2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems

Efficient Timeout Synthesis in Fixed-Delay CTMC Using Policy Iteration

Ľuboš Korenčiak, Antonín Kučera, Vojtěch Řehák
Faculty of Informatics, Masaryk University, Brno, Czech Republic
Email: {korenciak, kucera, rehak}@fi.muni.cz

***Abstract*—We consider the fixed-delay synthesis problem for continuous-time Markov chains extended with fixed-delay transitions (fdCTMC). The goal is to synthesize concrete values of the fixed-delays (timeouts) that minimize the expected total cost incurred before reaching a given set of target states. The same problem has been considered and solved in previous works by computing an optimal policy in a certain discrete-time Markov decision process (MDP) with a huge number of actions that correspond to suitably discretized values of the timeouts.**

In this paper, we design a *symbolic* fixed-delay synthesis algorithm which avoids the explicit construction of large action spaces. Instead, the algorithm computes a small sets of "promising" candidate actions on demand. The candidate actions are selected by minimizing a certain objective function by computing its symbolic derivative and extracting a univariate polynomial whose roots are precisely the points where the derivative takes zero value. Since roots of high degree univariate polynomials can be isolated very efficiently using modern mathematical software, we achieve not only drastic memory savings but also speedup by three orders of magnitude compared to the previous methods.

I. Introduction

Continuous-time Markov chains (CTMC) are a fundamental formalism widely used in performance and dependability analysis. CTMC can model exponentially distributed events, but not *fixed-delay* events that occur after a fixed amount of time with probability one[1]. Since fixed-delay events are indispensable when modeling systems with *timeouts* (i.e., communication protocols [29], time-driven real-time schedulers [32], etc.), a lot of research effort has been devoted to developing formalisms that generalize CTMC with fixed-delay transitions. Examples include deterministic and stochastic Petri nets [25], delayed CTMC [14], or fixed-delay CTMC (fdCTMC) [20], [5], [22].

In practice, the duration of fixed-delay events (timeouts) is usually determined ad-hoc, which requires a considerable amount of effort and expertise. Hence, a natural question is whether the (sub)optimal timeouts can be synthesized *algorithmically*. For fdCTMC, an algorithm synthesizing suboptimal timeouts was given in [5]. This algorithm is based on explicitly constructing and solving a discrete-time Markov decision process (MDP) whose actions correspond to suitably discretized admissible timeout values. Since the number of these actions is always large, the applicability of this algorithm is limited only to small instances for fundamental reasons.

Supported by the Czech Science Foundation, grant No. 15-17564S.

[1]A fixed-delay distribution is a typical example of a distribution where the standard phase-type approximation technique [27] produces a large error unless the number of auxiliary states is very large; see, e.g., [20], [11].

Our contribution. In this paper, we design a new *symbolic* algorithm for synthesizing suboptimal timeouts in fdCTMC up to an arbitrary small given error. Although we build on the results of [5], the functionality of our algorithm is different. First, the explicit construction of the aforementioned discrete-time Markov decision process is completely avoided, which drastically reduces memory requirements. Second, the search space of the underlying policy improvement procedure is restricted to a small subset of "promising candidates" obtained by identifying the local minima of certain analytical functions constructed "on-the-fly". This allows to safely ignore most of the discretized timeout values, and leads to speedup by three orders of magnitude compared to the algorithm of [5]. Consequently, our algorithm can synthesize suboptimal timeouts for non-trivial models of large size (with more than 20000 states) which would be hard to obtain manually.

The rest of this paper is structured as follows. In Section II, we introduce the fdCTMC formalism, explain its semantics, and formalize the objective of fixed-delay synthesis. In Section III, we describe the key ingredients of our algorithm in more detail. The experimental outcomes are presented in Section IV. In Section V, we explain the relationship to previous works. The missing details and further experimental results can be found in a full version of this paper [21].

II. Fixed-delay CTMC and the synthesis objective

A fdCTMC is a tuple $(S, \lambda, \mathrm{P}, S_{\mathrm{fd}}, \mathrm{F})$, where S is a finite set of states, $\lambda \in \mathbb{R}_{\geq 0}$ is a common exit rate for the states, $\mathrm{P} \in \mathbb{R}_{\geq 0}^{S \times S}$ is a stochastic matrix specifying the probabilities of "ordinary" exp-delay transitions between the states, $S_{\mathrm{fd}} \subseteq S$ is a subset of states where a fixed-delay transition is enabled, and $\mathrm{F} \in \mathbb{R}_{\geq 0}^{S \times S}$ is a stochastic matrix such that $\mathrm{F}(s, s) = 1$ for all $s \in S \setminus S_{\mathrm{fd}}$. For the states of S_{fd}, the matrix F specifies the probabilities of fixed-delay transitions. The states of $S \setminus S_{\mathrm{fd}}$ are declared as absorbing by F, which becomes convenient later. In addition, we specify a *delay function* $\mathbf{d} : S_{\mathrm{fd}} \to \mathbb{R}_{>0}$ which assigns a concrete delay (timeout) to each state of S_{fd}. Note that (S, λ, P) is an "ordinary" CTMC where the time spent in the states of S is determined by the exponential distribution with the same[2] parameter λ.

[2]We can assume without restrictions that the parameter λ is the same for all states of S, because every CTMC can be effectively transformed into an equivalent CTMC satisfying this property by the standard uniformization method; see, e.g., [28]. Note that the transformation causes zero error.

