 13 shows the number of exact evaluations. For this example, both strategies \textit{app-exa} and \textit{hybrid} use much less exact evaluations than in the previous example. Method \textit{app-exa} uses less than 5\% of the evaluations of \textit{exact} and \textit{hybrid} less than 20\%. Thus, even though the approximation is not as good as in the previous example, less function

![Figure 12. Best result values for the queueing network obtained in 20 generations](image-url)
evaluations are necessary since the differences between the fitness of elements in a population are larger too.

6 Conclusion

The paper presents a new approach to optimize parameters of Markov models according to some reward value using an evolutionary algorithm combined with exact and approximate solution techniques. It is shown that approximate and exact analysis can be integrated in the optimization algorithm and allow for a more efficient optimization without reducing the quality of the optimization approach. The central idea of the proposed technique is an estimation of the approximation error based on error values that have been observed at those points in the parameter space where both exact and approximate evaluations have been computed. With the help of those estimates, we can distinguish clear cases where an optimization method can proceed with approximate results from borderline cases where exact results are necessary.

The approach can be extended in several ways. It can be applied in combination with other solution techniques. The concept works whenever we have a fast approximate solver and a slow exact solver. If we accept long simulation runs as quasi exact results, we can apply the approach for example on the combination of simulation and product form queueing network analysis. In that case the approximation uses a more abstract model than the exact method. In addition, the approach can be combined with other black box optimization methods. In particular its integration in other metaheuristics like local search methods, pattern search or similar techniques [9] is immediate.

References