Reconstructing Model Parameters in Partially-Observable Discrete Stochastic Systems

Robert Buchholz, Claudia Krull, and Graham Horton

Otto-von-Guericke University Magdeburg,
PO Box 4120, 39016 Magdeburg, Germany
{robert,claudia,graham}@isg.cs.uni-magdeburg.de

Abstract. The analysis of partially-observable discrete stochastic systems reconstructs the unobserved behavior of real-world systems. An example for such a system is a production facility where indistinguishable items are produced by two machines in stochastically distributed time intervals and are then tested by a single quality tester. Here, the source of each defective item can be reconstructed later based solely on the time-stamped test protocol.

While existing algorithms can reconstruct various characteristics of the unobserved behavior, a fully specified discrete stochastic model needs to exist. So far, model parameters themselves cannot be reconstructed.

In this paper, we present two new approaches that enable the reconstruction of some unknown parameter values in the model specification, namely constant probabilities. Both approaches are shown to work correctly and with acceptable computational effort. They are a first step towards general model parameter inference for partially-observable discrete stochastic systems.

Keywords: partially-observable discrete stochastic system, Hidden non-Markovian Model, incomplete model, parameter inference.

1 Introduction

Many real-world systems can be regarded as partially-observable discrete stochastic (PODS) systems: their stochastic internal behavior can be modeled as a discrete stochastic model such as a non-Markovian stochastic Petri net [1,3]. And this internal behavior is of interest, but not directly observable; it can only be inferred through externally observable signals that are linked to the internal processes.

Analysis of these systems is the process of inferring the unobserved behavior of the real system using a time-stamped protocol of its signal emissions and the discrete stochastic model. It allows for additional insights into the inner workings of the real system.

A simple example of such a system is a small section of a production facility with two machines that produce indistinguishable items, and a quality tester to test items coming from both machines (see Figure 1). At the quality tester, the source of each item is no longer determinable (and thus unobserved), but the test protocol containing the time-stamped individual test results is a trace of the corresponding externally observable signals. Here, reconstructing the source of defective items can be of interest for purposes of quality assurance.

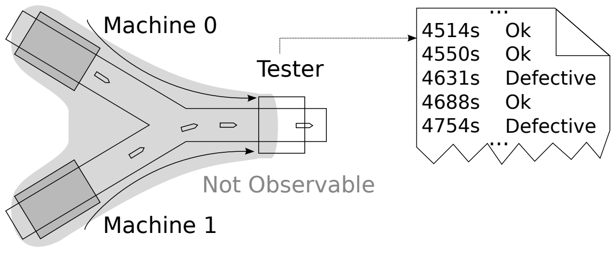

Fig. 1. Schematic representation of the partially-observable Tester system

So far, existing analysis algorithms only attempt to reconstruct the specific system behavior in time intervals for which signal protocols exist [7,11]. The model specification itself, which describes all *possible* behaviors has to exist beforehand. For the "Tester" system in Figure 1, one would like to determine the individual probabilities of the two machines to produce defective items, but these parameters are part of the model specification and therefore cannot be reconstructed using the existing approaches. This severely limits the applicability of the analysis of PODS systems, since it assumes that the unobservable system must have been be completely observable after all at some point in time, in order to derive the specification.

In this paper, we present two approaches that enable the reconstruction of some parameters of the model specification: We reconstruct the most likely values for unknown constant probabilities in incomplete model specifications with the help of a signal protocol. The first approach is an iterative algorithm that relies on an arbitrary initial estimate for unknown parameter values and iteratively improves that estimate. The second approach is based on computationally expensive symbolic computations in order to determine the most likely values directly.

The remaining paper is structured as follows: Section 2 classifies this work with respect to existing contributions. Section 3 introduces the current state-of-the-art in the analysis of PODS systems. Sections 4 and 5 describe the two new approaches. These are then evaluated in Section 6. The paper is concluded in Section 7.

2 Related Work

To our knowledge, the inference of model parameters for discrete stochastic systems from time-stamped stochastic signal emissions that represent partial observations of the real system has not been attempted before. However, similar problems have been solved before successfully.

