Minimal Critical Subsystems for Discrete-Time Markov Models*

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

[1] Albert-Ludwigs-University Freiburg, Germany
{wimmer, becker}@informatik.uni-freiburg.de
[2] RWTH Aachen University, Germany
{nils.jansen, abraham, katoen}@informatik.rwth-aachen.de

Abstract. We propose a new approach to compute *counterexamples* for violated ω-regular properties of discrete-time Markov chains and Markov decision processes. Whereas most approaches compute a set of system paths as a counterexample, we determine a *critical subsystem* that already violates the given property. In earlier work we introduced methods to compute such subsystems based on a search for shortest paths. In this paper we use *SMT solvers* and *mixed integer linear programming* to determine *minimal* critical subsystems.

1 Introduction

Systems with uncertainties often act in safety-critical environments. In order to use the advantages of formal verification, formal models are needed. Popular modeling formalisms for such systems are *discrete-time Markov chains (DTMCs)* and—in the presence of non-determinism—*Markov decision processes (MDPs)*.

State-of-the-art model checking algorithms verify probabilistic safety properties like "The probability to reach a safety-critical state is at most 10^{-3}" or, more generally, ω-regular properties [1], efficiently by solving linear equation systems [2]. Thereby, if the property is violated, they do not provide any information about the reasons why this is the case. However, this is not only strongly needed for debugging purposes, but it is also exploited for abstraction refinement in CEGAR frameworks [3, 4]. Therefore, in recent years much research effort has been made to develop algorithms for *counterexample generation* for DTMCs and MDPs (see, e. g., [5–13]). Most of these algorithms [6–9] yield *path-based* counterexamples, i. e., counterexamples in the form of a set of finite paths that all lead from the initial state to a safety-critical state and whose joint probability mass exceeds the allowed limit.

* This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS) and the DFG project "CEBug – Counterexample Generation for Stochastic Systems using Bounded Model Checking"

Unfortunately, the number of paths needed for a counterexample is often very large or even infinite, in particular if the gap between the allowed probability and its actual value is small. The size of the counterexample may be several orders of magnitude larger than the number of system states, rendering the counterexample practically unusable for debugging purposes. Different proposals have been made to alleviate this problem: [6] represents the path set as a regular expression, [7] detects loops on paths, and [8] shrinks paths through strongly connected components into single transitions.

As an alternative to path-based counterexamples, the usage of small *critical subsystems* has been proposed in [5, 10]. A critical subsystem is a part of the Markov chain such that the probability to reach a safety-critical state (or, more generally, to satisfy an ω-regular property) inside this part exceeds the bound. This induces a path-based counterexample by considering all paths leading through this subsystem. Contrary to the path-based representation, the size of a critical subsystem is bounded by the size of the model under consideration. Different heuristic methods have been proposed for the computation of small critical subsystems: The authors of [5] apply best first search to identify a critical subsystem, while in [10] a novel technique is presented that is based on a hierarchical abstraction of DTMCs in combination with heuristics for the selection of the states to be contained in the subsystem.

Both approaches use heuristic methods to select the states of a critical subsystem. However, we are not aware of any algorithm that is suited to compute a *minimal* critical subsystem, neither in terms of the number of states nor of the number of transitions. In this paper we fill this gap. We provide formulations as a SAT-modulo theories (SMT) problem and as a mixed integer linear program (MILP) which yield state-minimal critical subsystems of DTMCs and MDPs, respectively. We will present a number of optimizations which significantly speed up the computation times in many cases. Experimental results on some case studies are provided, which show the effectiveness of our approach. We show that our MILP approach yields significantly more compact counterexamples than the heuristic methods even if the MILPs cannot be solved to optimality due to time restrictions. We present our algorithms for probabilistic safety properties, but they can be extended to the more general case of arbitrary ω-regular properties.[3]

Structure of the Paper. In Section 2 we introduce the foundations of DTMCs, MDPs, and critical subsystems. In Sections 3 and 4 we present different approaches for the computation of state-minimal subsystems for DTMCs and MDPs. We discuss experimental results in Section 5 and finally draw a conclusion in Section 6.

2 Foundations

We first introduce discrete-time Markov chains and discrete-time Markov decision processes as well as critical subsystems for both models.

[3] They can be reduced to reachability after a product construction of a DTMC or MDP, resp., with a deterministic Rabin automaton, followed by a graph analysis [2]. For more details see [14].

Discrete-Time Markov Chains.

Definition 1. *A* discrete-time Markov chain (DTMC) *is a tuple* $M = (S, s_I, P)$ *with* S *being a finite set of states,* $s_I \in S$ *the initial state and* $P : S \times S \to [0, 1]$ *the matrix of transition probabilities such that* $\sum_{s' \in S} P(s, s') \leq 1$ *for all* $s \in S$.[4]

Let in the following $M = (S, s_I, P)$ be a DTMC, $T \subseteq S$ a set of target states, and $\lambda \in [0, 1]$ an upper bound on the allowed probability to reach a target state[5] in T from the initial state s_I. This property can be formulated by the PCTL formula $\mathcal{P}_{\leq\lambda}(\Diamond\, T)$. We assume this property to be violated, i. e., the actual probability of reaching T exceeds λ.

The probability to eventually reach a target state from a state s is the unique solution of a linear equation system [2, p. 760] containing an equation for each state $s \in S$: $p_s = 1$ if $s \in T$, $p_s = 0$ if there is no path from s to any state in T, and $p_s = \sum_{s' \in S} P(s, s') \cdot p_{s'}$ in all other cases.

Definition 2. *A* subsystem *of* M *is a DTMC* $M' = (S', s'_I, P')$ *such that* $S' \subseteq S$, $s'_I \in S'$, *and* $P'(s, s') > 0$ *implies* $P'(s, s') = P(s, s')$ *for all* $s, s' \in S'$.

