Dirac-Based Reduction Techniques for Quantitative Analysis of Discrete-Time Markov Models

Mohammadsadegh Mohagheghi, Behrang Chaboki

Dirac-based Reduction Techniques for Quantitative Analysis of Discrete-time Markov Models

Mohammadsadegh Mohagheghi[1][0000−0001−8059−3691] and Behrang Chaboki[1]

Valie-Asr University of Rafsanjan, Rafsanjan, Iran mohagheghi@vru.ac.ir
https://math.vru.ac.ir/computer-science

Abstract. Iterative methods are widely used in probabilistic model checking to compute quantitative reachability values. Redundant computation is a significant drawback of iterative methods that affects the performance of probabilistic model checking. In this paper we propose a new approach to avoid redundant computations for reachability analysis of discrete-time Markov processes. Redundant computations can be avoided by considering transitions with Dirac distributions. If two states of a Markov chain are connected with a Dirac transition, the iterative method can postpone the update of the source state until the value of the destination state is computed. Using this approach, we propose two heuristics to improve the performance of iterative methods for computing unbounded reachability probabilities in discrete-time Markov chains and Markov decision processes. The proposed heuristics can be lifted to the computations of expected rewards and have been implemented in PRISM model checker. Several standard case studies have been used and experimental results show that our heuristics outperform most well-known previous iterative methods.

Keywords: Probabilistic Model Checking · Quantitative Verification · Markov Decision Processes · Discrete-time Markov Chains.

1 Introduction

Model checking is a formal approach for verifying quantitative and qualitative properties of computer systems. In this way, the system is modelled by a labelled transition system, and its properties are specified in temporal logic. Because of some stochastic behaviours of these systems, we can use probabilistic model checking to analyse the quantitative property specifications of these systems [1–3]. In this domain, we can use discrete and continuous time Markov Chains to model fully probabilistic systems. Besides, Markov Decision Processes [5] are used to model systems with both probabilistic and non-deterministic behaviours. Probabilistic Computation Tree Logic (PCTL) [1] is used to formally specify the related system properties. A main part of PCTL properties against MDPs can be verified by computing the extremal reachability probability: The maximal or minimal probability of reaching a given set of goal states. For quantitative parts,

numerical computations are needed to calculate these reachability probabilities [2, 6]. Linear programming [1, 3], value iteration and policy iteration are well-known numerical approaches for computing the optimal reachability probabilities [2, 5]. PRISM [4] and STORM [7] are examples of probabilistic model checkers that use these numerical methods to compute reachability probabilities.

One of the main challenges of model checking in all variants is the state space explosion problem,i.e., the excessive space requirement to store the states of the model in the memory [1, 9]. For probabilistic model checking, we have the additional difficulty of solving linear programs. For the feasibility of the algorithms we need efficient heuristics to decrease the running time of these algorithms [3]. A wide range of approaches has been proposed for probabilistic model checking in previous works to tackle these problems. Symbolic model checking [9], compositional verification [10], symmetry reduction for probabilistic systems [11], incremental model construction [12] and statistical model checking [6] have been proposed to reduce the needed space for probabilistic model checking. In addition, several approaches are used to accelerate standard algorithms for probabilistic model checking. SCC-based approaches [13, 14] identify strongly connected components (SCCs) of the underlying model and compute reachability probabilities of the states of each component in a right order. Learning based algorithms use the idea of real-time dynamic programming to solve reachability probability problems of MDPs [15]. Prioritization methods focus on finding a good state ordering to update the values of states during iterative computations [13, 14]. The idea of finding Maximal End Components (MECs) is used in [3] to reduce the number of states of the model. Several techniques are proposed in [17] to reduce the size of a DTMC model. These techniques are used to reduce the model for finite-horizon properties.

In this paper, we propose new heuristics to reduce the number of updates and the running time of iterative methods for computing reachability probabilities. Our methods consider the set of transitions that lead to a Dirac distribution (a transition with probability one) and use the fact that any two states of a DTMC that are connected with a Dirac transition have the same (unbounded) reachability probabilities and the difference between their expected rewards equals to the defined reward of the transition. Although our heuristic can be considered as a special case of weak bisimulation, its time complexity is linear in the size of the underlying model. Our experiments show that an iterative method with our heuristic outperforms the one with bisimulation minimization. The main contributions of our work are as follows:

- We first propose a Dirac-based method to avoid redundant updates in the computation of reachability probabilities in DTMCs.
- To use our method for MDPs, we propose it as an extension of modified policy iteration method to compute the extremal reachability probabilities and expected rewards. We also apply the heuristic for SCC-based methods to use the benefits of SCC decomposition approaches.