2375-0227/16 $31.00 © 2016 IEEE
DOI 10.1109/MASCOTS.2016.34

The fdCTMC semantics can be intuitively described as follows. Imagine that the underlying CTMC (S, λ, P) is now equipped with an alarm clock. When the alarm clock is turned off, our fdCTMC behaves exactly as the underlying CTMC. Whenever a state s of S_{fd} is visited and the alarm clock is off at the time, it is turned on and set to ring after $\mathbf{d}(s)$ time units. Subsequently, the process keeps behaving as the underlying CTMC until either a state of $S \setminus S_{\mathrm{fd}}$ is visited (in which case the alarm clock is turned off), or the accumulated time from the moment of turning the alarm clock on reaches the value when the alarm clock *rings* in some state s' of S_{fd}. In the latter case, an outgoing fixed-delay transition of s' takes place, which means that the process changes the state randomly according to the distribution $\mathrm{F}(s', \cdot)$, and the alarm clock is either newly set or turned off (depending on whether a state of S_{fd} or $S \setminus S_{\mathrm{fd}}$ is entered, respectively).

Example 1: Consider a simple communication protocol where Alice tries to establish a connection with Bob via an unreliable communication channel. Alice starts by sending an *Invite* message to Bob, and then she waits for Bob's *Ack* message. Since each of these messages can be lost, Alice sets a timeout after which she restarts the protocol and sends another *Invite* (the *Ack* messages confirming a successful receipt of a "previous" *Invite* are recognized and ignored). The protocol terminates when a connection is established, i.e., both messages are delivered successfully before the timeout. The behaviour of the unreliable channel is stochastic; a message is successfully delivered with a (known) probability p, and the delivery time has a (known) distribution *Dtime*. A simplified fdCTMC model of the protocol is given in Fig. 1. The "ordinary" (i.e., exp-delay) and fixed-delay transitions are indicated by solid and dashed arrows, respectively, together with the associated probabilities. A faithful modeling of the *Dtime* distribution using the phase-type approximation requires extra auxiliary states which are omitted[3] in Fig. 1 for the sake of simplicity (the main point is to illustrate the use of fixed-delay transitions). Note that the alarm clock is set in the initial state A, and it is switched off in the terminal state C. If the alarm clock rings in any state except for C, the protocol is restarted and the alarm clock is reset. Now, the question is how to set the timeout so that the expected time needed to complete the protocol (i.e., to reach the state C from the state A) is minimized. If the timeout is too large, a lot of time is wasted by waiting in the failure state F. If it is too small, there is not enough time to complete the communication and the protocol is restarted many times before it succeeds. In this particular case, one may still argue that an optimal timeout can be computed by hand and no synthesis algorithm is needed. Now consider a more complicated scenario where Alice tries to establish a simultaneous connection with $\mathrm{Bob}_1, \ldots, \mathrm{Bob}_n$ via different unreliable channels which are also unstable (i.e., an already established link with Bob_i gets broken after a random time whose distribution is known). This scenario can still be modeled by a fdCTMC, and Alice needs to determine a suitable timeout which achieves the optimal expected time of completing the protocol. Since the properties of the individual channels can be different and the probability of breaking an already established connection increases as more and more Bobs get connected, the timeout chosen by Alice should actually depend on the subset of connections that remain to be established. The corresponding tuple of optimal timeouts is hard to compute manually. However, as we shall see in Section IV, a solution can be synthesized by our algorithm. □

[3]Hence, the simplified model corresponds to the situation when *Dtime* is the exponential distribution with parameter λ.

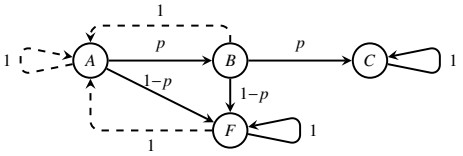

Fig. 1. A simplified fdCTMC model of a communication protocol.

Now we explain the objective of fixed-delay synthesis. Intuitively, given a fdCTMC, the goal is to compute a delay function which minimizes the expected total cost incurred before reaching a given set of target states G starting in a given initial state s_{in}. For the fdCTMC of Example 1, the set of target states is $\{C\}$, the initial state is A, and the costs correspond to the elapsed time. Our aim is to model general performance measures (not just the elapsed time), and therefore we use the standard cost structures that assign numerical costs to both states and transitions (see, e.g., [30]). More precisely, we consider the following three cost functions: $\mathcal{R} : S \to \mathbb{R}_{>0}$, which assigns a cost rate $\mathcal{R}(s)$ to every state s so that the cost $\mathcal{R}(s)$ is paid for every time unit spent in the state s, and functions $\mathcal{I}_{\mathrm{P}}, \mathcal{I}_{\mathrm{F}} : S \times S \to \mathbb{R}_{\geq 0}$ that assign to each exp-delay and fixed-delay transition the associated execution cost.

For every delay function $\mathbf{d}$, let $E_{\mathbf{d}}$ be the expected total cost incurred before reaching a target state of G starting in the initial state s_{in} (note that when $\mathbf{d}$ is fixed, the behaviour of the considered fdCTMC is fully probabilistic). For a given $\varepsilon > 0$, we say that a delay function $\mathbf{d}$ is *ε-optimal* if

$$\left| E_{\mathbf{d}} - \inf_{\mathbf{d}'} E_{\mathbf{d}'} \right| \;<\; \varepsilon.$$

Here, $\mathbf{d}'$ ranges over all delay functions. The *fixed-delay synthesis problem* for fdCTMC is to compute an ε-optimal delay function (for a given $\varepsilon > 0$).