We call a subsystem $M' = (S', s'_I, P')$ *of* M critical *if* $s'_I = s_I$, $S' \cap T \neq \emptyset$, *and the probability to reach a state in* $S' \cap T$ *from* s'_I *in* M' *is larger than* λ.

We want to identify a *minimal* critical subsystem (MCS) of M, which induces a counterexample for $\mathcal{P}_{\leq\lambda}(\Diamond\, T)$. Minimality can thereby be defined in terms of the number of *states* or the number of *transitions*. In this paper we restrict ourselves to state-minimal subsystems. However, our approaches can easily be adapted to transition minimality. In [4] it is shown that computing MCSs for arbitrarily nested PCTL formulae is NP-complete. It is unclear if this also holds for reachability properties.

We denote the *set of transitions* of M by $E_M = \big\{(s, s') \in S \times S \,\big|\, P(s, s') > 0\big\}$, the *set of successors* of state $s \in S$ by $\mathrm{succ}_M(s) = \big\{s' \in S \,\big|\, (s, s') \in E_M\big\}$, and its *predecessors* by $\mathrm{pred}_M(s) = \big\{s' \in S \,\big|\, (s', s) \in E_M\big\}$. A *finite path* π in M is a finite sequence $\pi = s_0 s_1 \dots s_n$ such that $(s_i, s_{i+1}) \in E_M$ for all $0 \leq i < n$.

Definition 3. *Let* $M = (S, s_I, P)$ *be a DTMC with target states* $T \subseteq S$. *A state* $s \in S$ *is called* relevant *if there is a path* $\pi = s_0 s_1 s_2 \dots s_n$ *with* $s_0 = s_I$, $s_i \notin T$ *for* $0 \leq i < n$, $s_n \in T$ *and* $s = s_j$ *for some* $j \in \{0, \dots, n\}$. *A transition* $(s, s') \in E_M$ *is* relevant *if both* s *and* s' *are relevant and* $s \notin T$.

We denote the *set of relevant states* of M by S^{rel}_M and the *set of relevant transitions* by E^{rel}_M. States and transitions that are not relevant can be removed from all critical subsystems without changing the probability to reach a target

[4] Please note that we allow sub-stochastic distributions. Usually, the sum of probabilities is required to be exactly 1. This can be obtained by defining $M' = (S \cup \{s_\perp\}, s_I, P')$ with $s_\perp$ a fresh sink state, $P'(s, s') = P(s, s')$ for all $s, s' \in S$, $P'(s_\perp, s_\perp) = 1$, and finally $P(s, s_\perp) = 1 - P(s, S)$ and $P'(s_\perp, s) = 0$ for all $s \in S$.

[5] Model checking PCTL properties can be lead back to the problem of computing reachability probabilities.

state. Since we are interested in MCSs, we only have to take relevant states and transitions into account.

Let $E_M^- = \{(s,s') \in S \times S \mid (s',s) \in E_M\}$ be the *set of reversed transitions* of M. We consider the *directed graphs* $G = (S, E_M)$ and $G^- = (S, E_M^-)$.

Lemma 1. *A state $s \in S$ is* relevant *iff s is reachable from the initial state s_I in G and s is reachable from a target state in G^-. A transition $(s,s') \in E_M$ is* relevant *iff s is reachable from the initial state s_I in G and s' is reachable from a target state in G^-, and $s \notin T$.*

This lemma shows that the set S_M^{rel} of relevant states and the set E_M^{rel} of relevant transitions can be determined in linear time in the size of the DTMC by two simple graph analyses.

Markov Decision Processes. Extending DTMCs with non-determinism yields the class of Markov decision processes:

Definition 4. *A* discrete-time Markov decision process (MDP) *M is a tuple $M = (S, s_I, A, P)$ such that S is a finite set of states, $s_I \in S$ the initial state, A a finite set of actions, and $P : S \times A \times S \to [0,1]$ a function such that $\sum_{s' \in S} P(s,a,s') \leq 1$ for all $a \in A$ and all $s \in S$.*

If $s \in S$ is the current state of an MDP M, its successor state is determined as follows: First a *non-deterministic* choice between the actions in A is made; say $a \in A$ is chosen. Then the successor state of s is determined *probabilistically* according to the distribution $P(s,a,\cdot)$.

We set $\mathrm{succ}_M(s,a) = \{s' \in S \mid P(s,a,s') > 0\}$, $\mathrm{pred}_M(s,a) = \{s' \in S \mid P(s',a,s) > 0\}$, and $E_M = \{(s,s') \in S \times S \mid \exists a \in A : P(s,a,s') > 0\}$. Relevant states S_M^{rel} and transitions E_M^{rel} are defined in the same way as for DTMCs.

Before probability measures can be defined for MDPs, the non-determinism has to be resolved. This is done by an entity called *scheduler*. For our purposes we do not need schedulers in their full generality, which are allowed to return a probability distribution over the actions A, depending on the path that led from the initial state to the current state. Instead, for reachability properties the following subclass suffices [2, Lemma 10.102]:

Definition 5. *Let $M = (S, s_I, A, P)$ be an MDP. A (deterministic memoryless)* scheduler *for M is a function $\sigma : S \to A$.*

Such a scheduler σ induces a DTMC $M^\sigma = (S, s_I, P^\sigma)$ with $P^\sigma(s,s') = P(s, \sigma(s), s')$. The probability of reaching a target state is now computed in this induced DTMC. The property $\mathcal{P}_{\leq \lambda}(\Diamond\, T)$ is satisfied in an MDP $M = (S, s_I, A, P)$ if it is satisfied in M^σ for all schedulers σ. Since all schedulers guarantee a reachability probability of at most λ, this implies that the maximal reachability probability is at most λ. If the property is violated, there is a non-empty set of schedulers for which the probability exceeds λ. We call them *critical schedulers*.

