Counterexamples for Expected Rewards

Tim Quatmann[1], Nils Jansen[1], Christian Dehnert[1], Ralf Wimmer[2],
Erika Ábrahám[1], Joost-Pieter Katoen[1], and Bernd Becker[2]

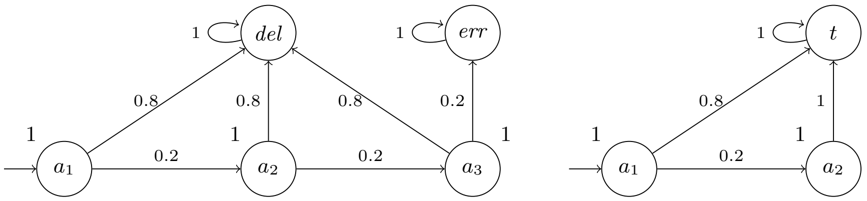
[1] RWTH Aachen University, Germany
[2] Albert-Ludwigs-Universität Freiburg, Germany

Abstract. The computation of counterexamples for probabilistic systems has gained a lot of attention during the last few years. All of the proposed methods focus on the situation when the probabilities of certain events are too high. In this paper we investigate how counterexamples for properties concerning *expected costs* (or, equivalently, expected rewards) of events can be computed. We propose methods to extract a minimal subsystem which already leads to costs beyond the allowed bound. Besides these exact methods, we present heuristic approaches based on path search and on best-first search, which are applicable to very large systems when deriving a minimum subsystem becomes infeasible due to the system size. Experiments show that we can compute counterexamples for systems with millions of states.

1 Introduction

Probabilistic model checking. Model checking is a well-established verification technique used in software and hardware industry. One of its key features is the ability to generate *counterexamples* in case the property is refuted [1]. Probabilistic model checkers such as PRISM [2] aim at verifying models that incorporate randomness. Successful applications include randomized distributed algorithms, hardware [3], security [4], and systems biology [5]. Properties typically quantify the likelihood of reachability objectives such as "Is the probability that the protocol successfully terminates at least 0.99?". Probabilistic model checking has recently been identified as one of the three main new avenues in verification [6].

Rewards. This paper focuses on the treatment of *resource consumption* in probabilistic model checking. In addition to probabilistic reachability this allows to consider the cost – measured in terms of units of used resources – of reaching a certain set of states. Such costs can be used to keep track of memory consumption, battery usage, and heat generation, to mention a few. Treating resource consumption in verification models has resulted in extensions of timed automata with "prices" [7], games with "energy" [8] and "battery" transition systems [9]. We consider discrete-time Markov chains (DTMCs, for short) that are extended with *rewards*. These so-called Markov reward models (MRMs) [10] are pivotal in the field of *performability*, i. e., the interdependent analysis of reliability and performance of systems, as stressed in [11].



(a) MRM of a simple communication protocol. (b) Critical subsystem.

Fig. 1. Example MRM and critical subsystem.

Topic of this paper. The verification of MRMs is well-developed since more than a decade [12], and is efficiently supported by tools like PRISM. Key performability questions that can be handled are of the form "Is the expected number of steps until termination at most ten?". Although such questions can be handled efficiently (and symbolically), the feedback in case such questions are violated is limited. Typically, only the expected resource consumption is provided, but no indication is provided about the cause of property violation. This paper attempts to fill this gap by providing several algorithms to generate counterexamples to expected cost properties. That is to say, we present automated means that yield diagnostic information in case the expected accumulated resource consumption exceeds an a priori given upper bound. These counterexamples are fragments of the MRM under consideration – so-called *critical subsystems* – that already violate the expected cost property. This paper thus extends the current facilities for counterexample generation for probabilistic reachability (and ω-regular) properties, cf. a recent survey [13], with expected cost properties. We consider two possible types of counterexamples which are both natural extensions of the ones for reachability: a critical part of the original MRM that is computed *narrowing down the faulty behavior* or the MRM in which the reward of irrelevant states is set to zero while *the probabilistic behavior of the system is preserved*.

An example. Let us illustrate this by means of a small example. Figure 1(a) presents a model of a simple communication protocol. Up to three times a message is being sent (states a_1, a_2, and a_3). The loss probability of a message is 0.2. The protocol terminates in state *del* when the message is successfully delivered or otherwise in state *err*. States a_i are equipped with reward one; all other states have reward zero. It is easy to see that the protocol refutes the property "the expected number of steps until termination is below 1.2". A fragment of the protocol model already violating the property is given in Figure 1(b).

Approach and related work. Our approach is based on extending the notion of *critical subsystems*, which were introduced in [14, 15] to expected cost properties. This gives the first direct method for counterexamples against expected cost properties. As explained above, two types of such subsystems are considered. To compute these counterexamples, we present three different techniques. The first one is an encoding of critical subsystems by means of *mixed-integer linear programming*

(MILP). Together with different optimization functions, *minimal* counterexamples can be obtained using standard MILP solvers such as Gurobi [16]. This approach is applicable to both types of counterexamples. The second algorithm is based on path searching algorithms. Intuitively, a subsystem is incrementally built by connecting path fragments of high costs with respect to their probability. This extends an approach presented in [15]. The last approach exploits *best-first* (bf) search which is in fact an on-the-fly exploration the MRM's state space [14]. The last two methods are applicable to critical subsystems and strongly depend on an appropriate *value function* that takes both the rewards and the path probabilities into account; we will discuss several different options for such functions. Note that all approaches on counterexample generation in the probabilistic setting suffer from the fact that the verification process—mostly based on the solving of linear equation systems—does not incorporate the computation of counterexamples as a by-product, see for instance [13]. We have implemented all our approaches and compare their applicability on several PRISM benchmarks. The conducted experiments show that the MILP approaches often yield a (nearly) minimal critical subsystem in just a few seconds, whereas the best-first search approach scales to models of 10^7 states and 10^8 transitions while yielding larger results than the path search.

2 Preliminaries

In this section we introduce the foundations needed for our methods.

Definition 1 (Discrete-time Markov Chain). *A* discrete-time Markov chain (DTMC) *is a tuple $\mathcal{D} = (S, s_I, P)$ with a finite set of states S, an initial state $s_I \in S$, and a transition probability matrix $P\colon S \times S \to [0,1] \subseteq \mathbb{R}$ with $\sum_{s' \in S} P(s,s') = 1$ for all $s \in S$.*[3]

Assume a DTMC $\mathcal{D}$. The *graph* of $\mathcal{D}$ is given by $\mathcal{G}_{\mathcal{D}} = (S, E)$ where $(s,s') \in E \Leftrightarrow P(s,s') > 0$. E is called the set of *transitions*. A state $s \in S$ is *absorbing* iff $P(s,s) = 1$.

