Minimal Critical Subsystems as Counterexamples for ω-Regular DTMC Properties

Ralf Wimmer Bernd Becker
Albert-Ludwigs-University Freiburg, Germany
{wimmer,becker}@informatik.uni-freiburg.de

Nils Jansen Erika Ábrahám Joost-Pieter Katoen
RWTH Aachen, Germany
{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. 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 methods have been introduced to compute such subsystems for safety properties, based on a search for shortest paths. In this paper we use *mixed integer linear programming* to determine *minimal* critical subsystems for arbitrary ω*-regular* properties.

1. Introduction

Systems with uncertainties often act in safety-critical environments. In order to use the advantages of formal verification, formal models are needed. A popular modeling formalism for such systems are *discrete-time Markov chains (DTMCs)*.
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 (see, e. g., [5, 6, 7, 8, 9, 10, 11, 12, 13]). With the exception of [12], which handles LTL specifications, all of these approaches are restricted to reachability properties. Furthermore, the algorithms in [6, 7, 8, 9] yield *path-based* counterexamples for violated safety properties, 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.
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

*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"

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 [10] 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 a formulation as a mixed integer linear program (MILP) which yields state-minimal critical subsystems of DTMCs. We also present some optimizations which significantly speed up the computation times in many cases. Experimental results on some case studies are provided, which show that our 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 first for probabilistic reachability properties and extend them afterwards to general ω-regular properties.

Structure of the Paper. In Sec. 2 we introduce the foundations of DTMCs and critical subsystems. In Sec. 3 we present our approach for the computation of state-minimal subsystems for DTMCs. We discuss experimental results in Sec. 4 and finally draw a conclusion in Sec. 5.

2. Foundations

We first introduce discrete-time Markov chains, ω-regular properties, and critical subsystems. Then we take a look at mixed integer linear programs, which we use for the computation of minimal critical subsystems.

2.1. Discrete-Time Markov Chains and ω-Regular Properties

Definition 1 *Let AP be a set of atomic propositions. A* discrete-time Markov chain (DTMC) *is a tuple $M = (S, s_I, P, L)$ 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 $s \in S$.[1] $L : S \to 2^{AP}$ is a labeling function which assigns to each state $s \in S$ the propositions that are valid in s.*

[1] 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 $M' = (S \cup \{s_\perp\}, s_I, P', L')$ with $s_\perp$ a fresh sink state, $P'(s, s') = P(s, s')$ and $L'(s) = L(s)$ for all $s, s' \in S$, $L'(s_\perp) = \emptyset$, $P'(s_\perp, s_\perp) = 1$, and finally $P(s, s_\perp) = 1 - P(s, S)$ and $P'(s_\perp, s) = 0$ for all $s \in S$.

Let in the following $M = (S, s_I, P, L)$ be a DTMC over the set AP of atomic propositions. We denote the *set of transitions* of M by $E_M = \{(s, s') \in S \times S \mid P(s, s') > 0\}$, the *set of successors* of state $s \in S$ by $\mathrm{succ}_M(s) = \{s' \in S \mid (s, s') \in E_M\}$, and its *predecessors* by $\mathrm{pred}_M(s) = \{s' \in S \mid (s', s) \in E_M\}$. A finite (infinite) *path* in M is a finite (infinite) sequence $\pi = s_0 s_1 s_2 \ldots$ of states such that $(s_i, s_{i+1}) \in E_M$ for all i. We denote the set of finite (infinite) paths starting in s by $\mathrm{Paths}_{\mathrm{fin}}(s)$ ($\mathrm{Paths}_{\mathrm{inf}}(s)$, resp.). The *trace* of π, $\mathrm{trace}(\pi)$, is the sequence $L(s_0)L(s_1)L(s_2)\ldots$
A *linear-time property* over AP is a set $\mathcal{L}$ of infinite sequences $\alpha_0\alpha_1\alpha_2\ldots$ with $\alpha_i \subseteq AP$ for all i. A path π satisfies a linear-time property $\mathcal{L}$ if $\mathrm{trace}(\pi) \in \mathcal{L}$.
In this paper we will deal with a certain subclass of linear-time properties, namely ω-regular properties. In order to define them, we need the notion of Rabin automata[2].