III. Our algorithm for the fixed-delay synthesis problem

For purposes of this section, we fix a fdCTMC $(S, \lambda, \mathrm{P}, S_{\mathrm{fd}}, \mathrm{F})$, cost functions $\mathcal{R}$, $\mathcal{I}_{\mathrm{P}}$, $\mathcal{I}_{\mathrm{F}}$, an initial state s_{in}, and a set of target states G.

As we already mentioned, our fixed-delay synthesis algorithm for fdCTMC is *symbolic* in the sense that it avoids explicit constructions of large action spaces and allows to safely disregard a large subsets of actions that correspond to discretized timeout values. To explain what all this means, we need to introduce some extra notions. Let $S_{\mathrm{off}} = S \setminus S_{\mathrm{fd}}$ be

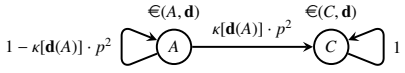

Fig. 2. The structure of $\mathcal{M}_{\mathbf{d}}$ for the fdCTMC of Fig. 1.

the set of all states where fixed-delay transitions are disabled. Further, let $S_{\text{set}} \subseteq S_{\text{fd}}$ be the set of all states where a timeout can be (re)set, i.e., S_{set} consists of all $s \in S_{\text{fd}}$ such that s has an incoming exp-delay transition from a state of S_{off}, or an incoming fixed-delay transition (from any state). For the fdCTMC of Fig. 1, we have that $S_{\text{off}} = \{C\}$ and $S_{\text{set}} = \{A\}$ (note the timeout is never set in the states B and F, although the "alarm clock" is turned on in these states). Without restrictions, we assume that the initial state s_{in} and the each target state of G belong to $S_{\text{off}} \cup S_{\text{set}}$ (otherwise, we can trivially adjust the structure of our fdCTMC).

Now, let us fix a delay function $\mathbf{d}$. If the execution of our fdCTMC is initiated in a state $s \in S_{\text{off}} \cup S_{\text{set}}$ (for $s \in S_{\text{set}}$, the timeout is set to $\mathbf{d}(s)$), then a state s' such that either $s' \in S_{\text{off}}$, or $s' \in S_{\text{set}}$ and the timeout is (re)set in s', is visited with probability one. Note that the timeout is (re)set in $s' \in S_{\text{set}}$ if the transition used to enter s' is either fixed-delay (i.e., the alarm clock just rang and needs to be set again), or exp-delay and the previous state belongs to S_{off} (i.e., the alarm clock was off and needs to set on now). Hence, for every $s \in S_{\text{off}} \cup S_{\text{set}}$, we can define the probability distribution $T(s, \mathbf{d})$ over $S_{\text{off}} \cup S_{\text{set}}$, where $T(s, \mathbf{d})(s')$ is the probability that the first visited state satisfying the above condition is s'. At the moment, it is not yet clear how to compute/approximate the distribution $T(s, \mathbf{d})$, but it is correctly defined. Further, for every $s \in S_{\text{off}} \cup S_{\text{set}}$, let $€(s, \mathbf{d})$ be the expected total cost incurred before reaching a state s' satisfying the above condition (and starting in s). Thus, we obtain a *discrete-time* Markov chain $\mathcal{M}_{\mathbf{d}}$ with the set of states $S_{\text{off}} \cup S_{\text{set}}$ where each state s is assigned the cost $€(s, \mathbf{d})$. For the fdCTMC of Fig. 1, the structure of $\mathcal{M}_{\mathbf{d}}$ is shown in Fig. 2. Here, $\kappa[\mathbf{d}(A)]$ is the probability of executing at least two exp-delay transitions in time $\mathbf{d}(A)$. Note that $€(C, \mathbf{d})$ is independent of $\mathbf{d}$.

It is not hard to show that the Markov chain $\mathcal{M}_{\mathbf{d}}$ faithfully mimics the behaviour of the considered fdCTMC for the delay function $\mathbf{d}$. More precisely, $E_{\mathbf{d}}$ (i.e., the expected total cost incurred in our fdCTMC before reaching a target state of G starting in s_{in}) is equal to the expected total cost incurred in $\mathcal{M}_{\mathbf{d}}$ before reaching a state of G starting in s_{in}. Since we do not aim at computing $E_{\mathbf{d}}$ for a given $\mathbf{d}$ but on synthesizing a suboptimal $\mathbf{d}$, the Markov chain $\mathcal{M}_{\mathbf{d}}$ does not appear very useful. However, $\mathcal{M}_{\mathbf{d}}$ can be transformed into a *discrete-time Markov decision process* $\mathcal{M}$ which serves this goal. Here we use the result of [5] which, for a given $\varepsilon > 0$ and every $s \in S_{\text{fd}}$, allows to construct a *finite* set $Dval(s)$ of discrete timeout values such that an ε-optimal $\mathbf{d}$ is guaranteed to exist even if $\mathbf{d}(s)$ is restricted to $Dval(s)$. For technical reasons, we also put $Dval(s) = \{\infty\}$ for all $s \in S_{\text{off}}$.