A *path* of a DTMC $\mathcal{D}$ is a non-empty (finite or infinite) sequence $\pi = s_0 s_1 \ldots$ of states $s_i \in S$ such that $P(s_i, s_{i+1}) > 0$ for all i. Let $\mathrm{Paths}^{\mathcal{D}}_{\mathit{fin}}$ denote all finite paths of $\mathcal{D}$ and $\mathrm{Paths}^{\mathcal{D}}_{\mathit{fin}}(s)$ those starting in $s \in S$. For a set $T \subseteq S$, we denote the set of finite paths starting in s and ending in the first visit of some $t \in T$ by $\mathrm{Paths}^{\mathcal{D}}_{\mathit{fin}}(s, \Diamond T) = \{s_0 \ldots s_n \in \mathrm{Paths}^{\mathcal{D}}_{\mathit{fin}}(s) \mid s_n \in T \text{ and } s_i \notin T \text{ for all } i < n\}$. A state $s' \in S$ is *reachable* from s iff $\mathrm{Paths}^{\mathcal{D}}_{\mathit{fin}}(s, \Diamond\{s'\}) \neq \emptyset$. Let $\mathrm{Paths}^{\mathcal{D}}_{\mathit{fin}}(S', S'')$ for $S', S'' \subseteq S$ denote the set of paths starting in a state from S' and ending in a state from S'' without visiting a state from $S' \cup S''$ in between.

Let $\pi = s_0 s_1 \ldots s_n \in \mathrm{Paths}^{\mathcal{D}}_{\mathit{fin}}$ be a finite path. Its probability is given by $P(\pi) = \prod_{i=0}^{n-1} P(s_i, s_{i+1})$. Consider a state $s \in S$ and a set of dedicated *target*

[3] Note that for our methods we assume probabilities and rewards to be from $\mathbb{Q}$.

states $T \subseteq S$. The *reachability probability*, i. e., the probability to eventually reach a state $t \in T$ when starting in s is given by

$$\Pr^{\mathcal{D}}(s \models \lozenge T) := \sum_{\pi \in \mathrm{Paths}^{\mathcal{D}}_{\mathit{fin}}(s, \lozenge T)} P(\pi).$$

Note that no path $\pi \in \mathrm{Paths}^{\mathcal{D}}_{\mathit{fin}}(s, \lozenge T)$ is a proper prefix of another path $\pi' \in \mathrm{Paths}^{\mathcal{D}}_{\mathit{fin}}(s, \lozenge T)$ as these paths end at the first visit of a state in $T \subseteq S$. Therefore we can take the sum of their probabilities to obtain the reachability probability.

Definition 2 (Markov Reward Model). *A* Markov reward model (MRM) *is a tuple* $\mathcal{M} = (\mathcal{D}, \mathrm{rew})$ *with the* underlying DTMC $\mathcal{D} = (S, s_I, P)$ *and the* reward function $\mathrm{rew}\colon S \to \mathbb{R}_{\geq 0}$.

Note that for our applications only rational numbers are used as rewards. The presented notions for DTMCs are also applicable to MRMs and refer to the underlying DTMC. For instance, $\mathrm{Paths}^{\mathcal{M}}_{\mathit{fin}}$ refers to paths in the DTMC $\mathcal{D}$ of the MRM $\mathcal{M} = (\mathcal{D}, \mathrm{rew})$. Intuitively, the reward $\mathrm{rew}(s)$ is earned on leaving the state $s \in S$. The *(cumulative) reward* of a finite path $\pi = s_0 \ldots s_n \in \mathrm{Paths}^{\mathcal{M}}_{\mathit{fin}}$ is given by $\mathrm{rew}^{\mathcal{M}}(\pi) = \sum_{i=0}^{n-1} \mathrm{rew}(s_i)$. The *expected reward* is the expected amount of reward that has been accumulated until a set of target states $T \subseteq S$ is reached when starting in a state s. If $\Pr^{\mathcal{D}}(s \models \lozenge T) < 1$, we follow the usual definition and set $\mathrm{ExpRew}^{\mathcal{M}}(s \models \lozenge T) := \infty$.[4] Otherwise we define

$$\mathrm{ExpRew}^{\mathcal{M}}(s \models \lozenge T) := \sum_{\pi \in \mathrm{Paths}^{\mathcal{D}}_{\mathit{fin}}(s, \lozenge T)} P(\pi) \cdot \mathrm{rew}(\pi)\,.$$

For all notations, we will in the following omit the superscript $\mathcal{M}$ (or $\mathcal{D}$) if it is clear from the context. Note that rewards can also be defined for transitions. These transition rewards can be transformed to state rewards by means of a simple transformation.

Definition 3 (Reachability Property, Expected Reward Property). *For a DTMC* $\mathcal{D} = (S, s_I, P)$ *with* $s \in S$ *and* $T \subseteq S$*, a probability bound* $\lambda \in [0, 1]$*, and a comparison operator* $\triangleleft \in \{<, \leq\}$*, the* reachability property $\mathbb{P}_{\triangleleft\lambda}(\lozenge T)$ *is satisfied in s, written* $s \models \mathbb{P}_{\triangleleft\lambda}(\lozenge T)$*, iff* $\Pr(s \models \lozenge T) \triangleleft \lambda$.

Given an MRM $\mathcal{M} = (\mathcal{D}, \mathrm{rew})$ *and a reward bound* $\lambda' \in \mathbb{R}_{\geq 0}$*, the* expected reward property $\mathbb{E}_{\triangleleft\lambda'}(\lozenge T)$ *is satisfied in s, denoted by* $s \models \mathbb{E}_{\triangleleft\lambda'}(\lozenge T)$*, iff* $\mathrm{ExpRew}(s \models \lozenge T) \triangleleft \lambda'$.

Let $\mathcal{D} \models \mathbb{P}_{\triangleleft\lambda}(\lozenge T) \Leftrightarrow s_I \models \mathbb{P}_{\triangleleft\lambda}(\lozenge T)$ *and* $\mathcal{M} \models \mathbb{E}_{\triangleleft\lambda'}(\lozenge T) \Leftrightarrow s_I \models \mathbb{E}_{\triangleleft\lambda'}(\lozenge T)$.

As transitions leaving a target state $t \in T$ do not affect reachability probabilities and expected rewards, we assume that all target states are absorbing. Note that

[4] The intuition is as follows: If a state with positive reward from which no target state is reachable is visited infinitely often, an infinite amount of reward will be collected. If this case is excluded, the definition can be generalized.

we explicitly do not include properties with lower bounds here. For reachability properties with lower bounds, we can formulate equivalent properties with upper bounds by considering the probability to reach a state from which no state $t \in T$ is reachable. This transformation is, however, not applicable for expected reward properties. The standard method to check whether $s \models \mathbb{E}_{\trianglelefteq\lambda}(\Diamond T)$ (or $s \models \mathbb{P}_{\trianglelefteq\lambda}(\Diamond T)$) for a state $s \in S$ is to solve a linear equation system. For expected rewards the equation system has the following shape (assuming $\Pr^{\mathcal{M}}(s \models \Diamond T) = 1$):

$$r_s = \begin{cases} 0 & \textit{for } s \in T, \\ \mathrm{rew}(s) + \sum_{s' \in S} P(s, s') \cdot r_{s'} & \textit{otherwise}. \end{cases} \tag{1}$$

The unique solution for r_s yields the values $\mathrm{ExpRew}(s \models \Diamond T))$ for every state $s \in S$. For more details and the corresponding linear equation system for reachability probabilities we refer to [17].