Definition 2 *A* deterministic Rabin automaton (DRA) *is a tuple $A = (Q, q_I, \Sigma, \delta, F)$ such that Q is a finite, non-empty set of states, $q_I \in Q$ an initial state, Σ an input alphabet, $\delta : Q \times \Sigma \to Q$ a transition function, and $F \subseteq 2^Q \times 2^Q$ an acceptance condition.*

Let $w = \sigma_0\sigma_1\sigma_2\ldots \in \Sigma^\omega$ be an infinite word over Σ. The *run* r of A on w is given by the state sequence $q_0q_1q_2\ldots \in Q^\omega$ with $q_0 = q_I$ and $q_{i+1} = \delta(q_i, \sigma_i)$ for all i. We denote the set of states which occur infinitely often on a run r by $\inf(r)$. Given the acceptance condition $F = \{(L_i, K_i) \mid i = 1, \ldots, n\}$, a run r is *accepting* if, for some $i \in \{1, \ldots, n\}$, $\inf(r) \cap L_i = \emptyset$ and $\inf(r) \cap K_i \neq \emptyset$. The set of words with an accepting run is denoted by $\mathcal{L}(A)$.

Definition 3 *A linear-time property $\mathcal{L}$ over AP is* ω-regular *if there is a deterministic Rabin automaton $A = (Q, q_I, 2^{AP}, \delta, F)$ such that $\mathcal{L} = \mathcal{L}(A)$.*

For a state s of the DTMC M and an ω-regular property $\mathcal{L}$, we denote the probability to walk along a path which satisfies $\mathcal{L}$ when starting in state $s \in S$ by $\Pr^s_M(\mathcal{L}) = \Pr^s_M(\{\pi \in \mathrm{Paths}_{\mathrm{inf}}(s) \mid \mathrm{trace}(\pi) \in \mathcal{L}\})$. We consider ω-regular properties where this probability may be at most a given bound $\lambda \in [0, 1]$. Such properties are denoted by $\mathcal{P}_{\leq\lambda}(\mathcal{L})$. We assume that the initial state of M violates this property. The goal is now to identify a smallest possible part of M which already violates $\mathcal{P}_{\leq\lambda}(\mathcal{L})$.

Definition 4 *A* subsystem *of a DTMC $M = (S, s_I, P, L)$ is a DTMC $M' = (S', s'_I, P', L')$ such that $S' \subseteq S$, $s'_I \in S'$, $L'(s) = L(s)$ and $P'(s, s') > 0$ implies $P'(s, s') = P(s, s')$ for all $s, s' \in S'$.*
For a property $\mathcal{P}_{\leq\lambda}(\mathcal{L})$, we call M' critical *if $s'_I = s_I$ and $\Pr^{s'_I}_{M'}(\mathcal{L}) > \lambda$.*

We want to compute a *minimal* critical subsystem (MCS) of M for $\mathcal{P}_{\leq\lambda}(\mathcal{L})$. Minimality can be defined in terms of the number of *states* or the number of *transitions*. Though in this paper we focus on state-minimality, our approach can be easily adapted to transition-minimality.

For a proposition $a \in AP$, the property to reach an a-state is specified by $\mathcal{L} = \{\alpha_0\alpha_1\alpha_2\ldots \mid \exists i \geq 0 : a \in \alpha_i\}$. We denote this *reachability property* by the PCTL-Formula $\mathcal{P}_{\leq\lambda}(\Diamond\, a)$.
To check reachability properties, specialized algorithms have been developed, which are much more efficient than the algorithms for arbitrary ω-regular properties. To be more precise, the

[2]It is more common to use nondeterministic Büchi automata to define ω-regular properties, which have the same expressiveness as deterministic Rabin automata, but can be much more compact. However, we need to construct a product automaton of a DTMC and an appropriate ω-automaton. In case of a deterministic Rabin automaton, the result is again a DTMC, but not in case of a nondeterministic Büchi automaton.

probability to eventually reach an a-state from a state s is the unique solution of a linear equation system [2, p. 760] containing for each state $s \in S$ the equation $p_s = 1$ if $a \in L(s)$, $p_s = 0$ if there is no path from s to any a-state, and $p_s = \sum_{s' \in S} P(s, s') \cdot p_{s'}$ in all other cases.