Note that for every $s \in S_{\text{set}}$, the distribution $T(s, \mathbf{d})$ and the cost $€(s, \mathbf{d})$ depends just of $\mathbf{d}(s)$. To simplify our notation, we often write $T(s, \tau)$ and $€(s, \tau)$ to denote $T(s, \mathbf{d})$ and $€(s, \mathbf{d})$ where $\mathbf{d}(s) = \tau$. For $s \in S_{\text{off}}$, the distribution $T(s, \mathbf{d})$ and the cost $€(s, \mathbf{d})$ are independent of $\mathbf{d}$. To unify our notation for all elements of $S_{\text{off}} \cup S_{\text{set}}$, we write $T(s, \tau)$ and $€(s, \tau)$ also when $s \in S_{\text{off}}$, even if the τ is irrelevant.

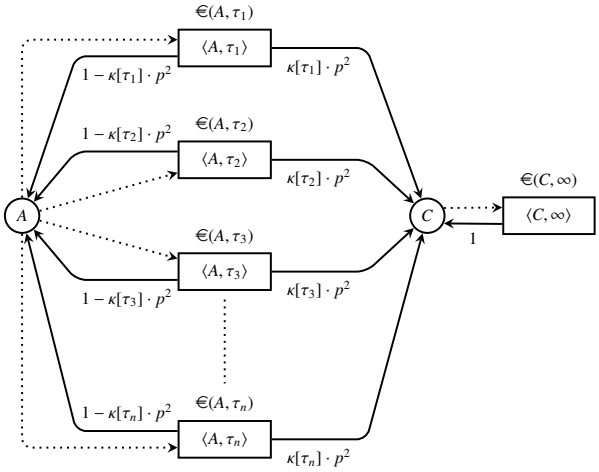

Fig. 3. The structure of $\mathcal{M}$ for the fdCTMC of Fig. 1.

The MDP $\mathcal{M}$ is constructed as follows. For every state $s \in S_{\text{off}} \cup S_{\text{set}}$ and every $\tau \in Dval(s)$, we add a special action $\langle s, \tau \rangle$ enabled in s. The outgoing transitions of $\langle s, \tau \rangle$ are now "copied" from $\mathcal{M}_{\mathbf{d}}$, i.e., the probability of entering a state $s' \in S_{\text{off}} \cup S_{\text{set}}$ after selecting the action $\langle s, \tau \rangle$ is $T(s, \tau)(s')$. Further, the action $\langle s, \tau \rangle$ is assigned the cost $€(s, \tau)$. For the fdCTMC of Fig. 1, the structure of $\mathcal{M}$ is shown in Fig. 3.

An ε-optimal delay function $\mathbf{d}$ can now be obtained by computing an optimal stationary policy minimizing the expected total cost incurred in $\mathcal{M}$ before reaching a target state of G starting in s_{in} (this can be achieved by a standard policy improvement algorithm; see, e.g., [30]). For every $s \in S_{\text{set}}$, we put $\mathbf{d}(s) = \tau_j$, where $\langle s, \tau_j \rangle$ is the action selected by the optimal stationary policy. For the remaining $s \in S_{\text{fd}} \setminus S_{\text{set}}$, we set $\mathbf{d}(s)$ arbitrarily.

The fixed-delay synthesis algorithm of [5] constructs the MDP $\mathcal{M}$ explicitly, where all $T(s, \tau)$ and all $€(s, \tau)$ are approximated up to a sufficiently small error before computing an optimal policy. Note that for the fdCTMC of Fig. 1, this essentially means to try out all possibilities in the discretized candidate set $Dval(A)$. Since the candidate sets $Dval(s)$ are large, this approach cannot be applied to larger instances.

The algorithm presented in this paper avoids the explicit construction of $\mathcal{M}$. The key idea is to express $T(s, \tau)$ and $€(s, \tau)$ *analytically* as functions of τ. More precisely, for each $s \in S_{\text{off}} \cup S_{\text{set}}$, we consider the following two functions:

- $T_s : \mathbb{R}_{\geq 0} \to \mathcal{D}(S_{\text{off}} \cup S_{\text{set}})$, where $\mathcal{D}(S_{\text{off}} \cup S_{\text{set}})$ is the set of all probability distributions over $S_{\text{off}} \cup S_{\text{set}}$. The function is defined by $T_s(\tau) = T(s, \tau)$.
- $€_s : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ defined by $€_s(\tau) = €(s, \tau)$.

Further, for every $s, s' \in S_{\text{off}} \cup S_{\text{set}}$, let

- $T_{s,s'} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be defined by $T_{s,s'}(\tau) = T(s,\tau)(s')$.