We will need mixed integer linear programs (MILPs) which optimize a linear objective function under a condition specified by a conjunction of linear inequalities. A subset of the variables in the inequalities is restricted to take only integer values, which makes solving MILPs NP-hard [18, Problem MP1].

Definition 4 (Mixed Integer Linear Program). *Let $A \in \mathbb{Q}^{m \times n}$, $B \in \mathbb{Q}^{m \times k}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, and $d \in \mathbb{Q}^k$. A* mixed integer linear program *(MILP) consists of minimizing $c^T x + d^T y$ such that $Ax + By \leq b$ and $x \in \mathbb{R}^n$, $y \in \mathbb{Z}^k$.*

MILPs are typically solved by a combination of a branch-and-bound algorithm with the generation of so-called cutting planes. These algorithms heavily rely on the fact that relaxations of MILPs which result by removing the integrality constraints can be solved efficiently. Efficient tools are available, e. g., Gurobi [16]. We refer the reader to [19] for more information on solving MILPs.

We now briefly recall the central definitions of counterexamples for reachability properties.

Definition 5 (Evidence, Counterexample [20]). *Let $\mathcal{D} = (S, s_I, P)$ be a DTMC, $T \subseteq S$ be a set of target states, and $\mathbb{P}_{\trianglelefteq\lambda}(\Diamond T)$ be a reachability property violated by $\mathcal{D}$. Paths in $\mathrm{Paths}^{\mathcal{D}}_{\mathit{fin}}(s_I, \Diamond T)$ are called* evidences. *A* counterexample *C is a set of evidences such that $P(C) := \sum_{\pi \in C} P(\pi) \ntrianglelefteq \lambda$. C is* minimal *if $|C| \leq |C'|$ holds for all counterexamples C' and* smallest *if it is minimal and $P(C) \geq P(C')$ holds for all minimal counterexamples C'.*

Smallest counterexamples can be computed using algorithms for finding the k shortest paths in a directed graph. As it is infeasible to compute smallest counterexamples for large DTMCs, numerous heuristic approaches have been proposed, based on best-first search, SAT-based bounded model checking, and BDD-based symbolic methods.

The drawback of all path-based counterexamples is that the number of paths even in a minimal counterexample can be very large; [20] presents an example where the number of paths in a minimal counterexample is doubly exponential in the system parameters – and therefore much larger than the number of system states. To obtain a more compact representation, the usage of critical subsystems has been proposed [14, 15].

Definition 6 (Selection, Critical Subsystem). *Let $\mathcal{D} = (S, s_I, P)$ be a DTMC, $T \subseteq S$ a set of target states, and $\mathbb{P}_{\triangleleft\lambda}(\lozenge T)$ a reachability property which is violated by $\mathcal{D}$. A subset $S' \subseteq S$ of states with $s_I \in S'$ is called a* selection. *The* subsystem *of $\mathcal{D}$ induced by a selection S' is the DTMC $\mathcal{D}' = (S' \uplus \{t\}, s_I, P')$ where $t \notin S$ is a new state and P' is defined by*

$$P'(s, s') = \begin{cases} P(s, s') & \text{if } s, s' \in S' \\ \sum_{s'' \in S \setminus S'} P(s, s'') & \text{if } s \in S' \text{ and } s' = t \\ 1 & \text{if } s = s' = t \\ 0 & \text{otherwise.} \end{cases}$$

We call a selection S' and its induced subsystem critical *for $\mathbb{P}_{\triangleleft\lambda}(\lozenge T)$ if $\mathbb{P}_{\triangleleft\lambda}(\lozenge T')$ is violated in the induced subsystem with $T' = T \cap S'$.*

The additional absorbing state $t \notin S$ is only required here to ensure that the probabilities of the transitions leaving a certain state sum up to one. Critical subsystems have the property that the set of paths from the initial to a target state forms a counterexample according to Definition 5.

The computation of critical subsystems with a minimum number of states using MILP has been investigated in [21]. Heuristic approaches which typically yield small, but not necessarily minimal counterexamples are also available. They are mostly based on the path-search approaches mentioned above.

3 Critical Subsystems for Expected Rewards

Consider in the following an MRM $\mathcal{M} = (\mathcal{D}, \mathrm{rew})$ with $\mathcal{D} = (S, s_I, P)$ and an expected reward property $\mathbb{E}_{\triangleleft\lambda}(\lozenge T)$ with $\triangleleft \in \{<, \leq\}$, $\lambda \in \mathbb{R}_{\geq 0}$ such that $\mathcal{M} \not\models \mathbb{E}_{\triangleleft\lambda}(\lozenge T)$. We assume all target states $t \in T$ to be absorbing.

If $\Pr(s_I \models \lozenge T) < 1$, the expected reward is infinite, and it directly follows that $\mathbb{E}_{\triangleleft\lambda}(\lozenge T)$ is violated for all $\lambda \in \mathbb{R}_{\geq 0}$. In this case, a path from s_I to a state from which no state in T is reachable indicates why $\Pr(s_I \models \lozenge T) < 1$ holds and can therefore serve as a counterexample. Such a path can be found via simple reachability analysis on the graph of the DTMC.

In the following, we will only consider the more interesting case where $\Pr(s_I \models \lozenge T) = 1$ and thus $\mathrm{ExpRew}(s_I \models \lozenge T) < \infty$ holds.

To indicate why the MRM $\mathcal{M}$ violates the property $\mathbb{E}_{\triangleleft\lambda}(\lozenge T)$, we adjust the notion of critical subsystems for reachability properties to expected reward properties. The idea is to select states of the original system in order to form a subsystem that already refutes the property.