Definition 5 *Let $M = (S, s_I, P, L)$ be a DTMC and $a \in AP$. A state $s \in S$ is* relevant *for a if there is a path $\pi = s_0 s_1 s_2 \dots s_n$ with $s_0 = s_I$, $a \notin \bigcup_{i=0}^{n-1} L(s_i)$, $a \in L(s_n)$ 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 $a \notin L(s)$.*

We write S_M^{rel} for the *set of relevant states* of M and E_M^{rel} for the *set of relevant transitions*. States and transitions that are not relevant can be removed from a DTMC without changing the probability to reach a target state. The following lemma shows that S_M^{rel} and E_M^{rel} can be determined in linear time in the size of the DTMC by two simple graph analyses.

Lemma 1 *Assume a DTMC $M = (S, s_I, P, L)$ and a proposition $a \in AP$, and let $T = \{s \in S | a \in L(s)\}$. Let furthermore $E_M^- = \left\{(s, s') \in S \times S \,\middle|\, (s', s) \in E_M\right\}$ be the set of reversed transitions of M. We consider the directed graphs $G = (S, E_M)$ and $G^- = (S, E_M^-)$.*
A state $s \in S$ is relevant for a iff s is reachable from s_I in G and s is reachable from a state $t \in T$ in G^-. A transition $(s, s') \in E_M$ is relevant iff s is reachable from s_I in G and s' is reachable from a state $t \in T$ in G^-, and $s \notin T$.

2.2. Mixed Integer Linear Programming

A *mixed integer linear program* optimizes an objective function under a condition specified by a conjunction of linear inequalities. A subset of the variables 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., [14] for more information on solving MILPs.

3. Computing Minimal Critical Subsystems for DTMCs

We give an MILP formulation[3] of the problem to find state-minimal critical subsystems for DTMCs and violated reachability properties in Sec. 3.1. We report on optimizations in Sec. 3.2 and extend our approach to general ω-regular properties in Sec. 3.3.

3.1. Minimal Critical Subsystems for Reachability Properties

Let $M = (S, s_I, P, L)$ be a DTMC, $a \in AP$ a proposition, $T = \{s \in S \,|\, a \in L(s)\}$ the set of so-called target states, and $\mathcal{P}_{\leq \lambda}(\Diamond\, a)$ a reachability property which is violated by M. We want to

[3] We also worked out Sat-Modulo-Theories formulations, but they showed to be less efficient.

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

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

$$\forall s \in S_M^{\text{rel}} \cap T : p_s = x_s \tag{1c}$$

$$\forall s \in S_M^{\text{rel}} \setminus T : p_s \leq x_s \tag{1d}$$

$$p_s \leq \sum_{s' \in \text{succ}_M(s) \cap S_M^{\text{rel}}} P(s, s') \cdot p_{s'} \; . \tag{1e}$$

Figure 1: MILP formulation for state-minimal critical subsystems of DTMCs

identify a state-minimal critical subsystem of M such that the probability to walk from s_I to a state in T in the subsystem is larger than λ. This can be formulated by the MILP shown in Fig. 1. We introduce a variable $x_s \in [0, 1] \subseteq \mathbb{Z}$ for each $s \in S_M^{\text{rel}}$, indicating if s belongs to the subsystem ($x_s = 1$) or not ($x_s = 0$). A real-valued variable $p_s \in [0, 1] \subseteq \mathbb{R}$ for each state $s \in S_M^{\text{rel}}$ contains the probability of reaching a target state from s inside the MCS. Selected target states have probability one, non-selected target states zero (line 1c). For non-target states $s \in S_M^{\text{rel}} \setminus T$, either s is not selected and $p_s = 0$, or s is selected and p_s is the sum of the probabilities $p_{s'}$ of all relevant successor states s', weighted by the according transition probabilities $P(s, s')$. For non-selected states s, line 1d ensures that $p_s = 0$. To keep our inequations linear, line 1e defines the above-mentioned sum to be an upper bound on the probabilities, covering both selected and unselected states. Line 1b assures that the subsystem is critical.
In order to obtain a state-minimal critical subsystem, we have to minimize the number of x_s-variables with value 1, or equivalently, the sum over all x_s-variables. However, due to the inequations in line 1e, minimizing this sum would give us only lower bounds on the exact probabilities of reaching target states in the resulting MCS. We can achieve to compute the exact probabilities by forcing the solver to additionally maximize p_{s_I} (line 1a). This way the solver even computes a minimal MCS with maximal probability. The factor $1/2$ is needed because if we only subtract the probability of the initial state, the solver may add an additional state if this results in $p_{s_I} = 1$.