The functions T_s and $\in_s$ can be expressed as certain infinite sums, but for every fixed error tolerance, this sum can be effectively truncated to finitely many summands. The functions T_s and $\in_s$ are then used in the symbolic policy improvement algorithm. We start with some (randomly chosen) eligible delay function such that $\mathbf{d}(s) \in \mathit{Dval}(s)$ for all $s \in S_{\mathrm{fd}}$. Then, we repeatedly improve $\mathbf{d}$ until no progress is achieved. Each improvement round has two phases. First, we evaluate the *current* $\mathbf{d}$ in all states of $S_{\mathrm{off}} \cup S_{\mathrm{set}}$. That is, for each $s \in S_{\mathrm{off}} \cup S_{\mathrm{set}}$ we approximate the value $E^s_{\mathbf{d}}$, which is equal to $E_{\mathbf{d}}$ when the initial state is changed to s, up to a sufficient precision. Then, for each state $s \in S_{\mathrm{set}}$, we try to identify the action $\langle s, \tau \rangle$ such that the timeout τ *minimizes* the function $K_s : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ defined by $K_s(\tau) = \sum_{s' \in S_{\mathrm{off}} \cup S_{\mathrm{set}}} T_{s,s'}(\tau) \cdot E^{s'}_{\mathbf{d}} + \in_s(\tau)$. Instead of trying out all $\tau \in \mathit{Dval}(s)$ one by one, we compute the *symbolic derivative* of K_s, which is possible due to the analytical form of T_s and $\in_s$. Further, it turns out that the derivative takes zero value iff a certain effectively constructable *univariate polynomial* takes zero value. Hence, we only need to deal with those $\tau \in \mathit{Dval}(s)$ which are "close" to the roots of this polynomial, and we may safely ignore the others. Since the roots of univariate polynomials are easy to approximate using modern mathematical software (such as Maple), this approach is rather efficient and the set of relevant τ's obtained in this way is *much* smaller than $\mathit{Dval}(s)$. This is why our algorithm outperforms the one of [5] so significantly.

Now we give analytical definitions of the two crucial functions T_s and $\in_s$. For $s \in S_{\mathrm{off}}$, we simply put $T_s(\tau) = \mathrm{P}(s, \cdot)$ and $\in_s(\tau) = \mathcal{R}(s)/\lambda + \sum_{s' \in S} \mathrm{P}(s,s') \cdot \mathcal{I}_{\mathrm{P}}(s,s')$. Observe that both of these functions are constant. Now let $s \in S_{\mathrm{set}}$. Then $T_s(\tau)$ and $\in_s(\tau)$ need to "summarize" the behaviour of our fdCTMC C starting in the configuration (s,τ) until a state s' is reached such that either $s' \in S_{\mathrm{off}}$, or $s' \in S_{\mathrm{set}}$ and the timeout is reset in s'. To achieve that, we define the stochastic matrix $\overline{\mathrm{P}} \in \mathbb{R}^{S \times S}_{\geq 0}$ where $\overline{\mathrm{P}}(s,\cdot) = \mathrm{P}(s,\cdot)$ for all $s \in S_{\mathrm{fd}}$, and $\overline{\mathrm{P}}(s,s) = 1$ for all $s \in S_{\mathrm{off}}$. Thus, we obtain

$$T_s(\tau) = \sum_{i=0}^{\infty} e^{-\lambda\tau} \frac{(\lambda\tau)^i}{i!} \cdot \left(\mathbf{1}_s \cdot \overline{\mathrm{P}}^i \right) \cdot \mathrm{F} \,.$$

The function $\in_s(\tau)$ is slightly more complicated, because we also need to evaluate the total costs incurred before reaching a state s' satisfying the condition stated above. Here we also employ a function $\overline{\mathcal{R}}$ which is the same as $\mathcal{R}$ but returns 0 for all states of S_{off}, and functions $\overline{\mathcal{I}_Q}, \overline{\mathcal{I}_{\mathrm{F}}} : S \to \mathbb{R}_{\geq 0}$ that assign to each state the expected impulse cost of the next exp-delay and the next fixed-delay transition, respectively. We obtain

$$\in_s(\tau) = \sum_{i=0}^{\infty} e^{-\lambda\tau} \frac{(\lambda\tau)^i}{i!} \left(\sum_{j=0}^{i-1} \left(\mathbf{1}_s \cdot \overline{\mathrm{P}}^j \right) \cdot \left(\frac{\tau \cdot \overline{\mathcal{R}}}{i+1} + \overline{\mathcal{I}_Q} \right) + \left(\mathbf{1}_s \cdot \overline{\mathrm{P}}^i \right) \cdot \left(\frac{\tau \cdot \overline{\mathcal{R}}}{i+1} + \overline{\mathcal{I}_{\mathrm{F}}} \right) \right).$$

One can verify that $T_s(\tau) = T(s,\tau)$ and $\in_s(\tau) = \in(s,\tau)$. For a more detailed explanation and a proof we refer to [6].

Algorithm 1: Policy Iteration for $\mathcal{M}$ [30]

input : $\mathcal{M}$ and a *Dval*-consistent delay function $\mathbf{d}'$
output: a *Dval*-consistent delay function $\mathbf{d}$ optimal for $\mathcal{M}$

```
repeat
    d := d'
    // policy evaluation
    Compute a vector x such that x(s) := E_{M[s](d)}
    foreach s ∈ S_set do
        // policy improvement
        L := argmin_{τ∈Dval(s)} T^I_s(τ) · x + ∈^I_s(τ)
        if d(s) ∈ L then
            d'(s) := d(s)
        else
            d'(s) := min L
until d = d'
```

Since $T_s(\tau)$ and $\in_s(\tau)$ are defined as infinite sums, the next step is to compute a large enough $I \in \mathbb{N}$ such that the first I summands of $T_s(\tau)$ and $\in_s(\tau)$ approximate $T(s,\tau)$ and $\in(s,\tau)$ with a sufficient accuracy. Here we borrow another result of [5], where a sufficiently small approximation error κ for evaluating $T(s,\tau)$ and $\in(s,\tau)$ when constructing the MDP $\mathcal{M}$ was given. Hence, it suffices to find a sufficiently large $I \in \mathbb{N}$ such that the reminder of the constructed series is bounded by κ (for all $s \in S_{\mathrm{off}} \cup S_{\mathrm{set}}$ and $\tau \in \mathit{Dval}(s)$). Since we have an upper bound $\tau_{\max}$ on the size of τ, an appropriate I can be computed easily. From now on, we use $T^I_s(\tau)$ and $\in^I_s(\tau)$ as $T_s(\tau)$ and $\in_s(\tau)$, respectively, where the infinite sums are truncated to the first I summands only.