Deriving the parameters of a linear dynamical system description from time series is a widely used tool in econometrics [2]. Here, a time series consists of vectors of continuous-valued quantities sampled at discrete time intervals, where each sample describes the complete system state. More recent advances in that area were made by Malyarenko et al. [14]. These analyses assume, however, that the observations are not the result of a completely stochastic process. Rather, they are assumed to be based on a combination of deterministic processes and interference from noise. The parameters to be determined are thus the coefficients for the influence of each component. Our focus on the other hand, is to derive parameters of a truly stochastic system with a discrete number of system states. Additionally, our samples are consequences of internal system state changes, which may occur at any real-valued point in time.

The inference of model parameters for truly stochastic discrete processes has been performed by many researchers: Hidden Markov Models are routinely used in speech recognition and other fields to learn all parameters of a hidden model from observed data alone [15]. However, in HMMs, all model parameters are constant probabilities and the models are discrete in time, limiting the expressiveness of the models. Thus, HMMs are not directly applicable to most real systems.

For some application fields, specialized analysis algorithms have been developed: Gibson et al. derived parameter values of a compartmental epidemiological model from partial observations of the number of specimens in some of the compartments [9]. Boys et al. [5] and Wang et al. [16] applied different algorithms to the problem of finding the rates at which chemical and biochemical reactions occur in stochastic settings where the number of chemical reactants is so low that the effects of randomness become important. In both cases, the observed data consisted of time series of the number of molecules for each type of reactant.

All of these approaches, however, impose the same two limitations: the stochastic model itself has to be Markovian, restricting the classes of discrete stochastic systems that can be modeled and analyzed. And the observations from the real system are sampled at discrete points in time. Thus, the approaches are only applicable to systems where (partial) observations of the system state can be made regularly "on demand" at regular intervals.

We, on the other hand, focus on discrete stochastic systems whose state changes can follow arbitrary probability distributions. And our approach relies only on "incidental" observations of signals that the real system emits on some of its state changes. Thus, our approach works even if the observations are made irregularly and if they are only stochastically linked to the system state changes (i.e. a signal can have multiple possible causes, and a state change can cause one of many different signals to be emitted). Consequently, our approach is computationally more expensive and thus cannot analyze models as big as the other approaches.

3 Background

In this section, we present background information relevant to the understanding of the new algorithms. We introduce Hidden non-Markovian Models (HnMMs) as a way of modelling PODS systems, and the Event-Driven Proxel algorithm as a means to analyze these models.

3.1 PODS Models

Hidden non-Markovian Models (HnMMs) were introduced as a paradigm to model arbitrary PODS systems [12]. They can be represented compactly by extending non-Markovian stochastic Petri nets [1,3]: In non-Markovian SPNs, the system is modeled through *places* (hollow circles) that contain a number of *markings* (small black circles), whose distribution represent the current system state. The system state can only change through the firing of *transitions* (hollow rectangles), which destroy markings in places leading to the transition through *arcs* (arrows) and create markings in places lead to by the transition. To represent HnMMs, this specification is extended by adding probabilities for the transitions to emit certain externally-observable symbols when they fire.

The representation of the Tester system from Figure 1 as an HnMM is given in Figure 2: The system has only a single state, so the only place always contains a single token. The two machines produce items in $N(150, 25)$ and $N(120, 20)$ distributed time intervals, respectively. Every production of an item emits a symbol “Ok” or “Defective” (as determined by the system’s quality tester). Machine 0 produces defectives with probability p_0, Machine 1 with p_1. In our application scenario, these two probabilities are unknown and thus the model specification is incomplete.

This model is slightly simplified since item production is the only possible event and each production event yields the item’s test predicate right away. This simplification in the model can be made when the time span between item production and testing is approximately constant. Then, each time of production can trivially be derived from the time of testing (by subtracting that constant time span from the test time stamps) and be recorded in the trace along with test results.

If those symbol emission probabilities were known, existing analysis algorithms could reconstruct the unobserved behavior for a system specified in this manner. The most efficient known algorithm for this task is detailed in the next section.

3.2 Analyzing PODS Systems

Several algorithms exist for the analysis of different classes of PODS systems. In this paper, we will only detail the analysis algorithm for the class E_{all} [12], to which the Tester model belongs. The algorithmic modifications presented in this paper to derive parts of the system specification can, however, be applied to all other classes of PODS systems and their analysis algorithms as well.