We want to compute a critical scheduler and a critical subsystem in the corresponding induced DTMC that is state- (transition-) minimal among all critical subsystems for all critical schedulers. Computing state-minimal critical subsystems for reachability properties of MDPs is NP-complete [4].

SAT-Modulo-Theories. SAT-modulo-theories (SMT) [15] refers to a generalization of the classical propositional satisfiability problem (SAT). Compared to SAT problems, in an SMT formula atomic propositions may be replaced by atoms of a given theory. For the computation of MCSs this theory is linear real arithmetic.

SMT problems are solved by the combination of a DPLL-procedure (as used for deciding SAT problems) with a theory solver that is able to decide the satisfiability of conjunctions of theory atoms. For a description of such a combined algorithm see [16].

Mixed Integer Linear Programming. In contrast to SMT, mixed integer linear programs consist only of a *conjunction* of linear inequalities. A subset of the variables occurring in the inequalities are restricted to take only integer values, which makes solving MILPs NP-hard.

Definition 6. *Let* $A \in \mathbb{Q}^{n\times m}$, $B \in \mathbb{Q}^{k\times m}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, *and* $d \in \mathbb{Q}^k$. *A* mixed integer linear program *(MILP) consists in computing* $\min c^T x + d^T y$ *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 from removing the integrality constraints, can be solved efficiently. MILPs are widely used in operations research, hardware-software codesign and numerous other applications. Efficient open source as well as commercial implementations are available like Scip or Cplex. We refer the reader to, e. g., [17] for more information on solving MILPs.

3 Computing Minimal Critical Subsystems for DTMCs

The problem to find state-minimal critical subsystems for DTMCs can be specified as an SMT problem over linear real arithmetic, which we present in this section. As the experimental results were not satisfactory, we additionally elaborated MILP formulations of this problem. We also report on further optimizations that lead to a noticeable speed-up in many cases.

3.1 Formulation as an SMT Problem

We first specify an SMT formula over linear real arithmetic whose satisfying variable assignments correspond to the critical subsystems of M. The SMT formula is shown in Fig. 1. We use $\oplus$ for the binary XOR operator.

We introduce a variable $x_s \in [0,1] \subseteq \mathbb{R}$ for each relevant state $s \in S_M^{rel}$. We require in the formula that $x_s = 1$ or $x_s = 0$ holds. A state $s \in S_M^{rel}$ is contained in the subsystem iff $x_s = 1$ for the computed optimal satisfying assignment. In order to obtain a state-minimal critical subsystem, we have to minimize the number of x_s-variables to which the value 1 is assigned, or equivalently, the sum over all x_s-variables (line 1a). Besides the x_s variables we need one real-valued

$$\text{minimize} \quad \sum_{s \in S_M^{rel}} x_s \tag{1a}$$

$$\text{such that} \quad p_{s_I} > \lambda \tag{1b}$$

$$\forall s \in S_M^{rel} \cap T : \big((x_s = 0 \wedge p_s = 0) \oplus (x_s = 1 \wedge p_s = 1)\big) \tag{1c}$$

$$\forall s \in S_M^{rel} \setminus T : \Big((x_s = 0 \wedge p_s = 0) \oplus \big(x_s = 1 \wedge p_s = \sum_{s' \in \mathrm{succ}_M(s) \cap S_M^{rel}} P(s, s') \cdot p_{s'}\big)\Big) \,. \tag{1d}$$

Fig. 1. SMT formulation for state-minimal critical subsystems of DTMCs

variable $p_s \in [0, 1] \subseteq \mathbb{R}$ for each state $s \in S_M^{rel}$ to which the probability of reaching a target state from s inside the subsystem is assigned.

If x_s is zero, the corresponding state s does not belong to the subsystem. Then its probability contribution is also zero. Target states that are contained in the subsystem have probability one (line 1c). Note that the MCS does not need to contain all target states. The probability of all non-target states in the subsystem is the weighted sum over the probabilities of the relevant successor states (line 1d). In order to obtain a critical subsystem we additionally have to require that $p_{s_I} > \lambda$ (line 1b).

The size of this formula is linear in the size of M. Since most of the state-of-the-art SMT solvers for linear real arithmetic cannot cope with the minimization of objective functions, we apply binary search in the range $\{1, \ldots, |S_M^{rel}|\}$ for the optimal value of the objective function. Starting with $k_l = 1$ and $k_u = |S_M^{rel}|$, we iteratively search for critical subsystems whose number of states is between k_l and $k_m := k_l + (k_u - k_l)/2$. If we find such a subsystem with k states, then we set k_u to $k - 1$. Otherwise we set k_l to $k_m + 1$. We repeat the search until $k_u < k_l$.

3.2 Formulation as a Mixed Integer Linear Program

The formulation as an SMT problem gives a good intuition how an MCS can be computed using solver technologies. However, as the experiments will show, the solution process is very time-consuming. This might be due to the fact that SMT solvers distinguish many cases while searching for a solution because of the involved disjunctions. We therefore reformulate the problem as an MILP that does not contain any disjunctions. The MILP is shown in Fig. 2.

In order to avoid the disjunctions of the SMT formulation, we need to explicitly require the variables x_s to be integer in contrast to the SMT formulation with $x_s \in [0, 1] \subseteq \mathbb{R}$. Hence, the MILP contains the variables $x_s \in [0, 1] \subseteq \mathbb{Z}$ and $p_s \in [0, 1] \subseteq \mathbb{R}$ for all states $s \in S_M^{rel}$.