Definition 7 (Critical Subsystem for Expected Reachability). *Let $\mathcal{M} = (\mathcal{D}, \mathrm{rew})$ be an MRM with $\mathcal{D} = (S, s_I, P)$. For a selection of states $S' \subseteq S$, a* subsystem *of $\mathcal{M}$ is given by the MRM $\mathcal{M}' = (\mathcal{D}', \mathrm{rew}')$ with $\mathcal{D}' = (S' \uplus \{t\}, s_I, P')$*

where $t \notin S$ is a new state, $\mathrm{rew}'(t) = 0$*,* $\mathrm{rew}'(s) = \mathrm{rew}(s)$ *for all $s \in S'$ and P' as in Definition 6. The subsystem $\mathcal{M}'$ is* critical *for a property $\mathbb{E}_{\lhd\lambda}(\Diamond T)$ iff $\mathcal{M}' \not\models \mathbb{E}_{\lhd\lambda}(\Diamond T')$ where $T' = (T \cap S') \cup \{t\}$.*

In contrast to critical subsystems for reachability properties, the new absorbing state t has to be considered as a target state. This is reasonable because in the original MRM the probability to reach a target state is one by assumption. If t were not a target state, the probability to reach a target state in the subsystem would be less than one and therefore the expected reward infinite. That means, the subsystem would be critical for every bound $\lambda \in \mathbb{R}_{\geq 0}$, even if $\mathcal{M} \models \mathbb{E}_{\lhd\lambda}(\Diamond T)$. Such a subsystem can obviously not be considered a counterexample. The definition above ensures that $\mathrm{Pr}^{\mathcal{M}'}(s_I \models \Diamond T') = 1$ holds for every possible subsystem $\mathcal{M}'$ of $\mathcal{M}$ by adding the new state t to the set of target states. From $\mathrm{Pr}^{\mathcal{M}}(s_I \models \Diamond T) = 1$, it also follows that $\mathrm{Pr}^{\mathcal{M}}(s \models \Diamond T) = 1$ holds for every state s that lies on a path in $\mathrm{Paths}^{\mathcal{M}}_{\mathit{fin}}(s_I, \Diamond T)$.

This means that the reachability of target states is already given by assumption. In a subsystem $\mathcal{M}'$, the transitions leading to t can be interpreted as "shortcuts" to a target state. If $\mathcal{M}'$ is critical, then the reward collected within the subsystem is already too large, even if all other states would have reward zero. This allows us to hide the unimportant details of how a state in T will eventually be reached and we can focus on the parts of the model where reward is collected. Note that this reasoning would not be valid if we allow for states with negative rewards: Parts of the model where enough reward is collected might be followed by states with negative rewards. Hence, the details of how a state in T is reached would not be unimportant anymore. A critical subsystem can also not serve as a counterexample for an expected reward property with a *lower* bound as this would require to indicate an upper bound of the expected reward of the model.

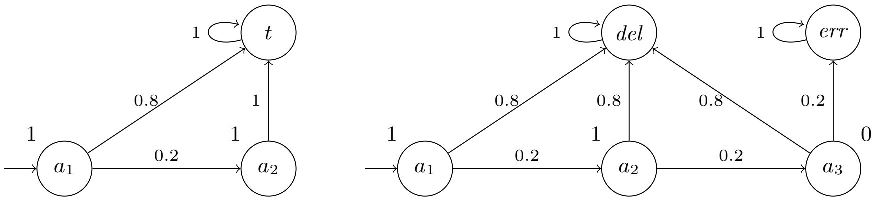
Alternative Definition of Critical Subsystems. We discuss another notion for critical subsystem for expected reward properties. According to Definition 7, the original system was restricted with respect to its states. It is also natural to change the rewards that are assigned to states *without changing the behavior of the system*. The goal is now to only restrict the positive rewards in the system as much as possible. To avoid confusion, this is called a *critical reward subsystem.*

Definition 8 (Critical Reward Subsystem). *Let $\mathcal{M} = (\mathcal{D}, \mathrm{rew})$ be an MRM with $\mathcal{D} = (S, s_I, P)$. For a selection $S' \subseteq S$, the* reward subsystem *of $\mathcal{M}$ induced by S' is defined as the MRM $\mathcal{M}' = (\mathcal{D}, rew')$ where*

$$rew'(s) = \begin{cases} rew(s) & \text{for } s \in S' \\ 0 & \text{otherwise.} \end{cases}$$

The reward subsystem $\mathcal{M}'$ is critical *for a property $\mathbb{E}_{\lhd\lambda}(\Diamond T)$ iff $\mathcal{M}' \not\models \mathbb{E}_{\lhd\lambda}(\Diamond T)$.*

A critical reward subsystem indicates the parts of the system that give enough reward to refute a property without ruling out possible system behavior (i. e.,



(a) Critical subsystem

(b) Critical reward subsystem

Fig. 2. The different notions of critical subsystems.

without disregarding states in the underlying DTMC). Depending on the particular application, this can be advantageous, although there are examples where a critical subsystem according to Definition 7 should be preferred. For instance, consider an MRM $\mathcal{M}$ with exactly one state s such that $rew(s) > 0$. Every reward subsystem of $\mathcal{M}$ is either equal to $\mathcal{M}$ or only has states with reward zero and is therefore less useful for debugging purposes.

In conclusion, we reconsider the MRM depicted in Figure 1(a). For the violated property $\mathbb{E}_{<1.2}(\lozenge\{del, err\})$, we show the two notions of critical subsystems in Figure 2. The critical subsystem according to Definition 7 in Figure 2(a) has reduced state space, while for the critical reward subsystem according to Definition 8 in Figure 2(b), a reward of zero is assigned to state a_3.

4 Generation of Critical Subsystems

We now present different approaches to generate a subsystem of the MRM $\mathcal{M} = (\mathcal{D}, \mathrm{rew})$, $\mathcal{D} = (S, s_I, P)$, that is critical for the property $\mathbb{E}_{<\lambda}(\lozenge T)$[5]. We assume in the following that $\mathcal{M}$ violates the property, that $\mathrm{Pr}^{\mathcal{M}}(s \models \lozenge T) = 1$, that the target states in T are absorbing, that all states in S are reachable from the initial state $s_I \notin T$, and that $\mathrm{ExpRew}^{\mathcal{M}}(s_I \models \lozenge T) > 0$.

4.1 Minimal Critical Subsystem Generation

We start with the problem of generating *minimal critical subsystems*. A critical subsystem is called *minimal* if it is induced by a selection $S' \subseteq S$ such that $|S'| \leq |S''|$ holds for all selections $S'' \subseteq S$ that induce critical subsystems. We fix the set of states with positive reward by $S_R \coloneqq \{s \in S \mid \mathrm{rew}(s) > 0\}$. To reduce the number of states that have to be considered, the first step is to determine the set of *contributing states*, which are given by

$$\widehat{S} = \{s \in S \setminus T \mid \text{ a state } s' \in S_R \text{ is reachable from } s\}\,.$$

$\widehat{S}$ can be obtained via a reachability analysis in the underlying graph of the MRM. Starting from a non-contributing state $s \in S \setminus \widehat{S}$, a state with positive

[5] the subsequently discussed Path search and Best-first search approaches are also applicable for $\mathbb{E}_{\leq\lambda}(\lozenge T)$.

reward cannot be reached. Therefore, adding s to a selection S' does not have any effect on the expected reward of the subsystem induced by S' and minimal critical subsystems contain only contributing states.