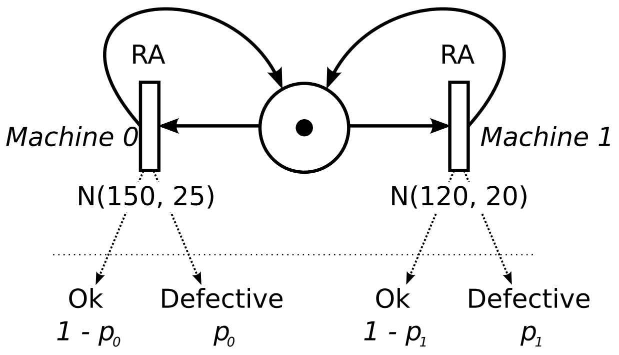

Fig. 2. The Tester system modeled as a Hidden non-Markovian Model

For models of class E_{all}, all internal state changes result in an externally observable symbol emission. In the HnMM representation, this means that all transitions emit symbols in all cases. Yet, since the type of the symbol is allowed to vary stochastically, and multiple transitions are allowed to emit the same symbol, the system state cannot be inferred unambiguously from the emitted symbol.

The consequence of this limitation is that all points in time where the system changes its state are known to the outside observer. Hence, it is certain that no event occurred between the emission of two symbols. This restriction allows E_{all} models to be analyzed using the efficient event-driven analysis algorithm described in the next section.

The Event-Driven Analysis Algorithm. The event-driven analysis algorithm for HnMMs [6] is derived from the state space-based Proxel simulation method [10,13]. It is a simplified version of the former algorithm, based on the observation that for E_{all} models no events occur in between two symbol emissions and thus possible system behaviors need only be studied at the times of symbol emissions.

The algorithm therefore divides the time domain into time steps corresponding to the time between two signal emissions. For each time step, it follows all possible single system state changes (i.e. those that can occur exactly at the end of the time step and are consistent with the trace) in parallel and computes the likelihood for each of them. The analysis result is then the set of all states that the system can be in after it has emitted the last signal recorded in the trace, along with the probability of being in each state.

In effect, the algorithm dynamically builds and solves the underlying inhomogeneous Markov chain for each time step, where the set of Markov chain states is the combination of the set of reachable system states from the HnMM with supplementary variables that store the age of non-Markovian transitions in each state.

For a given point in time, the possible system evolutions are stored as so-called Proxels, tuples of type (m, τ, p). Here, m is the marking of the system state in the HnMM's underlying Petri net. The vector τ contains the supplementary variables, the ages of all relevant transitions (i.e. the duration for which each of these transitions is already active); p is a measure of probability for the marking-age combination.

The analysis starts by creating Proxels for the known initial system state (or states) at $t = 0$. Then, successor Proxels are created iteratively for each time step, one for each possible single state transition of each Proxel. Thus, these new Proxels represent all states that are reachable from Proxels of the previous emission time. The probability of a Proxel representing a state change through transition i is hereby computed as the product of:

– the probability of the predecessor Proxel.
– the probability that no event happened in between the two symbol emissions. This is computed by numerically solving an ordinary differential equation based on the hazard rate of all active transitions over the time interval between both symbol emissions [10].
– the hazard rate $\mu_i(\tau_i + \Delta t)$ of transition i, which is the likelihood of transition i to fire in exactly than instant. Here, Δt is the time between the previous and current symbol emissions.
– the probability that the transition in question emits the observed symbol, as recorded in the HnMM specification.

The algorithm so far would result in an exponential growth in the number of Proxels over time, since each Proxel of a given time step causes the creation of multiple Proxels (one for each possible state transition) at the next time step. This effect is countered by Proxel-merging: If two Proxels with identical marking and age vector (m, τ, p_1) and (m, τ, p_2) exist for the same time step, they are merged to form a single Proxel $(m, \tau, p_1 + p_2)$. Proxel merging mitigates the exponential growth of the number of Proxels and for most models even creates an upper bound for the number of Proxels per time step.

The algorithm in pseudocode is detailed in Algorithm 1.

Encoding Additional Information. The algorithm as detailed above computes the probability measures for all reachable states, but does not retain any information on the history of how a state was reached. Thus, the set of Proxels output by the algorithm only contains information on which states the system could be in after emitting the last element of the protocol, and with what probability.

In order to be able to reconstruct other characteristics of the unobserved behavior, the Proxel definition needs to be extended with further supplementary variables that store information on those characteristics. This can for example be the number of past occurrences of a particular event. In the case of the Tester model, one would encode the number of times that machine 0 has produced defective items so far. This way, the set of Proxels at the final time step contains detailed information about how often the relevant event may have occurred with what probability. This information can then be used to compute a likelihood histogram, the expected value or the most likely value for the quantity in question.