3.2. Optimizations

In the following we describe optimizations by adding redundant constraints to the problem, which may help the solver to detect unsatisfiable branches in the search space earlier.

3.2.1. Successor and Predecessor Constraints

We can guide the solver not to select states that are unreachable in the subsystem by adding the following constraints to the MILP formulation in Fig. 1:

$$\forall s \in S_M^{\text{rel}} \setminus T : \; -x_s + \sum_{s' \in (\text{succ}_M(s) \cap S_M^{\text{rel}}) \setminus \{s\}} x_{s'} \geq 0 \tag{2a}$$

$$\forall s \in S_M^{\text{rel}} \setminus \{s_I\} : \; -x_s + \sum_{s' \in (\text{pred}_M(s) \cap S_M^{\text{rel}}) \setminus \{s\}} x_{s'} \geq 0 \; . \tag{2b}$$

The first set of constraints (2a), which we call *forward cuts*, states that each non-target state in the MCS must have a proper successor state in the MCS. Proper means that self-loops are ignored. The second set of constraints (2b), called *backward cuts*, requires that each non-initial state in the MCS has a proper predecessor in the MCS. Forward and backward cuts do not modify the feasible solutions, but add cutting planes which tighten its LP-relaxation.

3.2.2. SCC Constraints

The forward respectively backward cuts do not encode that all states of the MCS are forwards respectively backwards reachable, since they allow loops that are unreachable in the MCS. 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 $M = (S, s_I, P, L)$ 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* $\mathrm{In}(C) = \{s \in C \mid \exists s' \in S \setminus C : P(s', s) > 0\}$ of an SCC C have an in-coming transition from outside the SCC. The *output states* $\mathrm{Out}(C) = \{s \in S \setminus C \mid \exists s' \in C : P(s', s) > 0\}$ of C are those states outside C which can be reached from C via a single transition.
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 C assuring that, if none of C's input states is included in the MCS, then the MCS does not contain any states from C:

$$\sum_{s \in C \setminus \mathrm{In}(C)} x_s \leq \big|C \setminus \mathrm{In}(C)\big| \cdot \sum_{s \in \mathrm{In}(C)} x_s \,. \tag{3a}$$

Analogously, starting from a state inside an SCC, all paths to a target state lead through one of the SCC's output states. Therefore, the *SCC output cut* states that if no output state of an SCC C is selected then we do not want to select any state from C:

$$\sum_{s \in C} x_s \leq \big|C\big| \cdot \sum_{s \in \mathrm{Out}(C)} x_s \,. \tag{4a}$$

3.2.3. Complete Reachability Encoding

The SCC cuts still do not encode reachability exactly, since they allow the selection of unreachable loops in reachable SCCs. For a complete encoding of *forward* reachability, we introduce a variable $r_s^{\rightarrow} \in [0, 1] \subseteq \mathbb{R}$ for each state $s \in S_M^{\mathrm{rel}}$. The values of these variables define a partial order on the states, which we use to encode that for each selected state s there is a path $s_0 \dots s_n$ in the MCS from the initial state $s_0 = s_I$ to $s_n = s$ such that $r_i^{\rightarrow} < r_{i+1}^{\rightarrow}$ for all $0 \leq i < n$. For each transition from a state s to s' we additionally need an integer variable $t_{s,s'}^{\rightarrow} \in [0, 1] \subseteq \mathbb{Z}$ representing the choice of the predecessor state in the above paths. Note that proper values always exist, e.g., choosing for each state s the longest loop-free path from s_I to s and setting $r_s^{\rightarrow} = n_s / |S_M^{\mathrm{rel}}|$ with n_s the length of the path. Forward reachability is encoded as follows:

$$\forall s{\in}S_M^{\mathrm{rel}}\ \forall s' \in (\mathrm{succ}_M(s){\cap}S_M^{\mathrm{rel}}) : 2t^{\rightarrow}_{s,s'} \leq x_s + x_{s'} \tag{5a}$$

$$r^{\rightarrow}_s < r^{\rightarrow}_{s'} + (1 - t^{\rightarrow}_{s,s'}) \tag{5b}$$

$$\forall s \in S_M^{\mathrm{rel}} \setminus \{s_I\} : (1 - x_s) + \sum_{s' \in \mathrm{pred}_M(s) \cap S_M^{\mathrm{rel}}} t^{\rightarrow}_{s',s} \geq 1\,. \tag{5c}$$