As we already mentioned, our fixed-delay synthesis algorithm is essentially a "symbolic" variant of the standard policy iteration algorithm [30] applied to the MDP $\mathcal{M}$ where the actions of $\mathcal{M}$ are not constructed explicitly but generated "on demand". We start by recalling the standard policy iteration which assumes that $\mathcal{M}$ is given explicitly (see Algorithm 1). This algorithm starts with some (arbitrary) *Dval*-consistent delay function $\mathbf{d}'$ (recall that we do not distinguish between *Dval*-consistent delay functions and policies) and gradually improves this function until reaching a fixed point. Each iteration consists of two phases: *policy evaluation* and *policy improvement*. In the policy evaluation phase, the vector $\mathbf{x}$ is computed, such that $\mathbf{x}(s)$ is the expected total cost until reaching a target state when starting from s and using the policy $\mathbf{d}$. This can be done in polynomial time by solving a set of linear equations. In the policy improvement phase, a new delay function is obtained by choosing a new action separately for each state of S_{set}[4]. First, the set of actions $\mathrm{argmin}_{\tau \in \mathit{Dval}(s)} T^I_s(\tau) \cdot \mathbf{x} + \in^I_s(\tau)$ is computed and then some of them is picked, but the old action $\mathbf{d}(s)$ must be chosen whenever possible. Policy iteration terminates in a finite number of steps and returns an optimal policy [12].

[4]For the remaining states we have only one action, so there is nothing to improve.

Our symbolic algorithm is obtained by modifying Algorithm 1. The policy evaluation step is efficient and here we do not need to implement any changes. In the policy improvement, we proceed differently. Due to our analytical representation of $T^I_s(\tau)$ and $\in^I_s(\tau)$, we can now interpret $T^I_s(\tau) \cdot \mathbf{x} + \in^I_s(\tau)$ as a function

$$f_{s,\mathbf{x}}(\tau) = e^{-\lambda\tau} \cdot p_{s,\mathbf{x}}(\tau),$$

where $p_{s,\mathbf{x}}(\tau)$ is a univariate polynomial whose degree is bounded by I. Note that $f_{s,\mathbf{x}}$ is continuous and easily differentiable. Hence, we can identify the (global) minima of $f_{s,\mathbf{x}}$ in the interval $[\alpha_s, \beta_s]$, where $\alpha_s = \min Dval(s)$ and $\beta_s = \max Dval(s)$, which are the points where the first derivative of $f_{s,\mathbf{x}}$ is zero, or the bounds of the interval. Let $f'_{s,\mathbf{x}}$ be the first derivative of $f_{s,\mathbf{x}}$. Then

$$f'_{s,\mathbf{x}}(\tau) = e^{-\lambda\tau} \cdot \Big((p_{s,\mathbf{x}}(\tau))' - \lambda \cdot p_{s,\mathbf{x}}(\tau) \Big)$$

where $e^{-\lambda\tau} > 0$ for all $\tau \in \mathbb{R}_{\geq 0}$. Thus, we can restrict ourselves to root isolation of a univariate polynomial

$$q_{s,\mathbf{x}}(\tau) = (p_{s,\mathbf{x}}(\tau))' - \lambda \cdot p_{s,\mathbf{x}}(\tau)$$

with a finite degree bounded by I. A full description of the resulting algorithm and a proof of its correctness can be found in [21].

IV. Experimental Evaluation

In this section we present the results achieved by our "symbolic" algorithm, and compare its efficiency against the "explicit" algorithm of [5] and its outcomes that have been reported in [22]. We start with some notes on implementation, and then compare the two algorithms on the model of Example 1.

a) The "explicit" algorithm of [5]: The implementation details of the algorithm are explained in [22]. It is an extension of PRISM model checker [23] employing the explicit computation engine. First, a finite discretized MDP is built using the optimizations reported in [22], and then this MDP is solved by the standard algorithms of PRISM. Currently there are three solution methods available for computing an optimal policy for total reachability cost in a finite MDP: policy iteration, value iteration, and Gauss-Seidl value iteration. The policy iteration has been identified as the fastest one.

b) Our "symbolic" algorithm: We have a prototype implementation of our "symbolic" algorithm that is also implemented as an extension of PRISM and uses the "symbolic" policy iteration method. We tested several libraries and tools for isolating real roots of polynomials (Apache Commons, Matlab, Maple, and Sage). The best performance was achieved by Maple [3], and we decided to use this software in our proof-of-concept implementation. Currently, we call Maple directly from Java, providing the polynomial and the required precision for the roots. We measure the CPU time for all Maple calls and add it to the final result.

All the computations were run on platform HP DL980 G7 with 8 64-bit processors Intel Xeon X7560 2.26GHz (together 64 cores) and 448 GiB DDR3 RAM. The time and space was measured by the Linux command `time`. The N/A result stands for out of memory exception.