Algorithm 1. Pseudocode for the event-driven Proxel solver.

```
Data: initialProxels, trace
Output: nextStepProxels
currentProxels = initialProxels;
prevEmissionTime = 0;
foreach element ∈ trace do
    nextStepProxels.clear();
    $\Delta t$ = element.emissionTime - prevEmissionTime;
    prevEmissionTime = element.emissionTime;
    foreach proxel ∈ currentProxels do
        stayProb = computeInactivityProbability(proxel , $\Delta t$);
        foreach trans ∈ proxel.state.transitions do
            pSucc = proxel.createSuccessor(trans, $\Delta t$);
            pSucc.$p$ = proxel.$p$ * stayProb * $\mu_t$(p.$\tau + \Delta t$) *
            trans.emissionProbability(element.symbol);
            nextStepProxels.addOrMerge(pSucc);
    currentProxels = nextStepProxels;
return nextStepProxels
```

The downside of this approach is that the number of reachable Proxel markings increases by a constant factor depending on the range of the new supplementary variable. Thus, computation time and memory consumption increase by at least the same factor. While this often has a heavy impact (the computation time for the Tester model increases about 50-fold), it does not in principle affect the feasibility of the analysis.

With this modification, the algorithm is able to reconstruct the likely system behavior during the time interval for which the trace was recorded. However, for reasons detailed in the next section, it does not allow for the reconstruction of model parameter values.

Obstacles for the Reconstruction of Model Parameters. By encoding additional information using supplementary variables, one could theoretically also reconstruct parts of the system specification. In the example of the Tester model, the unknown defect probability of each machine is simply the number of produced defective items divided by the total number of produced items. The latter can be estimated from the expected value of the probability distributions associated with the production process. And the former could be derived from a model analysis for which the number of produced *defective* items from one of the machines is encoded into each Proxel as a supplementary variable[1].

However, the parameters that are to be computed by this algorithm are also required as inputs to it: The unknown defect probabilities are the symbol emission probabilities of the model. Thus, the algorithm could only be used to confirm

[1] The number of produced defective items needs to be determined only for one of the two machines. Since the total number of defects is known from the signal trace, the number of defects produced by the other machine can easily be derived from the number of the first one.

a known set of parameters, but not to compute them in the first place. The main contribution of this work are therefore two new algorithms that resolve this contradiction. These are detailed in the following sections.

4 Iterative Analysis Algorithm

The iterative analysis approach was developed based on conclusions drawn from the naïve approach of simply guessing the unknown values and assuming that the guess will not impact the accuracy of the analysis. This naïve approach was evaluated using the Tester system, where the symbol emission probabilities of both machines to produce defective or working items are unknown. For the Tester, the "real" system is actually a simulation model. Therefore, the usually unobservable quantities can be observed here, and the quality of the reconstructed parameter values can be assessed through comparison with the actual system parameter values.

Without additional information, a suitable guess for the defect rates of the two machines is that both are equal and together account for all defects actually occurred, i.e. $p_0 = p_1 = 0.07$. Using these estimates, the *reconstructed* defect probabilities are 0.081 and 0.060, respectively, while the *actual* defect probabilities are 0.100 and 0.046. The data shows that the initial assumption was wrong and that there is a big discrepancy between the reconstructed parameter values and the actual ones. However, it also shows that in this case the result of the reconstruction *is* closer to the (usually unknown) actual system behavior than the initial guess.

This observation that an event-driven analysis "refines" estimates of unknown parameter values lead to the following iterative analysis algorithm: initially, one guesses arbitrary initial values for the unknown parameters. Then, the most recently estimated set of parameter values is iteratively used to perform an event-driven Proxel analysis in order to obtain more accurate estimates for the parameters. The iteration process terminates, if the difference in the parameter values between two iterations falls below a given threshold. This is shown in Algorithm 2.