The constraints can be translated as follows: For target states $s \in S_M^{rel} \cap T$, the condition (1c) of the SMT formulation corresponds to $p_s = x_s$ (line 2c). For the remaining states $s \in S_M^{rel} \setminus T$, we ensure by $p_s \leq x_s$ that the probability contribution of not selected states is zero (line 2d). For all non-target states s, an upper bound on the probability contribution p_s is given by the sum of the

$$\text{minimize} \qquad \Big(-\frac{1}{2}p_{s_I} + \sum_{s \in S_M^{rel}} x_s\Big) \tag{2a}$$

$$\text{such that} \qquad p_{s_I} > \lambda \tag{2b}$$

$$\forall s \in S_M^{rel} \cap T : p_s = x_s \tag{2c}$$

$$\forall s \in S_M^{rel} \setminus T : p_s \leq x_s \tag{2d}$$

$$p_s \leq \sum_{s' \in \text{succ}_M(s) \cap S_M^{rel}} P(s, s') \cdot p_{s'} \ . \tag{2e}$$

Fig. 2. MILP formulation for state-minimal critical subsystems of DTMCs

probabilities $p_{s'}$ of the relevant successor states s', weighted by the according transition probabilities $P(s, s')$ (line 2e). Together with the requirement that the probability of the initial state has to be larger than λ (line 2b) this describes the critical subsystems of the DTMC under consideration.

Using this formulation and the same objective function as in the SMT formula, the exact probability of reaching target states in the resulting MCS is not computed as a by-product. We would only compute a lower bound, because line (2e) is an inequality. However, we can achieve this by forcing the solver to maximize p_{s_I}. We change the objective function to $\min\big(-\frac{1}{2}p_{s_I} + \sum_{s \in S_M^{rel}} x_s\big)$. Then the solver computes not only an arbitrary MCS, but among all MCSs one with maximal probability, and assigns to the variable p_{s_I} its actual reachability probability. A factor $0 < c < 1$ is needed because if we only subtract the probability of the initial state, the solver may add an additional state if this results in $p_{s_I} = 1$. We chose $c = \frac{1}{2}$.

3.3 Optimizations

In the following we describe optimizations both of the SMT and the MILP formulation. They add redundant constraints to the problem. These constraints may help the solver to detect unsatisfiable branches in the search space earlier.

Successor and Predecessor Constraints. In order to guide the solver to choose states that form complete paths leading from the initial state to the set of target states, we firstly add the following optional constraints to the MILP formulation in Fig. 2:

$$\forall s \in S_M^{rel} \setminus T : \ -x_s + \sum_{s' \in (\text{succ}_M(s) \cap S_M^{rel}) \setminus \{s\}} x_{s'} \geq 0 \tag{3a}$$

$$\forall s \in S_M^{rel} \setminus \{s_I\} : \ -x_s + \sum_{s' \in (\text{pred}_M(s) \cap S_M^{rel}) \setminus \{s\}} x_{s'} \geq 0 \ . \tag{3b}$$

The first set of constraints (3a), which we call *forward cuts*, states that each non-target state in the MCS must have a proper successor state which is also

contained in the MCS. Proper in this case means that self-loops are ignored. The second set of constraints (3b), called *backward cuts*, requires that each non-initial state in the MCS has a proper predecessor in the MCS.

For MILP, forward and backward cuts do not modify the feasible solutions but add cutting planes which tighten the LP-relaxation of the MILP and may lead to better lower bounds on the optimal value.

For the SMT formulation similar constraints can be constructed. We omit their description here because in our experimental results they did not improve the performance. A reason for this phenomenon could be that these constraints come with an additional effort in propagation and theory solving that is not compensated by their positive effect of restricting the solution set.

SCC Constraints. The forward respectively backward cuts do not encode that all states of the MCS are forwards respectively backwards reachable: A satisfying assignment could define a loop to belong to the subsystem even if in the solution the states of the loop are connected neither to the initial nor to any target state.

To strengthen the effect of forward and backward cuts, we make use of strongly connected components. Formally, a *strongly-connected component (SCC)* of a DTMC $M = (S, s_I, P)$ is a maximal subset $C \subseteq S$ such that each state $s \in C$ is reachable from each state $s' \in C$ visiting only states from C. The *input states* $\mathrm{In}(C)$ of an SCC C are those states which have an in-coming transition from outside the SCC, i. e., $\mathrm{In}(C) = \{s \in C \mid \exists s' \in S \setminus C : P(s', s) > 0\}$. The *output states* of C, denoted $\mathrm{Out}(C)$, are those states outside C which can be reached from C via a single transition. Hence, $\mathrm{Out}(C) = \{s \in S \setminus C \mid \exists s' \in C : P(s', s) > 0\}$.

A state of an SCC can be reached from the initial state only through one of the SCC's input states. Therefore we define an *SCC input cut* for each SCC assuring that, if none of its input states is included in the MCS, then the MCS does not contain any states of the SCC. Line 4a shows the SMT variant of this constraint, whereas line 4b gives the corresponding MILP formulation:

$$\bigwedge_{s \in \mathrm{In}(C)} x_s = 0 \quad \Rightarrow \quad \bigwedge_{s \in C \setminus \mathrm{In}(C)} x_s = 0 \tag{4a}$$

$$\sum_{s \in C \setminus \mathrm{In}(C)} x_s \leq \big|C \setminus \mathrm{In}(C)\big| \cdot \sum_{s \in \mathrm{In}(C)} x_s \ . \tag{4b}$$

Analogously, starting from a state inside an SCC, all paths to a target state lead through one of the SCC's output states. Therefore, if no output state of an SCC C is selected, we do not want to select any state of the SCC. Line 5a contains the SMT and line 5b the MILP formulation of this *SCC output cut*:

$$\bigwedge_{s \in \mathrm{Out}(C)} x_s = 0 \quad \Rightarrow \quad \bigwedge_{s \in C} x_s = 0 \tag{5a}$$

$$\sum_{s \in C} x_s \leq \big|C\big| \cdot \sum_{s \in \mathrm{Out}(C)} x_s \ . \tag{5b}$$

Complete Reachability Encoding. Although the SCC cuts further restrict the selection of unreachable states, they still do not encode reachability exactly: We could, for example, select a path from an input to an output state of an SCC and additionally select an unreachable loop inside the SCC.