Lines 5a and 5b encode that each transition from s to s' with $t^{\rightarrow}_{s,s'} = 1$ connects selected states with $r^{\rightarrow}_s < r^{\rightarrow}_{s'}$. Under this assumption, the constraints in line 5c imply by induction that for each selected state there is a reachable selected predecessor state.
Backward reachability is encoded analogously using variables $r^{\leftarrow}_s$ and $t^{\leftarrow}_{s,s'}$, which we substitute for the variables $r^{\rightarrow}_s$ and $t^{\rightarrow}_{s,s'}$ in the above constraints.

$$\forall s{\in}S_M^{\mathrm{rel}}\ \forall s' \in (\mathrm{succ}_M(s){\cap}S_M^{\mathrm{rel}}) : 2t^{\leftarrow}_{s,s'} \leq x_s + x_{s'} \tag{6a}$$

$$r^{\leftarrow}_s < r^{\leftarrow}_{s'} + (1 - t^{\leftarrow}_{s,s'}) \tag{6b}$$

$$\forall s \in S_M^{\mathrm{rel}} \setminus T : (1 - x_s) + \sum_{s' \in \mathrm{succ}_M(s) \cap S_M^{\mathrm{rel}}} t^{\leftarrow}_{s,s'} \geq 1\,. \tag{6c}$$

These encodings come at the cost of new variables, but they exclude all subsystems with unreachable states.

3.3. General ω-Regular Properties

We now show how the techniques described so far for reachability properties can be generalized to arbitrary ω-regular properties $\mathcal{L}$ with an upper bound λ on the probability of satisfying $\mathcal{L}$, i. e., properties of the form $\mathcal{P}_{\leq\lambda}(\mathcal{L})$. To do so we need the product automaton of the DTMC under consideration with the DRA of $\mathcal{L}$.

Definition 7 *Let $M = (S, s_I, P, L)$ be a DTMC and $A = (Q, q_I, 2^{AP}, \delta, F)$ with $F = \{(L_i, K_i) | 1 \leq i \leq n\}$ a DRA for the ω-regular property $\mathcal{L}$. The* product automaton *$M \otimes A$ is the DTMC $M \otimes A = (S \times Q, (s_I, \delta(q_I, L(s_I))), P', L')$ such that $P'((s,q),(s',q'))$ equals $P(s,s')$ if $q' = \delta(q, L(s'))$ and 0 otherwise. We use the sets L_i, K_i as new atomic propositions for labeling the states, i. e., for $H \in \{L_i, K_i \,|\, 1 \leq i \leq n\}$, $H \in L'((s,q))$ iff $q \in H$.*

SCCs of $M \otimes A$ that cannot be left, the so-called *bottom SCCs* (BSCCs), play a crucial role in the computation of $\Pr^s_M(\mathcal{L})$. We call a BSCC T of $M \otimes A$ *accepting* iff there is an index $i \in \{1, \ldots, n\}$ such that $T \cap (S \times L_i) = \emptyset$ and $T \cap (S \times K_i) \neq \emptyset$.

Lemma 2 (Theorem 10.56 of [2]) *Let M be a DTMC, s a state of M, A a DRA for $\mathcal{L}$, and let T be the union of all accepting BSCCs in $M \otimes A$. Then*

$$\Pr^s_M(\mathcal{L}) = \Pr^{(s,\delta(q_I,L(s)))}_{M\otimes A}(\Diamond\, T).$$