Algorithm 2. Iterative Algorithm for Parameter Reconstruction

```
Data: model, parameterEstimate, trace, threshold
Output: parameterEstimate
repeat
    oldEstimate = parameterEstimate;
    initialProxels = model.getInitialProxels(parameterEstimate);
    finalStepProxels = eventDrivenAnalysis(initialProxels, trace);
    parameterEstimate = extractParameters(finalStepProxels);
until |parameterEstimate - oldEstimate| < threshold ;
return computedParameters
```

Figure 3 shows the analysis result for the defect probability of Machine 0 of the first 14 iterations for the Tester model. With every new iteration, the results

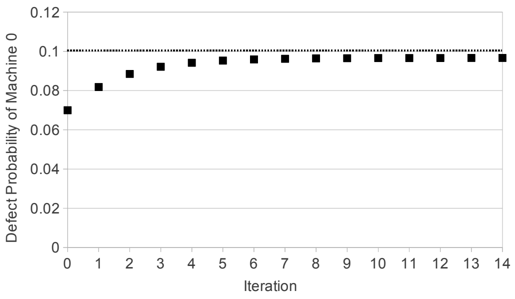

Fig. 3. Computed expected defect probability of Machine 0 using iterative analysis. Iteration 0 is the initial estimate. The dotted line represents the actual system behavior.

further approach the actual system behavior. And the results converge quickly to a value close to the actual system behavior. The difference between the value of convergence and the actual system behavior is tested and discussed further in the "Experiments" and "Conclusion" sections.

The size of the difference between the reconstructed parameter value of two iteration steps (and thus the rate of convergence) seems to correspond to the slope of the likelihood polynomial for that parameter (cf. figures 3 and 4). We therefore assume that this algorithm explores the parameter space in the same way that the gradient descent algorithm does. In this case, it will reliably find a parameter set that is a local likelihood maximum. It may, however, not necessarily find the *global* likelihood maximum (i.e. the most likely parameter set). All experiments conducted support this hypothesis, but no formal proof of this behavior exists as of yet.

5 Symbolic Polynomial Analysis

The iterative analysis is a simple algorithm that usually finds accurate estimates for unknown model parameter values, but does not give any guarantees about time-to-convergence and optimality of the values found. In those cases where all unknown probabilities can be expressed as polynomials in a single common variable, this problem can be solved more reliably by direct computation.

The direct computation approach follows the paradigm of *maximum likelihood estimation* (MLE): Determine a mathematical expression that represents the likelihood of the model to explain the observations made, i.e. the symbol emissions as recorded in the trace, dependent on the unknown parameter. Then, find the global maximum of said expression to determine the most likely parameter value.

In the case of PODS systems, the observations recorded in the trace are usually not statistically independent (e.g. the time at which a machine produces an item depends on the time at which it finished production of the previous item). Thus, the usual MLE approach of computing a likelihood expression by either multiplying the probabilities for the occurrences of the individual observations or adding their log-likelihoods is not applicable. But the normal event-driven analysis algorithm readily serves as an alternative: Since the probability of a single Proxel of the last time step of an analysis is a measure for the likelihood of the system to be in a single state after having emitted the trace, the sum of the probabilities of *all* Proxels for that time step is a measure for the likelihood of the system to be in any state at all after having emitted that trace, and therefore a measure of likelihood for the trace to have been emitted by that model.

Thus, for a given value of an unknown model parameter a, the event-driven analysis computes a measure of likelihood for the model to have emitted the trace, if the value of the unknown parameter was a. If no value is given for the unknown parameter and the computations are performed symbolically by keeping a as a symbolic variable, then the event-driven analysis computes a mathematical expression - dependent on the unknown parameters - of the likelihood for the model to have emitted the trace - which is the first step in an MLE algorithm.

To achieve this, we propose the following approach: First, express all unknown parameters by polynomials in a single variable. This is necessary in order to make the following computations practically feasible. Since this is not always possible, the approach is not applicable to all models.

Then, perform an event-driven Proxel analysis of the trace with the incomplete model, but perform all computations of Proxel probabilities not numerically, but with symbolic polynomials. This allows for the computations to be performed even while some parameter values are missing. Consequently, the Proxel probabilities will not be numbers, but symbolic polynomials as well. Since the polynomials are limited to a single variable, the length of the Proxel probability polynomials increases only by a constant amount with each time step (since a probability polynomial may be multiplied by the constant size polynomial of an unknown model parameter; merging Proxels does not increase the degree of the polynomial at all), making the algorithm computationally feasible. After the algorithm has terminated, sum up the polynomials of all Proxel probabilities of the final time step in order to obtain the likelihood polynomial (cf. Figure 4 for the likelihood polynomial of the incomplete Tester model).