In a second step we formulate an MILP to find a selection $S' \subseteq \widehat{S}$ that induces a minimal critical subsystem $\mathcal{M}'$ of $\mathcal{M}$ for $\mathbb{E}_{<\lambda}(\lozenge T)$. For all $s \in \widehat{S}$, variables $x_s \in \{0, 1\}$ are used with the interpretation that $x_s = 1$ iff $s \in S'$. Furthermore, variables $r_s \in \mathbb{R}_{\geq 0}$ are used to take into account the expected reward for the states in the resulting subsystem, more precisely, $0 \leq r_s \leq \mathrm{ExpRew}^{\mathcal{M}'}(s \models \lozenge T')$ with $T' = (T \cap S') \cup \{t\}$. The MILP can be formulated as follows:

$$\text{minimize} \qquad -\frac{1}{2 \cdot \mathrm{ExpRew}^{\mathcal{M}}(s_I \models \lozenge T)} \cdot r_{s_I} + \sum_{s \in \widehat{S}} x_s \tag{2a}$$

such that

$$\forall s \in \widehat{S}\colon \quad r_s \leq (\mathrm{ExpRew}^{\mathcal{M}}(s \models \lozenge T)) \cdot x_s \tag{2b}$$

$$\forall s \in \widehat{S}\colon \quad r_s \leq \mathrm{rew}(s) + \sum_{s' \in \widehat{S}} P(s, s') \cdot r_{s'} \tag{2c}$$

$$r_{s_I} \geq \lambda \tag{2d}$$

We can obtain the values $\mathrm{ExpRew}^{\mathcal{M}}(s \models \lozenge T)$ for all $s \in \widehat{S}$ as a side-product from model checking. Condition 2b ensures that, if the state is not included, i. e., $x_s = 0$, r_s is explicitly set to zero in order to avoid unwanted contribution to the expected reward of the initial state in the subsystem. For states in the selected subsystem, i. e., $x_s = 1$, Condition 2b is not a real restriction as $\mathrm{ExpRew}^{\mathcal{M}'}(s \models \lozenge T') \leq \mathrm{ExpRew}^{\mathcal{M}}(s \models \lozenge T)$ holds for all states $s \in S'$. In Constraint 2c, the value of r_s is bounded from above by the actual expected reward in the subsystem by using the linear equation system as in Equation 1. Transitions that lead to states in $S \backslash \widehat{S}$ are not considered since the expected reward of these states is always 0. Constraint 2d ensures the criticality of the subsystem by forcing the expected reward of the initial state to be at least λ. Finally, consider the objective function in (2a), which enforces a minimal critical subsystem with maximal expected reward among all minimal critical subsystems: The second summand ensures the minimality of the critical subsystem by minimizing the sum of all x_s-variables. The first summand ensures a maximal value for the r_{s_I}-variable of the initial state by minimizing its negative value. Additionally, this value needs to be in the open interval $(0, 1)$ as otherwise the solver could include another state s, i. e., $x_s = 1$, and thereby break the minimality criterion. This is achieved by the factor $c \cdot {}^{1}\!/_{\mathrm{ExpRew}^{\mathcal{M}}(s_I \models \lozenge T)}$ for arbitrary $0 < c < 1$ (we chose $c = {}^{1}\!/_{2}$). This maximal value will be exactly the expected reward of the initial state in the minimal critical subsystem. Note that this is not necessary in order to achieve a state-minimal subsystem. In our experiments as well as in the following objective functions, we omit this summand.

Redundant constraints which prune sub-optimal solutions from the search space can be added to this MILP to assist the solver in finding an optimal solution quickly. We omit these constraints here as they have a very similar shape as for reachability properties. We refer the reader to [21] for details.

Alternative Objective Functions. Instead of a critical subsystem with a minimal number of states, other notions might be beneficial dependent on the application at hand. We discuss a few possibilities below to give an intuition on how our approach can be adapted in the desired way. For instance, we replace (2a) with one of the following objective functions:

$$\text{minimize} \qquad \sum_{s \in S_R} x_s \quad (3) \qquad \sum_{s \in S_R} \text{rew}(s) \cdot x_s \quad (4) \qquad \sum_{s \in S_R} \text{rew}(s)^2 \cdot x_s \quad (5)$$

To obtain a minimal *number of selected states with positive rewards*, (3) can be used. By (4), the *sum of all rewards* occurring in the subsystem is minimized, while we minimize the *norm of the rewards* occurring in the subsystem by (5).

Critical Reward Subsystems. We also give an MILP formulation to generate minimal critical reward subsystems as in Definition 8:

$$\text{minimize} \qquad \sum_{s \in S_R} x_s \tag{6a}$$

such that

$$\forall s \in \widehat{S}\colon \quad r_s \leq \text{rew}(s) \cdot x_s + \sum_{s' \in \widehat{S}} P(s, s') \cdot r_{s'} \tag{6b}$$

$$r_{s_I} \geq \lambda \tag{6c}$$

The objective function (6a) only minimizes the number of states with positive reward from S_R. In Constraint 6b, the expected reward for each state $s \in \widehat{S}$ that is included in the selection is computed and assigned as upper bound to r_s. Constraint 6c ensures the criticality of the subsystem.

Note that in contrast to the MILP formulation (2a)–(2d), we do not need to explicitly assign non-selected states an expected reward of zero as it suffices to only set the contributing reward of non-selected states to zero.

4.2 Path Search Approach

The *path search approach* is an extension of the local path search presented in [15]. Originally, this heuristic approach is used to generate small critical subsystems of DTMCs for reachability properties. The algorithm can be adapted to work for MRMs and expected reward properties by taking the rewards into consideration.

For this purpose, a *value function* $V\colon S \times S \to [0, 1] \subseteq \mathbb{R}$ is used to evaluate the benefit of transitions. The function should take rewards and probabilities into account such that a high value $V(s, s')$ means that it might be beneficial to include the states s and s' in the subsystem. A sequence of states $s_0 \ldots s_n$ with $V(s_i, s_{i+1}) > 0$ for all $0 \leq i < n$ should be a valid path of $\mathcal{M}$ and vice versa. We therefore require $V(s, s') > 0$ iff $P(s, s') > 0$ for all $s, s' \in S$. For a path

$\pi = s_0 \ldots s_n \in \mathrm{Paths}^{\mathcal{M}}_{\mathit{fin}}$ the value of a path is given by $V(\pi) = \prod_{i=0}^{n-1} V(s_i, s_{i+1})$. Given a set of paths $\mathit{\Pi} \subseteq \mathrm{Paths}^{\mathcal{M}}_{\mathit{fin}}$, a path $\pi \in \mathit{\Pi}$ is called *most valuable* if $V(\pi) \geq V(\pi')$ for all $\pi' \in \mathit{\Pi}$. To ensure that there is always a most valuable path, we require that $0 \leq V(s, s') \leq 1$ for all $s, s' \in S$. If $V(s, s') = 1$, we additionally require $V(s, s'') = 0$ for all $s'' \neq s'$ to exclude infinitely many most valuable paths that arbitrarily often take loops with value one. Reasonable definitions for V will be discussed later.