For a complete encoding of *forward* reachability, we introduce a variable $r_s^{\rightarrow} \in [0,1] \subseteq \mathbb{R}$ for each state $s \in S_M^{rel}$. The values of these variables define a partial order on the states. We make use of this partial order to express forward reachability in critical subsystems: We encode that for each selected state s there is a path $s_0 \dots s_n$ from the initial state $s_0 = s_I$ to $s_n = s$ such that $r_i^{\rightarrow} < r_{i+1}^{\rightarrow}$ for all $0 \leq i < n$ and all states on the path are selected, i. e., $x_{s_i} = 1$ for all $0 \leq i \leq n$. Note that we can assign a proper value to $r_s^{\rightarrow}$ for each reachable state s, for example the value $n_s/|S_M^{rel}|$ with n_s being the size of the longest loop-free path leading from the initial state to s.

An SMT encoding of forward reachability can be defined as follows:

$$\forall s \in S_M^{rel} \setminus \{s_I\} : \left(\neg x_s \vee \bigvee_{s' \in \mathrm{pred}_M(s) \cap S_M^{rel}} (x_{s'} \wedge r_{s'}^{\rightarrow} < r_s^{\rightarrow})\right) . \tag{6a}$$

The SMT encoding of *backward* reachability is analogous, using a variable $r_s^{\leftarrow} \in [0,1] \subseteq \mathbb{R}$ for each state $s \in S_M^{rel}$:

$$\forall s \in S_M^{rel} \setminus T : \left(\neg x_s \vee \bigvee_{s' \in \mathrm{succ}_M(s) \cap S_M^{rel}} (x_{s'} \wedge r_s^{\leftarrow} < r_{s'}^{\leftarrow})\right) . \tag{7a}$$

For the MILP encoding of *forward* reachability, for each transition from a state s to s' we additionally use an integer variable $t_{s,s'}^{\rightarrow} \in [0,1] \subseteq \mathbb{Z}$. These variables correspond to the choice of the predecessor states in the disjunctions of the SMT encoding. Again, we encode that for each selected state s there is a path in the selected subsystem leading from the initial state to s. The variable $t_{s',s}^{\rightarrow}$ encodes if the transition from s' to s appears in that path.

$$\forall s \in S_M^{rel}\ \forall s' \in (\mathrm{succ}_M(s) \cap S_M^{rel}) : 2t_{s,s'}^{\rightarrow} \leq x_s + x_{s'} \tag{8a}$$

$$r_s^{\rightarrow} < r_{s'}^{\rightarrow} + (1 - t_{s,s'}^{\rightarrow}) \tag{8b}$$

$$\forall s \in S_M^{rel} \setminus \{s_I\} : (1 - x_s) + \sum_{s' \in \mathrm{pred}_M(s) \cap S_M^{rel}} t_{s',s}^{\rightarrow} \geq 1 . \tag{8c}$$

Lines 8a and 8b encode that each transition from s to s' with $t_{s,s'}^{\rightarrow} = 1$ connects selected states with $r_s^{\rightarrow} < r_{s'}^{\rightarrow}$. Under this assumption, the constraints defined in line 8c imply by induction that for each selected state there is a reachable selected predecessor state.

Backward reachability is analogous using a variable $t_{s,s'}^{\leftarrow} \in [0,1] \subseteq \mathbb{Z}$ for each transition:

$$\forall s \in S_M^{rel}\ \forall s' \in (\mathrm{succ}_M(s) \cap S_M^{rel}) : 2t_{s,s'}^{\leftarrow} \leq x_s + x_{s'} \tag{9a}$$

$$r_s^{\leftarrow} < r_{s'}^{\leftarrow} + (1 - t_{s,s'}^{\leftarrow}) \tag{9b}$$

$$\forall s \in S_M^{rel} \setminus T : (1 - x_s) + \sum_{s' \in \mathrm{succ}_M(s) \cap S_M^{rel}} t_{s,s'}^{\leftarrow} \geq 1 . \tag{9c}$$

$$\text{minimize} \quad \sum_{s \in S_M^{rel}} x_s \tag{10a}$$

$$\text{such that} \quad p_{s_I} > \lambda \tag{10b}$$

$$\forall s \in S_M^{rel} \cap T : \big((x_s = 0 \wedge p_s = 0) \oplus (x_s = 1 \wedge p_s = 1)\big) \tag{10c}$$

$$\forall s \in S_M^{rel} \setminus T : \Bigg(\Big(x_s = 0 \wedge p_s = 0\Big) \oplus \Big(x_s = 1 \wedge \bigvee_{a \in A} \big(a_s = a \wedge p_s = \sum_{s' \in \mathrm{succ}_M(s,a) \cap S_M^{rel}} P(s,a,s') \cdot p_{s'}\big)\Big)\Bigg) . \tag{10d}$$

Fig. 3. SMT formulation for state-minimal critical subsystems of MDPs

These encodings come at the cost of new variables, but they cut all subsystems with unreachable states, especially unreachable loops which were not covered by the previous two encodings.

4 Computing Minimal Critical Subsystems for MDPs

In this section we describe how to extend our SMT- and MILP-based formulations to Markov decision processes. Using these formulations, we not only provide a state-minimal critical subsystem but also the corresponding critical scheduler.