The final step of the MLE is then to determine the position of the maximum of this polynomial in the valid range (the range of values for the single free variable for which all unknown probabilities that were expressed as polynomials of it take on values in the interval $[0;1]$). This can be done either through numerical root-finding of the symbolic first derivative of that polynomial, or by evaluating the polynomial using random values for the free variable and accepting the position of the sample with the highest function value as a close-enough estimate for the maximum. In both cases, evaluating the probability polynomials for the unknown parameters at the position determined by this step then returns their most likely values and thereby reconstructs the missing model parameters.

For implementers of this algorithm, it is important to consider numerical accuracy: Since the algorithm deals with polynomials containing powers of a variable to 1000 and more, the computations are numerically unstable. We found

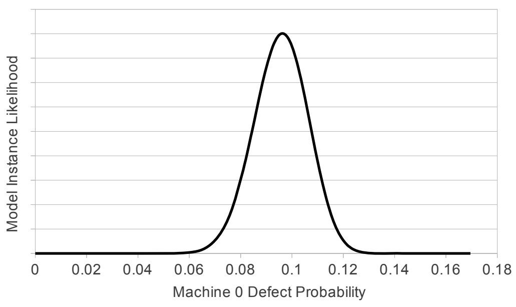

Fig. 4. Graph of the trace generation likelihood polynomial for the Tester model. The value on the horizontal axis for which the polynomial reaches its maximum is the most likely defect rate for Machine 0.

that a floating-point accuracy of at least 256 bits for all polynomial coefficients and all polynomial evaluations is required for reliable results.

A advantageous side effect of the symbolic computation approach is that it depends only on the likelihood polynomial for the *sum* of all Proxels. Thus, in contrast to the iterative analysis presented before, this approach does not need to encode history into Proxel as supplementary variables. This reduction of complexity partially offsets the increased complexity introduced by the symbolic computations and usage of the BigNum library GMP[8] for increased floating point accuracy. For an evaluation of computation times for both approaches, see the experiments in the next section.

6 Experiments

In this section, both algorithms are tested in order to determine their accuracy and computation times under varying trace length. Additionally, it is tested whether the algorithms can cope with ambiguous models.

The tests are conducted on the Tester model with traces of about 750 symbols (the equivalent of $50,000s$ of system behavior), generated using the discrete event simulator AnyLogic [4]. Using synthetic traces has the advantage that the actual defect probabilities (which are to be reconstructed using the two algorithms) can be recorded as well. Thus, the analysis result can be compared to the otherwise unobservable behavior of the "real" system.

In order to be analyzed by the polynomial approach, the four unknown parameters of the Tester model (the probabilities for each machine to produce defective or working pieces, respectively) need to be expressed as polynomials in a single variable. This is easily possible, since the "working" probability of each machine can easiliy be expressed by the corresponding "defective" probability. And the defective probability of one machine can easily be related to that of the other one, since both together have to exactly account for the actual fraction of produced defective items as recorded in the trace.

6.1 Computational Accuracy

To determine the accuracy of the approach, we created multiple signal traces from the Tester system, but from different system instances with different, randomly chosen defect probabilities for the two machines. The traces were then analyzed using the iterative and the polynomial analysis algorithms.

The results for the individual traces are shown in Figure 5. For each of the ten model instances, the actual value for the system parameter "defect probability of machine 0" are given in addition to the values reconstructed by the iterative and polynomial approaches. The results for the polynomial and iterative analysis are almost identical, giving confidence that they indeed answer the same question. And both results always differ only slightly from the actual system behavior. The difference between both reconstructions and the actual system behavior is likely to be unavoidable: Both algorithms only reconstruct the parameters values for which the system *most likely* caused the trace, but the actual system behavior is not necessarily identical to the most likely explanation.

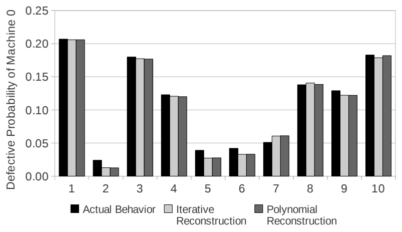

Fig. 5. Simulation results for the analysis of ten traces with different defect rates in the real system

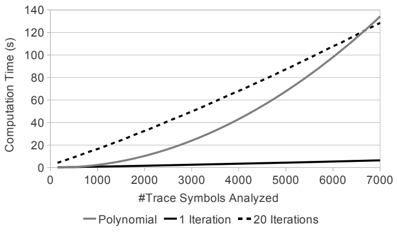

Fig. 6. Increase of computation time under increasing trace length for the polynomial and iterative analysis algorithms

6.2 Computation Time

The computation time of the algorithms is relevant to assess their practical applicability, especially with longer traces. For this experiment, we generated a trace covering $500,000s$ (about 5 days) of simulated time containing about $7,500$ symbols; ten times as many as the traces for the other experiments. The computation time used was then recorded every analyzed $10,000s$ for both algorithms.