We consider paths from $\mathrm{Paths}^{\mathcal{M}}_{\mathit{fin}}(S_{\mathrm{start}}, S_{\mathrm{end}})$ for $S_{\mathrm{start}}, S_{\mathrm{end}} \subseteq S$. Furthermore, for paths of length two we require that the last state is not contained in S_{start} (note that the sets S_{start} and S_{end} do not have to be disjoint).

The first step of the path search approach is always to find a most valuable path that starts in the initial state and ends in one of the target states. A selection S' is initialized with the states visited on that path. After that, most valuable paths connecting already selected states with target states or, again, selected states are repeatedly searched. The selection S' is extended by the states visited on these paths until it induces a critical subsystem. Algorithm 1 describes the procedure. Here, the function FindMostValuablePath$(V, S_{\mathrm{start}}, S_{\mathrm{end}})$ returns a most valuable path with respect to the value function V that connects S_{start} with S_{end}. Such a function can be realized by using an adaptation of Dijkstra's shortest path algorithm where the values are multiplied (instead of summed) and paths with maximal (instead of minimal) values are chosen. The function SubSys$(\mathcal{M}, S')$ returns the subsystem of $\mathcal{M}$ induced by S'.

Algorithm 1 Path Search Approach

```
Input: MRM $\mathcal{M}$, property $\mathbb{E}_{\trianglelefteq\lambda}(\Diamond T)$
Output: A critical subsystem of $\mathcal{M}$ for $\mathbb{E}_{\trianglelefteq\lambda}(\Diamond T)$

initialize value function $V : S \times S \to [0, 1]$
$\pi \leftarrow$ FindMostValuablePath($V, \{s_I\}, T$)
$S' \leftarrow \{s \in S \mid s$ is visited by $\pi\}$
while SubSys($\mathcal{M}, S'$) is not critical for $\mathbb{E}_{\trianglelefteq\lambda}(\Diamond T)$ do
    $\pi \leftarrow$ FindMostValuablePath($V, S', S' \cup T$)
    $S' \leftarrow S' \cup \{s \in S \mid s$ is visited by $\pi\}$
end while
return SubSys($\mathcal{M}, S'$)
```

It is not required to check in every iteration whether the current selection already induces a critical subsystem. To save computation time, the condition is checked only if at least $|S| \cdot c$ additional states have been selected since the last check (for a constant $0 < c \leq 1$).

Value functions. We propose two different value functions. First, we want to take both the probability of a transition $(s, s') \in E$ and the reward of the state s into account by using their product:

$$V_1(s, s') = P(s, s') \cdot \frac{\text{rew}(s) + \varepsilon}{\max_{s'' \in S}(\text{rew}(s'')) + \varepsilon}$$

In order to avoid a value of zero and the division by zero, we add a constant $\varepsilon > 0$ to both the numerator and the denominator. The value is scaled by the maximal occurring reward in order to ensure it to be from $[0, 1]$.

As a second proposal, we make use of the actual expected reward of all states inside the original system:

$$V_2(s, s') = \frac{\text{ExpRew}(s \models \lozenge T) + \varepsilon}{\max_{s'' \in S}(\text{ExpRew}(s'' \models \lozenge T)) + 2\varepsilon}$$

We scale these values by the maximal occurring expected reward and add constants. For the denominator we add a larger value in order to have a value which is smaller than one. Note that $V_2(s, s')$ is independent of s'. This is not disadvantageous since the value of a path will still depend on all visited states (except the last one which is either a target state or already selected). It is also possible to just use probabilities for our computations. However, in our experiments V_2 performed best for most cases.

4.3 Best-first Search Approach

The *best-first search approach* is another heuristic approach to generate critical subsystems. It is related to the extended best first search (XBF) presented in [14]. A value function $f\colon S \to \mathbb{R}$ evaluates how beneficial it is to select a given state. For the best-first search, we denote value functions with f (instead of V) to avoid confusion with the value functions of the path search approach. In contrast to XBF, where the model is explored in an on-the-fly manner, the model is analyzed in advance. The obtained information can be used for f. On the one hand, this increases the effort to get the values of the function but, on the other hand, the provided values can be more accurate.

Algorithm 2 illustrates how a critical subsystem is generated with the help of such a value function f. It uses two sets of states: S' and S_{explore}. S' is a selection to which more and more states are added until it induces a critical subsystem. The set S_{explore} always contains the states that are considered to be explored, starting with the initial state s_I. Repeatedly a state $s \in S_{\text{explore}}$ with maximal value $f(s)$ is explored. This means that s is added to the selection S', removed from S_{explore}, and all non-selected successors of s are added to S_{explore}. Target states do not need to be explored and are therefore directly added to S' and not added to S_{explore}. The procedure stops as soon as the subsystem of $\mathcal{M}$ induced by S' (denoted by $\textsc{SubSys}(\mathcal{M}, S')$) is critical. Similar to the path search approach, computation time can be saved by only checking this condition if at least $|S| \cdot c$ states have been added to S' since the last check.

Algorithm 2 Best-first Search Approach

```
Input: MRM $\mathcal{M}$, property $\mathbb{E}_{\lhd\lambda}(\Diamond T)$
Output: A critical subsystem of $\mathcal{M}$ for $\mathbb{E}_{\lhd\lambda}(\Diamond T)$

initialize function $f \colon S \to \mathbb{R}$
$S' \leftarrow \emptyset$
$S_{\text{explore}} \leftarrow \{s_I\}$
repeat
    choose $s \in S_{\text{explore}}$ with $f(s) \geq f(s')$ for all $s' \in S_{\text{explore}}$
    $S' \leftarrow S' \cup \{s\} \cup \{s' \in T \mid P(s,s') > 0\}$
    $S_{\text{explore}} \leftarrow (S_{\text{explore}} \cup \{s' \in S \mid P(s,s') > 0 \text{ and } s' \notin S'\}) \setminus \{s\}$
until SubSys$(\mathcal{M}, S')$ is critical for $\mathbb{E}_{\lhd\lambda}(\Diamond T)$
return SubSys$(\mathcal{M}, S')$
```

Possible value functions f for all $s \in S$ are:

$$f_1(s) \coloneqq \mathrm{ExpRew}(s \models \Diamond T) \qquad f_2(s) \coloneqq P(\pi^s_{s_I}) \cdot \mathrm{ExpRew}(s \models \Diamond T)$$

$$f_3(s) \coloneqq P(\pi^s_{s_I}) \cdot \max_{s' \in S}(P(\pi^{s'}_s) \cdot \mathrm{rew}(s'))$$

Here, $\pi^{s'}_s$ denotes a path from $s \in S$ to $s' \in S$ with maximal probability, i. e., $P(\pi^{s'}_s) \geq P(\pi)$ for all paths π from s to s'. The expected rewards of every state can be obtained as a side product from model checking. For a state $s \in S$, the value $f_1(s)$ provides information about the benefit of s itself as well as the "future" of s, i. e., the benefit of the states that are reachable via s. To also consider the "past" of s, f_2 uses the probability to reach s. Hence, the probability to reach a state s is estimated by the probability of a single path. The probabilities $P(\pi^s_{s_I})$ for all states $s \in S$ can be obtained by using a variant of Dijkstra's shortest path algorithm. Finally, function f_3 evaluates whether there is a state s' that is reachable from s with high probability and that has high reward.