4.1 SMT Formulation

The SMT formulation for MCSs for MDPs straightly follows the ideas for DTMCs. We additionally introduce a variable $a_s \in \big[0, |A|-1\big] \subseteq \mathbb{R}$ for all states $s \in S_M^{rel} \setminus T$ which stores the action selected by a critical scheduler. If each action is assigned a unique number in the range $0, \ldots, |A|-1$, this again results in an SMT problem over linear real arithmetic, which is shown in Fig. 3.

4.2 MILP Formulation

The corresponding MILP for computing state-minimal critical subsystems for MDPs is shown in Fig. 4. We again have the decision variables $x_s \in [0,1] \subseteq \mathbb{Z}$ for $s \in S_M^{rel}$ and the probability variables $p_s \in [0,1] \subseteq \mathbb{R}$ for $s \in S_M^{rel}$. In contrast to the SMT formulation, for the MILP constraints we need a variable $a_s \in [0,1] \subseteq \mathbb{Z}$ for each state $s \in S_M^{rel}$ and each action $a \in A$. The variable a_s will carry the value 1 if the critical scheduler selects action a in state s, and 0 otherwise.

The main difference to the MILP of DTMCs is line (11f). If the current action is not selected, i. e., $a_s = 0$, the constraint is not a restriction for p_s. Otherwise, if $a_s = 1$, the constraint is equivalent to $p_s \leq \sum_{s' \in \mathrm{succ}(s,a) \cap S_M^{rel}} P(s,a,s') \cdot p_{s'}$, which is the analogous constraint to the formulation for DTMCs.

$$\text{minimize} \qquad \Big(-\frac{1}{2}p_{s_I} + \sum_{s \in S_M^{rel}} x_s\Big) \tag{11a}$$

$$\text{such that} \qquad p_{s_I} > \lambda \tag{11b}$$

$$\forall s \in S_M^{rel} \cap T : x_s = p_s \tag{11c}$$

$$\forall s \in S_M^{rel} \setminus T : p_s \leq x_s \tag{11d}$$

$$x_s = \sum_{a \in A} a_s \tag{11e}$$

$$\forall s \in S_M^{rel} \setminus T \; \forall a \in A : p_s \leq \Big(\sum_{s' \in \mathrm{succ}_M(s,a) \cap S_M^{rel}} P(s,a,s') \cdot p_{s'} \Big) + (1 - a_s) \; . \tag{11f}$$

Fig. 4. MILP formulation for state-minimal critical subsystems of MDPs

The redundant constraints that we have added to the SMT and MILP formulations for DTMCs in order to make the solution process more efficient can easily be transferred to MDPs. We omit them here due to space restrictions.

5 Experimental Evaluation

In order to evaluate the performance of our SMT and MILP formulations for state-minimal critical subsystems of DTMCs, we implemented a tool called SubSys in C++ and applied it to two series of test cases. For all benchmarks, we used Prism [18] models, which are available at `http://prismmodelchecker.org`.

(1) The *crowds protocol* [19] provides a mechanism for anonymous web browsing by routing messages through a network of N nodes. If a node wants to send a message, it has a probabilistic choice whether to deliver the message directly to its destination or to forward it to a randomly selected successor node. This procedure preserves anonymous sending of messages, as the original sender of a message cannot be determined. One instance consists of R rounds of message deliveries. In the following tables we denote the different instances by crowds$N$-R. The set T of target states contains all those states where a bad group member could identify the sender of a message.

(2) The synchronous *leader election protocol* [20] models the selection of a distinguished leader node in a ring of N identical network nodes. In each round, every node randomly selects an integer number in the range $\{0 \dots K\}$. The node with the highest number becomes the leader, if this number is unique. Otherwise a new round starts. In the tables below, we denote the instances for different values of N and K by leader$N$-K.

All experiments were performed on a computer with four 2.3 GHz AMD Opteron Quad-Core CPUs and 64 GB memory, running Ubuntu 10.04 Linux in 64-bit mode. We aborted any experiment which did not finish within 7200 s or needed more than 4 GB of memory. A table entry "– TL –" means that the time limit was exceeded; the exceeding of the memory limit is denoted by "– ML –".

Table 1. Sizes of the benchmark models and comparison with the heuristic local search method of [10]

Model	$\lvert S\rvert$	$\lvert E_M\rvert$	$\lvert T\rvert$	λ	$\lvert S_{\mathrm{MCS}}\rvert$	$\lvert E_{\mathrm{MCS}}\rvert$	$\lvert S_{\mathrm{heur}}\rvert$	$\lvert E_{\mathrm{heur}}\rvert$
crowds2-3	183	243	26	0.09	22	27	23	27
crowds2-4	356	476	85	0.09	22	27	23	27
crowds2-5	612	822	196	0.09	22	27	23	27
crowds3-3	396	576	37	0.09	37	51	40	56
crowds3-4	901	1321	153	0.09	37	51	40	56
crowds3-5	1772	2612	425	0.09	37	51	40	56
crowds5-4	3515	6035	346	0.09	72	123	94	156
crowds5-6	18817	32677	3710	0.09	72	123	145	253
crowds5-8	68740	120220	19488	0.09	72	123	198	356
leader3-2	22	29	1	0.5	15	18	17	20
leader3-3	61	87	1	0.5	33	45	40	54
leader3-4	135	198	1	0.5	70	101	76	108
leader4-2	55	70	1	0.5	34	41	44	54
leader4-3	256	336	1	0.5	132	171	170	220
leader4-4	782	1037	1	0.5	395	522	459	605
leader4-5	1889	2513	1	0.5	946	1257	1050	1393
leader4-6	3902	5197	1	0.5	1953	2600	2103	2797

For solving the SMT formulae and the MILPs we used a number of state-of-the-art solvers, from which we selected, after a series of preliminary experiments, the most efficient ones, namely Z3 3.1 [21] (`http://research.microsoft.com/en-us/um/redmond/projects/z3`) as an SMT solver for linear real arithmetic, SCIP 2.0.2 (`http://scip.zib.de`) as a publicly available MILP solver, and CPLEX 12.3 (`http://www-01.ibm.com/software/integration/optimization/cplex-optimizer`) as a commercial MILP solver.