Figure 6 shows the raw computation times, which need to be interpreted with care: For the polynomial algorithm, a single analysis run is always sufficient to reconstruct the system behavior. The iterative approach on the other hand requires repetitive analyses until its results converge. To give an indication of how the computation times compare, we plotted the results for a single iteration as well as for 20 iterations – a reasonable number of iterations till convergence.

The computation time of the iterative approach increases linearly, that of the polynomial approach on the other hand increases polynomially with a low initial slope. This is due to the symbolic polynomials increasing in length over time and thus making their computations more costly. Overall, the magnitude of the computation time is similar for both algorithms. Which algorithm is faster for the analysis of a specific trace and model depends on the trace length and the number of iteration steps required for convergence.

With the polynomial approach having to multiply and store polynomials using high precision floating point numbers that the CPU does not natively support, one would expect the polynomial approach to be slower than the iterative one by orders of magnitude even for short traces. But the polynomial algorithm does not have to encode history information into the system states (cf. Section 3.2), which leads to a far lower number of Proxels required for the analysis and partially compensates for the increased computational effort required for each Proxel.

6.3 Analysis of Ambiguous Systems

Some systems have multiple equally likely explanations for their internal behavior. For example, if the two machines in the Tester system had identical production behavior (i.e. identical probability distribution and distribution parameters), they could not be distinguished. An algorithm suitable for practical applications should fail gracefully in these cases.

To test algorithm behavior on this class of models, we created a trace for a modified Tester model where both production durations were $N(120, 20)$ distributed, but had different defect probabilities. The trace was then analyzed by both new algorithms.

Figure 7 shows the progression of the iterative analysis. Since the initial guess for the defect probabilities is almost (due to floating point inaccuracies) identical to the local minimum of the likelihood polynomial (cf. Figure 8), the result barely changes between the first few iterations. The iterative algorithm, which uses the difference between the results of two iterations as a termination criterion, would ordinarily terminate right away and would return this locally most unlikely parameter value as the most defect probability. If forced to perform further iterations, the results diverge only after 60 iterations to one of the two local maxima, depending on on which side of the likelihood minimum the initial estimate was located. In effect, the algorithm randomly proclaims one of the two different but equally likely outcomes to be the most likely system behavior, and ignores the second one.

The polynomial approach reliably finds both maxima (cf. Figure 8). So, it correctly determines that one machine has caused about twice as many defects as the other one, and correctly states that it is not possible to distinguish the two machines.

Thus, the iterative analysis is unreliable in the general case since it may find the global maximum, or just a random local maximum, or may even be stuck in a local minimum of the likelihood. It is only reliable when the likelihood polynomial is guaranteed to only contain a single maximum and no minima. The polynomial approach, on the other hand, always finds the most likely value(s) for the unknown parameter irrespective of the shape of the likelihood polynomial.

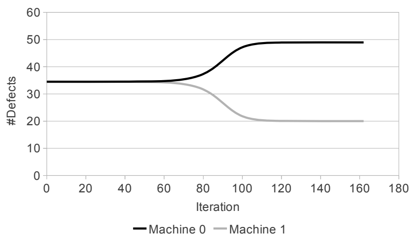

Fig. 7. Progression of the iterative analysis of a trace from an ambiguous model

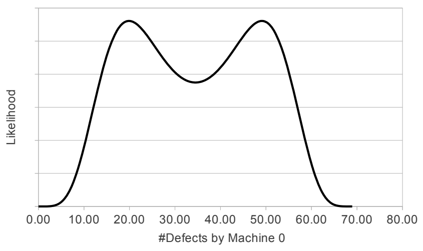

Fig. 8. Likelihood polynomial for a trace from an ambiguous Tester model

7 Conclusion and Outlook

In this paper, we have presented two approaches that enable the reconstruction of parts of the system specification of partially-observable discrete stochastic systems from an incomplete model and a trace of external observations of its behavior.