5 Experimental Results

In this section we report on a selection of our benchmark results. We implemented all approaches presented in the previous sections in C++ using Gurobi [16] as MILP solver. All experiments were conducted on a Windows 64 bit system with a 2.66 GHz CPU and 6 GB RAM. We used the following benchmarks which are all available (partly without the reward definitions) for Prism [2].

The aim of the *crowds protocol* [22] (crowds) is to hide the identity of the sender of a message by randomly routing the message within a group of crowd members, consisting of good and bad members, the latter ones trying to collect information about the identity of a sender. The model can be scaled in the number N of *good members* and the number K of message deliveries. The time a single crowd member requires to forward or deliver a message varies between one and five time units. We consider *the expected time needed to deliver K messages*.

Table 1. Experimental results (TO> $1\,h$, MO> 6 GB).

		Critical Subsystem				Crit. Rew. Subsys.
		Minimal	Heuristic			Mininimal
		MILP	MILP	Path Search	BF Search	MILP
model	$N[-K]$	$\lvert S'\rvert$	$\lvert S'\rvert$	$\lvert S'\rvert$	$\lvert S'\rvert$	$\lvert S'\rvert$ / $\lvert S_R\rvert$
	#states	time (seconds)	time (seconds)	time (seconds)	time (seconds)	time (seconds)
	#transitions	memory (MB)	memory (MB)	memory (MB)	memory (MB)	memory (MB)
crowds	10-3	109	**109**	161	292	26 / 1 560
	6 563	13.38	≤1	**0.091**	0.16	0.25
	15 143	48	24	**9**	**9**	16
	10-6	972*	**1 293**	2 447	3 780	229 / 87 360
	352 535	TO (25.30 %)	36	79.44	**2.47**	238.72
	833 015	1 678	819	**246**	**246**	439
	15-6	1 820*	**2 478**	5 208	10 578	424 / 610 470
	2 464 168	TO (41.30 %)	241	1 573.16	**10.35**	3 270.84
	7 347 928	4 662	4 793	1 783	**1 782**	3 181
	20-6				**23 386**	
	10 633 591	MO	MO	TO	**194.72**	MO
	38 261 191				**5 567.24**	
herman	7	25	**25**	31	45	14 / 128
	128	0.39	≤1	**0.004**	**0.003**	0.06
	2 188	9	9	**5**	**5**	8
	13	228*	292	**230**	231	109 / 8 192
	8 192	TO (15.1 %)	4	1.59	**1.48**	1.65
	1 594 324	655	**103**	104	104	132
	15	360*	415	**362**	381	167 / 32 768
	32 768	TO (16.2 %)	22	20.27	**19.26**	20.38
	14 348 908	1 177	**831**	832	832	1 057
	17	542*	550	**546**	572	248 / 131 072
	131 072	TO (19.8 %)	333	**237.63**	245.48	280.65
	129 140 164	5 267	**5 506**	5 623	5 542	5 466
egl	4-8	1 319	**1 319**	1 407	1 606	330 / 668
	31 486	7.34	2	**0.15**	1.33	0.2
	31 741	32.6	28	**25**	**25**	31
	5-2	2 353	**2 353**	2 481	2 848	574 / 1 163
	33 790	375.93	4	3.42	**2.74**	0.16
	34 813	58	33	**26**	**26**	33
	7-4	86 943*	**87 016**	106 538	93 681	9 462 / 18 987
	1 654 782	TO (0.01 %)	**45**	3 504.7	82.06	9.18
	1 671 165	1 448	**1 033**	**1 033**	**1 033**	1 273
	7-8	126 716*	**126 732**		134 383	11 240 / 22 543
	3 489 790	TO (0.06 %)	126	TO	**120.13**	23.41
	3 506 173	2 172	2 172		**2 171**	2 686

The *self-stabilization protocol* (herman) [23] considers a ring of N identical processes. A configuration is called stable if there is exactly one designated process. The purpose of this protocol is to transform the system from an arbitrary configuration into a stable one. We are interested in *the expected number of steps until the system reaches a stable configuration.*

The *contract signing protocol* (egl) [24] is dedicated to fairly exchange commitments to a contract between two parties A and B. It is assumed that both parties have N pairs of secrets of length L which will be exchanged. A party has committed to a contract whenever both secrets of one of its pairs are known by the other party. We investigate the *expected number of messages* that A needs to receive in order to know a pair of B where only the messages after B knows a pair of A are considered.

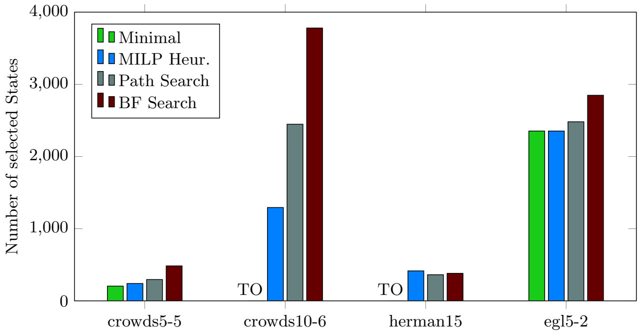


Fig. 3. Sizes of the critical subsystems generated by the different approaches.

The results for all benchmarks are depicted in Table 1. First, we computed critical subsystems using the MILP approach (see Equations 2a–2d) and the heuristic approaches (Path search and BF Search, see Sections 4.2 and 4.3). For all approaches we give the *size of the selection*, the *computation time in seconds* and the *memory consumption in MB*. The reward thresholds were set to half of the expected reward in the original system. Minimal MILP refers to the minimization including the proof of minimality. Whenever we reached the time limit, we depict the smallest critical subsystem found until that point (*) as well as the gap to the lower bound.

We also made use of the nature of MILP solving. Iteratively, an intermediate solution is compared w. r. t. its minimality to a certain lower bound on the optimal solution. This intermediate solution satisfies all conditions for a critical subsystem while minimality is not yet proven. We basically aborted the computation of Gurobi after roughly the same time as was consumed for the heuristic approaches and refer to this intermediate result as *MILP as heuristic approach*. This demonstrates the practical use of this approach as a heuristic method. The optimal results, excluding the minimization, are always depicted boldfaced.

We first observe that for the heuristic MILP a very small subsystem which is near the actual minimum is often found after a few seconds. This justifies the MILP approach also as a heuristic method, as benchmarks with millions of states and transitions are possible. Note that the herman benchmark is strongly connected having a large number of transitions. The heuristic approaches perform well for large benchmarks while the BF search is even able to compute results for over 10^7 states and 10^8 transitions. Note that the methods were not able to handle larger systems because this was the threshold for explicit storing.