This means, computing probabilities of ω-regular properties reduces to a reachability analysis in the product automaton. The probability to finally reach a BSCC of a DTMC and to visit all of its states infinitely often equals 1 [2, Theorem 10.27]. This implies, in order to obtain an MCS, we

$$\text{minimize} \quad \left(-\frac{1}{2}p_{(s_I,\delta(q_I,L(s_I)))} + \sum_{s\in S^{\text{rel}}_{M\otimes A}|_M} x_s\right) \tag{7a}$$

$$\text{such that} \quad p_{(s_I,\delta(q_I,L(s_I)))} > \lambda \tag{7b}$$

$$\forall (s,q) \in T \cap S^{\text{rel}}_{M\otimes A} : x_s = p_s \tag{7c}$$

$$\forall i \in \{1,\dots,k\} \forall s \in T_i \cap S^{\text{rel}}_{M\otimes A} : x_{T_i} = x_s \tag{7d}$$

$$\forall (s,q) \in S^{\text{rel}}_{M\otimes A} \setminus T : p_{(s,q)} \leq x_s \tag{7e}$$

$$p_{(s,q)} \leq \sum_{(s',q')\in \text{succ}_{M\otimes A}(s,q)\cap S^{\text{rel}}_{M\otimes A}} P'((s,q),(s',q')) \cdot p_{(s',q')}. \tag{7f}$$

Figure 2: Minimal critical subsystems for ω-regular properties

either have to take all states of an accepting BSCC of $M \otimes A$ or none of them. One can show that the size of the resulting MCS is the same for all DRAs A with $\mathcal{L}(A) = \mathcal{L}$.

Let $T_1, \dots, T_k$ be the accepting BSCCs of $M \otimes A$, $T = \bigcup_{i=1}^{k} T_i$, and $S^{\text{rel}}_{M\otimes A}$ the relevant states of $M \otimes A$ (regarding the reachability of T). For a set $R \subseteq S \times Q$ we denote by $R|_M$ the *projection* of R on S, i. e., $R|_M = \{s \in S \mid \exists q \in Q : (s,q) \in R\}$. Since we want an MCS of M, but compute the probabilities in $M \otimes A$, we introduce one variable $x_s \in [0,1] \subseteq \mathbb{Z}$ for each state $s \in S^{\text{rel}}_{M\otimes A}|_M$ and one variable $p_{(s,q)} \in [0,1] \subseteq \mathbb{R}$ for each pair $(s,q) \in S^{\text{rel}}_{M\otimes A}$. Furthermore we use a variable $x_{T_i} \in [0,1] \subseteq \mathbb{Z}$ for each accepting BSCC T_i which indicates if the BSCC is relevant for the MCS.

The MILP formulation for MCSs is shown in Fig. 2. It follows the ideas of MCSs for reachability (see Fig. 1), with the exception that we minimize the projection of the subsystem onto the original DTMC. Constraint (7d) takes care of the fact that we either have to take all states of a BSCC or none by requiring that for all $s \in T_i|_M$ the variables x_s have the same value.

All optimizations like SCC cuts or reachability cuts can be applied to ω-regular properties in a straightforward way.

4. Experimental Evaluation

We implemented our approach in a tool called SUBSYS in C++ and applied it to two series of test cases. For all benchmarks, we used PRISM [15] models available at `http://prismmodelchecker.org`.

(1) The *crowds protocol* [16] 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. 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* [17] 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 from $\{0,\dots,K\}$. The node with the highest number becomes the leader, if this number is unique. Otherwise a new round starts. We denote the instances for different N and K by leader$N$-K.

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

Model	$\|S\|$	$\|E_M\|$	$\|T\|$	λ	$\|S_{\text{MCS}}\|$	$\|E_{\text{MCS}}\|$	$\|S_{\text{heur}}\|$	$\|E_{\text{heur}}\|$
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

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 (denoted by "– TL –") or needed more than 4 GB of memory ("– ML –").

For solving 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 the publicly available SCIP 2.0.2 solver (`http://scip.zib.de`) and the commercial CPLEX 12.3 solver (`http://www-01.ibm.com/software/integration/optimization/cplex-optimizer`).

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, 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 using the local search approach of [10]. For all instances the heuristic tool terminated within 6 min.

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

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. The optimal running times were in some cases by orders of magnitude smaller, especially for the large benchmarks were the standard formulation could not be solved within the time limit.