It should be noted that the task of both algorithms is only to reconstruct parameters of the system that *most likely* has created the trace. In the same way that a star athlete may be most likely to win a sports competition, but is not guaranteed to win it, the reconstructed most likely explanation is not necessarily the actual cause.

The iterative algorithm is a simple modification to the existing algorithms and is usually able to correctly determine the most likely parameter values in about 20 iterations. However, being a local optimization algorithm, it does not necessarily find the actual most likely system behavior, but only a value that is more likely to represent the actual system behavior than neighboring values on both sides. Additionally, its behavior is undefined if the initial estimate for the unknown parameter is less likely to represent the actual system behavior than neighboring values on both sides.

The polynomial algorithm is a more complex modification to the existing algorithms, using symbolic computations on polynomials to cope with incomplete specifications. It is thus computationally more expensive than the iterative approach, but in turn is *guaranteed* to reconstruct the most likely real system behavior.

In practice, the iterative algorithm may be preferred in real-time application scenarios due to its simplicity and speed, especially for systems where local optimization algorithms are guaranteed to also find the global optimum (i.e. those with no local minima and only a single local maximum), and where more accurate initial estimates may be available (e.g. by using the results of the trace of the previous day as an estimate for the next day) and thus fewer iterations are required. The polynomial algorithm is better suited to analyze the likelihood of different system configurations off-line and as a benchmark of accuracy for other approaches.

7.1 Future Work

So far, the algorithms presented can only be used to reconstruct constant probability values in the model specification; probability distributions and their parameters always have to be known. In the future, we plan on investigating the possibility of approximating the influence of distribution parameters through polynomials and thereby enabling the reconstruction of the most likely probability distribution parameters. This would constitute a major step in the direction of model training for partially-observable discrete stochastic systems.

References

1. van der Aalst, W.M.P.: Analysis of discrete-time stochastic petri nets. Statistica Neerlandica 54(2), 237–255 (2000)
2. Aoki, M.: State space modeling of time series. Springer-Verlag New York, Inc., New York (1986)
3. Bobbio, A., Puliafito, A., Telek, M., Trivedi, K.S.: Recent developments in non-markovian stochastic petri nets. Journal of Systems Circuits and Computers 8(1), 119–158 (1998)
4. Borshchev, A., Filippov, A.: From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools. In: Proceedings of 22nd International Conference of the System Dynamics Society, Oxford, England (July 2004)
5. Boys, R.J., Wilkinson, D.J., Kirkwood, T.B.: Bayesian inference for a discretely observed stochastic kinetic model. Statistics and Computing 18, 125–135 (2008)
6. Buchholz, R., Krull, C., Horton, G.: Efficient event-driven proxel simulation of a subclass of hidden non-markovian models. In: 7th EUROSIM Congress on Modelling and Simulation (2010)
7. Buchholz, R., Krull, C., Horton, G., Strigl, T.: Using hidden non-markovian models to reconstruct system behavior in partially-observable systems. In: 3rd International ICST Conference on Simulation Tools and Techniques (2010)
8. FSF. The gnu multiprecision library (gmp), `http://gmplib.org`
9. Gibson, G.J., Renshaw, E.: Estimating parameters in stochastic compartmental models using markov chain methods. Mathematical Medicine and Biology 15(1), 19–40 (1998)
10. Horton, G.: A new paradigm for the numerical simulation of stochastic petri nets with general firing times. In: European Simulation Symposium. SCS European Publishing House, Dresden (2002)
11. Krull, C., Buchholz, R., Horton, G.: Matching hidden non-markovian models: Diagnosing illnesses based on recorded symptoms. In: The 24th annual European Simulation and Modelling Conference (October 2010)
12. Krull, C., Horton, G.: Hidden non-markovian models: Formalization and solution approaches. In: Proceedings of 6th Vienna International Conference on Mathematical Modelling, Vienna, Austria (February 2009)
13. Lazarova-Molnar, S.: The Proxel-Based Method: Formalisation, Analysis and Applications. Ph.D. thesis, Otto-von-Guericke University Magdeburg (2005)

14. Malyarenko, A., Vasiliev, V.: On guaranteed parameter estimation of discrete-time stochastic systems. In: Second International Conference on Innovative Computing, Information and Control ICICIC 2007, p. 140 (September 2007)
15. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
16. Wang, Y., Christley, S., Mjolsness, E., Xie, X.: Parameter inference for discretely observed stochastic kinetic models using stochastic gradient descent. BMC Systems Biology 4(1), 99 (2010)