Table 1 contains statistics on our benchmarks. The columns contain (from left to right) the model name, the number of states, the number of transitions with non-zero probability, the number of target states, the probability bound, the number of states in the MCS, and finally the number of transitions in the MCS. The last two columns contain the sizes of heuristically computed critical subsystems. We used the local search approach of [10] to determine these subsystems. For all instances the heuristic tool terminated within 6 min.

We give the running times of Z3, SCIP, and CPLEX in Table 2. CPLEX supports a parallel mode, in which we started 16 threads in parallel. Therefore we give for CPLEX the accumulated times of all threads and, in parentheses, the actual time from start to termination of the tool. All times are given in seconds.

The block of columns entitled "w/o redundant constraints" contains the running times of the solvers without any optimizations. The block "optimal conf." lists the optimal times, i. e., the times achieved by adding the combination of optional constraints that leads to the smallest computation time. These running times can be obtained in general by using a portfolio approach which runs the different combinations of redundant constraints in parallel. Using Z3 did not

Table 2. Running times of Z3, SCIP, and CPLEX for computing MCSs

	w/o redundant constraints			optimal conf.		
Model	Z3	SCIP	CPLEX	Z3	SCIP	CPLEX
crowds2-3	6.80	0.16	1.33 (0.28)	4.82	0.12	0.06 (0.11)
crowds2-4	123.34	0.47	0.30 (0.24)	23.72	0.30	0.30 (0.24)
crowds2-5	293.94	0.90	0.56 (0.45)	152.28	0.60	0.56 (0.24)
crowds3-3	5616.39	0.64	0.49 (0.33)	640.61	0.35	0.38 (0.30)
crowds3-4	– TL –	4.29	5.53 (2.07)	– TL –	1.45	0.89 (0.58)
crowds3-5	– TL –	23.49	6.66 (2.77)	– TL –	5.58	1.51 (0.87)
crowds5-4	– TL –	743.84	14.23 (5.07)	– TL –	13.28	12.51 (4.89)
crowds5-6	– TL –	– TL –	302.03 (38.39)	– TL –	1947.46	100.26 (23.52)
crowds5-8	– TL –	– TL –	– TL –	– TL –	– TL –	1000.79 (145.84)
leader3-2	0.07	0.07	0.62 (0.22)	0.05	0.01	0.21 (0.13)
leader3-3	– TL –	91.89	0.43 (0.22)	– TL –	0.06	0.02 (0.06)
leader3-4	– TL –	2346.59	0.70 (0.36)	– TL –	0.40	0.07 (0.09)
leader4-2	3.57	0.23	0.45 (0.21)	1.38	0.07	0.24 (0.17)
leader4-3	– TL –	1390.79	22.33 (3.38)	– TL –	0.21	0.49 (0.37)
leader4-4	– TL –	– TL –	– TL –	– TL –	1.49	1.88 (1.21)
leader4-5	– TL –	– TL –	– TL –	– TL –	1.15	4.06 (2.80)
leader4-6	– TL –	– TL –	– ML –	– TL –	– TL –	8.70 (5.92)

lead to satisfying results although the optimizations, especially the reachability cuts, decreased the running times clearly. A significant speed-up is recognizable using the MILP solvers. The optimal running times were in some cases smaller by orders of magnitude considering in particular the large benchmarks for which the standard formulation could not be solved within the time limit.

To gain more insight into the effects of the different kinds of redundant constraints, we list more detailed results for crowds5-6 and leader4-5 in Table 3. The left block of columns contains the running times without SCC cuts. The column "–" contains the values without forward and backward cuts, the column "$\rightarrow$" the values with forward cuts, "$\leftarrow$" the values with backward cuts and "$\leftrightarrow$" with both forward and backward cuts. The values for the four different combinations of reachability cuts (none, only forward, only backward, and both) are listed in the according rows of the table.

Comparing the values for the reachability cuts, we can observe that they have a negative effect on the running times for crowds5-6 with MILP solvers. However, they speed up the solution of leader4-5 by a factor of more than 10^3, decreasing the solution times from more than 7200 s to less than 5 s. The same tendency can be observed for all crowds and leader instances, respectively.

The addition of backward cuts to crowds5-6 reduces the running time to about one third, and they typically decrease the times for most of the instances. Since the SCC cuts are even less effective, we only give the minimal value of the three cases (with SCC input, SCC output, and with both kinds of SCC cuts).

Fig. 5 shows the size of the MCS of crowds5-6 for different values of λ (red solid lines), comparing it with the size of heuristically computed critical subsystems

Table 3. Runtimes for crowds5-6 and leader4-5 with and without redundant constraints using CPLEX as MILP solver