To illustrate the effects of the different optimizations, we give more detailed results for crowds5-6 and leader4-5 in Table 3. The left column block lists running times without SCC cuts. The column "–" contains the values without forward and backward cuts, "$\rightarrow$" with forward cuts, "$\leftarrow$" the with backward cuts and "$\leftrightarrow$" with both. The values for the four different combinations of

Table 2: Running times SCIP and CPLEX for computing MCSs

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

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 for crowds5-6. However, they speed up the solution of leader4-5 by a factor of 10^3. The same tendency can be observed for the other crowds and leader instances.

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 both).

Fig. 3 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 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 with a timeout of 2 hours. Also the local search tool

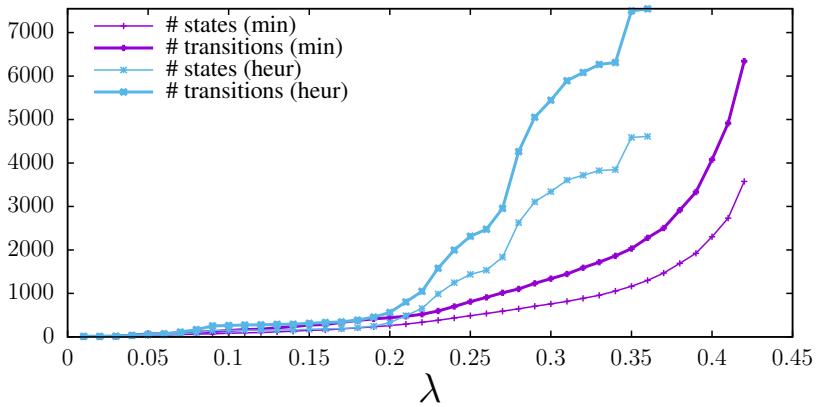

Figure 3: Size of the MCS and heuristically determined critical subsystems for crowds5-6 and different values of λ

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

		no SCC cuts				with SCC cuts			
	Reach	–	$\rightarrow$	$\leftarrow$	$\leftrightarrow$	–	$\rightarrow$	$\leftarrow$	$\leftrightarrow$
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)

ran into a timeout for $\lambda \geq 0.35$, however, without yielding a critical subsystem.

5. Conclusion

In this paper we used MILP solver to compute state-minimal critical subsystems for DTMCs. By adding redundant constraints, which tighten the LP-relaxation of the MILP, the solution process can be speeded up significantly. A topic for future research is to analyze the theoretical complexity of computing MCSs for DTMCs. We conjecture that this problem is NP-complete. We also plan to integrate the MILP approach into the hierarchical counterexample generation tool described in [10].

References

[1] Bustan, D., Rubin, S., Vardi, M.Y.: Verifying ω-regular properties of Markov chains. In: Int'l Conf. on Computer Aided Verification (CAV). Volume 3114 of Lecture Notes in Computer Science, 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: Int'l Conf. on Computer Aided Verification (CAV). Volume 5123 of Lecture Notes in Computer Science, Springer (2008) 162–175

[4] Chadha, R., Viswanathan, M.: A counterexample-guided abstraction-refinement framework for Markov decision processes. ACM Trans. on Computational Logic **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: Int'l Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI). Volume 5403 of Lecture Notes in Computer Science, Springer (2009) 366–380
[8] Andrés, M.E., D'Argenio, P., van Rossum, P.: Significant diagnostic counterexamples in probabilistic model checking. In: Haifa Verification Conference. Volume 5394 of Lecture Notes in Computer Science, 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: Int'l Workshop on Dynamic Aspects in Dependability Models for Fault-Tolerant Systems (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: Int'l Symp. on Automated Technology for Verification and Analysis (ATVA). Volume 6996 of Lecture Notes in Computer Science, Springer (2011) 443–452
[11] Kattenbelt, M., Huth, M.: Verification and refutation of probabilistic specifications via games. In: IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science (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: Int'l Conf. on Concurrency Theory. Volume 5710 of Lecture Notes in Computer Science, 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] Schrijver, A.: Theory of linear and integer programming. Wiley (1986)
[15] Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Int'l Conf. on Computer Aided Verification (CAV). Volume 6806 of Lecture Notes in Computer Science, Springer (2011) 585–591
[16] Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for web transactions. ACM Trans. on Information and System Security **1**(1) (1998) 66–92
[17] Itai, A., Rodeh, M.: Symmetry breaking in distributed networks. Information and Computation **88**(1) (1990) 60–87