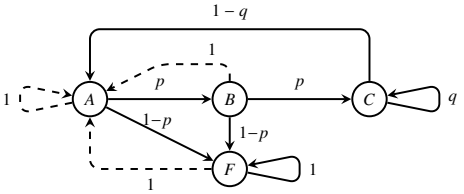

Fig. 4. A fdCTMC model of the communication with Bob_i.

Num. of Bobs	ε	Num. states	Num. roots	Max pol. degree	CPU time [s]	
					symbolic	explicit
1	10^{-2}	4	8	55	2.91	4.4
1	10^{-3}	4	8	60	2.94	11.84
1	10^{-4}	4	8	64	2.96	75.18
1	10^{-5}	4	10	69	3.01	3429.88
2	10^{-2}	32	16	122	3.65	33.00
2	10^{-3}	32	20	129	4.93	1265.45
2	10^{-4}	32	20	135	4.91	N/A
3	10^{-2}	192	30	202	6.02	1765.71
3	10^{-3}	192	31	210	7.16	N/A
3	10^{-4}	192	32	220	7.47	N/A
4	10^{-2}	1024	40	280	10.71	N/A
4	10^{-3}	1024	40	290	11.41	N/A
5	10^{-2}	5120	55	360	26.36	N/A
6	10^{-2}	24576	65	449	221.76	N/A

TABLE I
Performance characteristics.

Recall the model of Example 1 where Alice is communicating with $\text{Bob}_1, \ldots, \text{Bob}_n$. The communication with Bob_i is modeled as the fdCTMC of Fig. 4. So, the only difference from the fdCTMC of Fig. 1 is that now we also model the possibility of "breaking" an already established connection. We set $p = q = 0.9$, the rate costs are equal to 1, all fixed-delay transition incur the impulse cost 1, and the exp-delay transitions incur zero cost.

The whole protocol is modeled as a fdCTMC obtained by constructing the "parallel composition" of n identical copies of the fdCTMC of Fig. 4 (i.e., we assume that all Bobs use the same type of communication channel). The current state of this parallel composition is given by the n-tuple of current states of all components. In particular, the initial state is $(A, \ldots, A)$, and the only target state is $(C, \ldots, C)$. Obviously, the number of states grows exponentially with n.

Table I shows the outcomes achieved by the "explicit" and the "symbolic" algorithm. The first column gives the number of Bobs involved in the protocol, the second column is the error ε, the third columns specifies the total number of states of the resulting fdCTMC model, the fourth and the fifth column specify the maximal number of roots and the maximal degree of the constructed polynomials in the "symbolic" algorithm, and the last two columns give the time needed to compute the results. Note that the "explicit" algorithm cannot analyze a protocol with more that three Bobs, and tends to be significantly worse especially for smaller ε.

Let us note that the "symbolic" algorithm could handle even larger instances, but we cannot provide such results with our current experimental implementation because of the limitation of the double precision in floating types (we would need a higher precision).

V. Related work

The relationship to the work of [5] was already explained in Section III. In particular, we use the discretization constants developed in [5] to define the sets $Dval(s)$ (see Section III).

Our fdCTMC formalism can be seen as a subclass of deterministic and stochastic Petri nets [25]. The main restriction is that in fdCTMC, at most one fixed-delay event can be enabled at a time (i.e., we cannot have two different "alarm clocks" turned on simultaneously). fdCTMC can also be seen as a special variant of Markov regenerative processes [2]. Another related formalism are delayed CTMC introduced in [14]. Fixed-day events were used to model, e.g., deterministic durations in train control systems [35], time of server rejuvenation [13], timeouts in power management systems [31]. Some of these models contain specific impulse or rate costs.

To the best of our knowledge, no generic framework for fixed-delay synthesis in stochastic continuous-time systems has been developed so far. In previous works, some special cases were considered, e.g., timeout synthesis in finite models [9], [33], history dependent timeouts [24], [18], or timeout synthesis for a specific concrete model [34].

There is a number of papers on synthesizing other parameters of continuous-time models, such as parametric timed automata [1], parametric one-counter automata [15], parametric Markov models [16], etc. In the context of continuous-time stochastic systems, the synthesis of appropriate rates in CTMC was studied in [17], [19], [10]. In [17], a symbolic technique similar to ours is used to synthesize optimal rates in CTMC, but the results are not directly applicable in our setting due to the difference in objectives and possible cycles in the structure of fdCTMC. In [26], [8], [7], [4] the optimal controller synthesis for continuous-time (Semi)-Markov decision processes is studied, which can be also seen as a synthesis problem for *discrete* parameters in continuous-time systems (contrary to our result, the schedulers are only allowed to choose actions from a priori discrete and finite domains).

References

[1] R. Alur, T.A. Henzinger, and M.Y. Vardi. Parametric real-time reasoning. In *STOC*, pages 592–601. ACM, 1993.

[2] E.G. Amparore, P. Buchholz, and S. Donatelli. A structured solution approach for Markov regenerative processes. In *QEST*, volume 8657 of *LNCS*, pages 9–24. Springer, 2014.

[3] L. Bernardin et al. Maple 16 Programming Guide, 2012.

[4] T. Brázdil, V. Forejt, J. Krčál, J. Křetínský, and A. Kučera. Continuous-time stochastic games with time-bounded reachability. *Inf. Comput.*, 224:46–70, 2013.