We tested all value functions explained above where for Path search V_2 and for BF search f_2 performed best. For these approaches, the memory consumption is rather high due to the initial model checking that we perform. For the MILP approaches it is even higher due to the nature of the solving process.

Results for minimal critical reward subsystems as in Definition 8 depict both the size of the selection, i. e., the number of states having a positive reward in the subsystem as well as the original number of such states. We observe that in

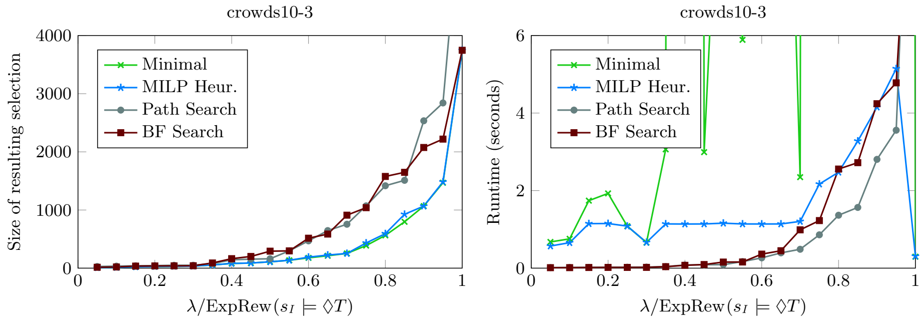


Fig. 4. Results of the different approaches plotted against different reward bounds λ.

a relatively small amount of time the number of states having positive reward can drastically be reduced; in some cases even by three orders of magnitude.

For a better overview, Figure 3 shows the sizes for different benchmarks for all approaches. In general, the path search computes smaller subsystems than the BF search. Figure 4 depicts results for a specific benchmark and different reward bounds in terms of the size of the subsystem and the running time. The nearer the reward bound is to the actual expected reward of the original system, the larger the size of the subsystems becomes. Similarly, the times required by the Path search and the BF search increase if the bound is high. The running times for computing minimal critical subsystems do not exhibit such a monotony. For certain reward bounds, the MILP solver is able to find a solution and proof its minimality comparatively fast. A notable example is the case where the ratio between the reward threshold and the actual expected reward is nearly one. Here, the MILP approaches only select the set of contributing states and therefore take nearly no time.

Summary. Using the MILP approaches we are able to compute *optimal* results for both types of critical subsystems. We note that computing critical reward subsystems is more efficient. This is due to the fact that the number of possible solutions is smaller. However, it might be beneficial to compute small counterexamples, in which case this approach is not feasible as the original system's size is not reduced. The path search yields smaller subsystems than the BF search. For very large benchmarks, BF is the only method that can compute results within the time limit.

6 Conclusion and Future Work

In this paper we thoroughly investigated different notions and methods to compute counterexamples in the form of critical system parts for Markov reward models. The experiments were very promising and showed the applicability for rather large benchmark instances. In the future, we will adapt the heuristic methods to symbolic data structures such as binary decision diagrams to enable the treatment of significantly larger systems.

References

1. Clarke, E.M.: The birth of model checking. In: 25 Years of Model Checking – History, Achievements, Perspectives. Volume 5000 of LNCS. Springer (2008) 1–26
2. Kwiatkowska, M., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: Proc. of CAV. Volume 6806 of LNCS, Springer (2011) 585–591
3. Norman, G., Parker, D., Kwiatkowska, M.Z., Shukla, S.K.: Evaluating the reliability of NAND multiplexing with PRISM. IEEE Trans. on CAD of Integrated Circuits and Systems **24**(10) (2005) 1629–1637
4. Norman, G., Shmatikov, V.: Analysis of probabilistic contract signing. Journal of Computer Security **14**(6) (2006) 561–589
5. Kwiatkowska, M.Z., Norman, G., Parker, D.: Using probabilistic model checking in systems biology. SIGMETRICS Performance Evaluation Review **35**(4) (2008) 14–21
6. Alur, R., Henzinger, T., Vardi, M.: Theory in practice for system design and verification. ACM Siglog News **2**(1) (2015) 46–51
7. Behrmann, G., Larsen, K.G., Rasmussen, J.I.: Priced timed automata: Algorithms and applications. In: Formal Methods for Components and Objects. Volume 3657 of LNCS, Springer (2005) 162–182
8. Chatterjee, K., Doyen, L., Henzinger, T.A., Raskin, J.F.: Generalized Mean-payoff and Energy Games. In: Proc. of FSTTCS. Volume 8 of LIPIcs, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2010) 505–516
9. Boker, U., Henzinger, T.A., Radhakrishna, A.: Battery transition systems. In: Proc. of POPL, ACM Press (2014) 595–606
10. Howard, R.A.: Dynamic Probabilistic Systems; Volume I: Markov models. John Wiley & Sons (1971)
11. Baier, C., Hahn, E.M., Haverkort, B.R., Hermanns, H., Katoen, J.P.: Model checking for performability. Mathematical Structures in Computer Science **23**(4) (2013) 751–795
12. Andova, S., Hermanns, H., Katoen, J.P.: Discrete-time rewards model-checked. In: Proc. of FORMATS. Volume 2791 of LNCS, Springer (2003) 88–104
13. Ábrahám, E., Becker, B., Dehnert, C., Jansen, N., Katoen, J.P., Wimmer, R.: Counterexample generation for discrete-time Markov models: An introductory survey. In: Proc. of SFM. Volume 8483 of LNCS, Springer (2014) 65–121
14. Aljazzar, H., Leue, S.: Directed explicit state-space search in the generation of counterexamples for stochastic model checking. IEEE Trans. on Software Engineering **36**(1) (2010) 37–60
15. Jansen, N., Ábrahám, E., Katelaan, J., Wimmer, R., Katoen, J.P., Becker, B.: Hierarchical counterexamples for discrete-time Markov chains. In: Proc. of ATVA. Volume 6996 of LNCS, Springer (2011) 443–452
16. Gurobi Optimization, Inc.: Gurobi optimizer reference manual. `http://www.gurobi.com` (2013)
17. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (2008)
18. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co Ltd (1979)
19. Schrijver, A.: Theory of Linear and Integer Programming. Wiley (1986)
20. Han, T., Katoen, J.P., Damman, B.: Counterexample generation in probabilistic model checking. IEEE Trans. on Software Engineering **35**(2) (2009) 241–257
21. Wimmer, R., Jansen, N., Ábrahám, E., Katoen, J.P., Becker, B.: Minimal counterexamples for linear-time probabilistic verification. Theoretical Computer Science **549** (2014) 61–100

22. Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for web transactions. ACM Trans. on Information and System Security **1**(1) (1998) 66–92
23. Herman, T.: Probabilistic self-stabilization. Information Processing Letters **35**(2) (1990) 63–67
24. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Communications of the ACM **28**(6) (1985) 637–647