		no SCC cuts				with SCC cuts			
	Reach	–	→	←	↔	–	→	←	↔
crowds5-6	none	302.03 (38.39)	367.49 (44.70)	**103.20** (23.52)	149.73 (26.07)	301.87 (38.64)	342.07 (42.76)	**100.26** (23.52)	138.36 (25.20)
	fwd	656.13 (120.04)	1292.59 (148.82)	651.47 (112.47)	966.57 (127.94)	634.40 (118.22)	833.37 (108.13)	646.64 (111.18)	925.95 (125.52)
	bwd	4043.93 (384.74)	3613.96 (358.49)	770.90 (121.02)	1070.50 (130.45)	3911.81 (375.48)	3603.49 (358.25)	756.28 (119.90)	1074.36 (130.72)
	both	2107.84 (251.41)	1403.44 (185.34)	5972.98 (546.83)	2191.83 (281.07)	1986.37 (238.58)	1379.78 (183.38)	5925.31 (542.18)	2210.68 (284.51)
leader4-5	none	– TL –	– TL –	– TL –	– TL –	– TL –	– TL –	– TL –	– TL –
	fwd	284.04 (40.02)	254.56 (35.97)	286.83 (40.85)	261.53 (36.46)	294.71 (41.15)	259.98 (36.21)	285.33 (40.57)	251.69 (35.80)
	bwd	6.30 (3.73)	6.29 (3.69)	6.27 (3.72)	6.10 (3.71)	5.89 (3.65)	5.73 (3.66)	5.78 (3.67)	5.95 (3.69)
	both	4.46 (2.77)	**4.06** (2.80)	4.34 (2.83)	4.56 (2.91)	**4.17** (2.77)	4.41 (2.83)	4.10 (2.84)	4.39 (2.90)

using the local search of [10] (blue dotted lines). For $\lambda \geq 0.23$, we could only compute an upper bound (within 8 % from the optimal value) on the size of the MCS using our MILP formulation due to the timeout of 2 hours. Also the local search tool ran into a timeout for $\lambda \geq 0.35$, however, without yielding a critical subsystem.

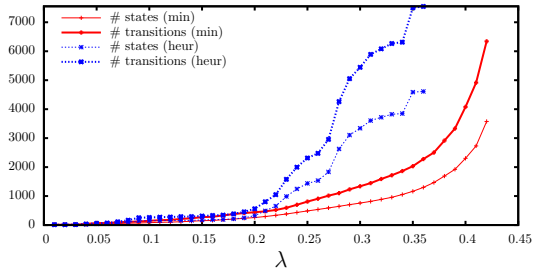

Fig. 5. Size of the MCS and heuristically determined critical subsystems for crowds5-6 and different values of λ

6 Conclusion

In this paper we have shown how to compute state-minimal critical subsystems for DTMCs and MDPs using SMT- and MILP-based formulations. By adding redundant constraints, which trigger implications for SMT and tighten the LP-relaxation of the MILP, the solution process can be speeded up clearly. Thereby the MILP formulation is more efficient to solve by orders of magnitude compared to the SMT formulation. A topic for future research is to analyze the theoretical complexity of computing MCSs for DTMCs. We conjecture that this problem is NP-complete. Furthermore we plan to integrate the MILP approach into the hierarchical counterexample generation tool described in [10].

Acknowledgments. The authors thank the reviewers for pointing out the relevance of [4].

References

1. Bustan, D., Rubin, S., Vardi, M.Y.: Verifying ω-regular properties of Markov chains. In: Proc. of CAV. Volume 3114 of LNCS, Springer (2004) 189–201
2. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (2008)
3. Hermanns, H., Wachter, B., Zhang, L.: Probabilistic CEGAR. In: Proc. of CAV. Volume 5123 of LNCS, Springer (2008) 162–175
4. Chadha, R., Viswanathan, M.: A counterexample-guided abstraction-refinement framework for Markov decision processes. ACM TOCL **12**(1) (2010) 1–45
5. 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
6. Han, T., Katoen, J.P., Damman, B.: Counterexample generation in probabilistic model checking. IEEE Trans. on Software Engineering **35**(2) (2009) 241–257
7. Wimmer, R., Braitling, B., Becker, B.: Counterexample generation for discrete-time Markov chains using bounded model checking. In: Proc. of VMCAI. Volume 5403 of LNCS, Springer (2009) 366–380
8. Andrés, M.E., D'Argenio, P., van Rossum, P.: Significant diagnostic counterexamples in probabilistic model checking. In: Proc. of HVC. Volume 5394 of LNCS, Springer (2008) 129–148
9. Günther, M., Schuster, J., Siegle, M.: Symbolic calculation of k-shortest paths and related measures with the stochastic process algebra tool CASPA. In: Proc. of DYADEM-FTS, ACM Press (2010) 13–18
10. 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
11. Kattenbelt, M., Huth, M.: Verification and refutation of probabilistic specifications via games. In: Proc. of FSTTCS. Volume 4 of LIPIcs, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2009) 251–262
12. Schmalz, M., Varacca, D., Völzer, H.: Counterexamples in probabilistic LTL model checking for Markov chains. In: Proc. of CONCUR. Volume 5710 of LNCS, Springer (2009) 587–602
13. Fecher, H., Huth, M., Piterman, N., Wagner, D.: PCTL model checking of Markov chains: Truth and falsity as winning strategies in games. Performance Evaluation **67**(9) (2010) 858–872
14. Wimmer, R., Becker, B., Jansen, N., Ábrahám, E., Katoen, J.P.: Minimal critical subsystems as counterexamples for ω-regular DTMC properties. In Brandt, J., Schneider, K., eds.: Proc. of MBMV, Kovač-Verlag (2012)
15. de Moura, L.M., Bjørner, N.: Satisfiability modulo theories: introduction and applications. Communication of the ACM **54**(9) (2011) 69–77
16. Dutertre, B., de Moura, L.M.: A fast linear-arithmetic solver for DPLL(T). In: Proc. of CAV. Volume 4144 of LNCS, Springer (2006) 81–94
17. Schrijver, A.: Theory of Linear and Integer Programming. Wiley (1986)
18. Kwiatkowska, M.Z., 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
19. Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for web transactions. ACM Trans. on Information and System Security **1**(1) (1998) 66–92
20. Itai, A., Rodeh, M.: Symmetry breaking in distributed networks. Information and Computation **88**(1) (1990) 60–87
21. de Moura, L.M., Bjørner, N.: Z3: An efficient SMT solver. In: Proc. of TACAS. Volume 4963 of LNCS, Springer (2008) 337–340