[5] T. Brázdil, Ľ. Korenčiak, J. Krčál, P. Novotný, and V. Řehák. Optimizing performance of continuous-time stochastic systems using timeout synthesis. In *QEST*, volume 9259 of *LNCS*, pages 141–159. Springer, 2015.

[6] T. Brázdil, Ľ. Korenčiak, J. Krčál, P. Novotný, and V. Řehák. Optimizing performance of continuous-time stochastic systems using timeout synthesis. *CoRR*, abs/1407.4777, 2016.

[7] T. Brázdil, J. Krčál, J. Křetínský, A. Kučera, and V. Řehák. Stochastic real-time games with qualitative timed automata objectives. In *CONCUR*, volume 6269 of *LNCS*, pages 207–221. Springer, 2010.

[8] P. Buchholz, E.M. Hahn, H. Hermanns, and L. Zhang. Model checking algorithms for CTMDPs. In *CAV*, volume 6806 of *LNCS*, pages 225–242. Springer, 2011.

[9] L. Carnevali, L. Ridi, and E. Vicario. A quantitative approach to input generation in real-time testing of stochastic systems. *IEEE Trans. Softw. Eng.*, 39(3):292–304, 2013.

[10] M. Češka, F. Dannenberg, M. Kwiatkowska, and N. Paoletti. Precise parameter synthesis for stochastic biochemical systems. In *CMSB*, volume 8859 of *LNCS*, pages 86–98. Springer, 2014.

[11] M. Fackrell. Fitting with matrix-exponential distributions. *Stochastic models*, 21(2-3):377–400, 2005.

[12] J. Fearnley. Exponential lower bounds for policy iteration. In *ICALP, Part II*, volume 6199 of *LNCS*, pages 551–562. Springer, 2010.

[13] R. German. *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*. Wiley, 2000.

[14] C. Guet, A. Gupta, T. Henzinger, T. Mateescu, and A. Sezgin. Delayed continuous-time Markov chains for genetic regulatory circuits. In *CAV*, volume 7358 of *LNCS*, pages 294–309. Springer, 2012.

[15] C. Haase, S. Kreutzer, J. Ouaknine, and J. Worrell. Reachability in succinct and parametric one-counter automata. In *CONCUR*, volume 5710 of *LNCS*, pages 369–383. Springer, 2009.

[16] E.M. Hahn, H. Hermanns, and L. Zhang. Probabilistic reachability for parametric markov models. *STTT*, 13(1):3–19, 2011.

[17] T. Han, J.P. Katoen, and A. Mereacre. Approximate parameter synthesis for probabilistic time-bounded reachability. In *Real-Time Systems Symposium*, pages 173–182. IEEE, 2008.

[18] P.G. Jensen and J.H. Taankvist. Learning optimal scheduling for time uncertain settings. Student project, Aalborg University, 2014.

[19] S.K. Jha and C.J. Langmead. Synthesis and infeasibility analysis for stochastic models of biochemical systems using statistical model checking and abstraction refinement. *TCS*, 412(21):2162–2187, 2011.

[20] Ľ. Korenčiak, J. Krčál, and V. Řehák. Dealing with zero density using piecewise phase-type approximation. In *EPEW*, volume 8721 of *LNCS*, pages 119–134. Springer, 2014.

[21] Ľ. Korenčiak, A. Kučera, and V. Řehák. Efficient timeout synthesis in fixed-delay ctmc using policy iteration. *CoRR*, abs/1607.00372, 2016.

[22] Ľ. Korenčiak, V. Řehák, and A. Farmadin. Extension of PRISM by synthesis of optimal timeouts in fdCTMC. In *iFM*, volume 9681 of *LNCS*, pages 130–138. Springer, 2016.

[23] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[24] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In *CONCUR*, volume 1877 of *LNCS*, pages 123–137. Springer, 2000.

[25] M.A. Marsan and G. Chiola. On Petri nets with deterministic and exponentially distributed firing times. In *Advances in Petri Nets*, pages 132–145. Springer, 1987.

[26] M.R. Neuhäusser and L. Zhang. Time-bounded reachability probabilities in continuous-time Markov decision processes. In *QEST*, pages 209–218. IEEE, 2010.

[27] M.F. Neuts. *Matrix-geometric Solutions in Stochastic Models: An Algorithmic Approach*. Courier Dover Publications, 1981.

[28] J.R. Norris. *Markov Chains*. Cambridge University Press, 1998.

[29] R. Obermaisser. *Time-Triggered Communication*. CRC Press, 1st edition, 2011.

[30] M.L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[31] Q. Qiu, Q. Wu, and M. Pedram. Stochastic modeling of a power-managed system: construction and optimization. In *ISLPED*, pages 194–199. ACM Press, 1999.

[32] K. Ramamritham and J.A Stankovic. Scheduling algorithms and operating systems support for real-time systems. *Proceedings of the IEEE*, 82(1):55–67, 1994.

[33] N. Wolovick, P. R. D'Argenio, and H. Qu. Optimizing probabilities of real-time test case execution. In *ICST*, pages 446–455. IEEE, 2009.

[34] W. Xie, H. Sun, Y. Cao, and K. S. Trivedi. Optimal webserver session timeout settings for web users. In *Computer Measurement Group Conferenceries*, pages 799–820, 2002.

[35] A. Zimmermann. Applied restart estimation of general reward measures. In *RESIM*, pages 196–204, 2006.