A main advantage of our heuristic is its time complexity that is linear in the size of model. The remainder of this paper is structured as follows: Section 2

reviews some related definitions. In section 3 we review the probabilistic model checking algorithms. Section 4 proposes the Dirac-based method to improve the iterative computations for reachability probabilities. Section 5 shows the experimental results and Section 6 concludes the paper.

2 Preliminaries

In this section, we provide an overview of DTMCs and MDPs and reachability properties. We mainly follow the notations of [1, 13]. Let S be a countable set. A discrete probability distribution on S is a function $P : S \to [0, 1]$ satisfying $\sum_{s \in S} P(s) = 1$. We use *Dist(S)* as the set of all distributions on S. The support of P is defined as the set $Supp(P) = \{s \in S | P(s) > 0\}$. A distribution P is *Dirac* if $Supp(P)$ has only one member. More details about the proposed definitions in this section and their related probability measures are available in [1, 2, 16].

2.1 Discrete-time Markov Chains

Definition 1. A Discrete-time Markov Chain (DTMC) is a tuple $D = (S, \hat{s}, \mathbf{P}, R, G)$ where S is a countable, non-empty set of *states*, $\hat{s} \in S$ is the *initial state*, $\mathbf{P} : S \times S \to [0, 1]$ is the *probabilistic transition function*, $R : S \times S \to \Re_{\geq 0}$ is a reward function wich assigns to each transition of P a non-negative reward value and $G \subseteq S$ is the set of *Goal* states.

A DTMC D is called finite if S is finite. For a finite D, $size(D)$ is the number of states of D plus the number of transitions of the form $(s, s') \in S \times S$ with $\mathbf{P}(s, s') > 0$. A *path* represents a possible execution of D [2] and is a non-empty (finite or infinite) sequence of states $s_0 s_1 s_2 ...$ such that $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i \geq 0$. We use $Paths_{D,s}$ to denote the set of all paths of D that start in the state s and we use $FPaths_{D,s}$ for the subset of *finite* paths of $Paths_{D,s}$. We also use $Paths_D$ and $FPaths_D$ for $\cup_{s \in S} Paths_{D,s}$ and $\cup_{s \in S} FPaths_{D,s}$ respectively. For a finite path $\pi = s_0 s_1 ... s_k$, the accumulated reward is defined as: $\sum_{i<k} R(s_i, s_{i+1})$. For an infinite path $\pi = s_0 s_1 ...$ and for every $j \geq 0$, let $\pi[j] = s_j$ denote the $(j + 1)$th state of π and $\pi[..j]$ the $(j + 1)$th prefix of the form $s_0 s_1 ... s_j$ of π. We use $pref(\pi)$ as the set of all prefixes of π.

2.2 Probability Measure of a Markov Chain

In order to reason about the behaviour of a Markov chain D, we need to formally use the *cylinder sets* of the finite paths of D [1].

Definition 2. The *Cylinder set* of a finite path $\hat{\pi} \in FPaths_D$ is defined as $Cyl(\hat{\pi}) = \{\pi \in Paths_D | \hat{\pi} \in pref(\pi)\}$. The probability measure Pr^D is defined on the cylinder sets as $Pr^D(Cyl(s_0 ... s_n)) = \prod_{0 \leq i < n} \mathbf{P}(s_i, s_{i+1})$.

2.3 Markov Decision Processes

Markov Decision Processes (MDPs) are a generalization of DTMCs that are used to model systems that have a combination of probabilistic and non-deterministic

behaviour. An MDP is a tuple $M = (S, \hat{s}, Act, \delta, R, G)$ where $S, \hat{s}$ and G are the same as for DTMCs, Act is a finite set of actions, $R : S \times Act \times S \to \Re_{\geq 0}$ is a reward function, assigns to each transition a non-negative reward value and $\delta : S \times Act \to Dist(S)$ is a probabilistic transition function. For every state $s \in S$ of an MDP M one or more actions of Act are defined as enabled actions. We use $Act(s)$ for this set and define it as $Act(s) = \{\alpha \in Act \mid \delta(s, \alpha) \text{ is defined}\}$.

For $s \in S$ and $\alpha \in Act(s)$ we use $Post(s, \alpha)$ for the set of α successors of s, $Post(s)$ for all successors of s and $Pre(s)$ for predecessors of s [1]:

$$Post(s, \alpha) \doteq \{s' \in S \mid \delta(s, \alpha, s') > 0\}, \tag{1}$$

$$Post(s) \doteq \cup_{\alpha \in Act(s)} Post(s, \alpha), \tag{2}$$

To evaluate the operational behaviour of an MDP M we should consider two steps to take a transition from a state $s \in S$. First, one enabled action $\alpha \in Act(s)$ is chosen non-deterministically. Second, according to the probability distribution $\delta(s, \alpha)$, a successor state $s' \in Post(s, \alpha)$ is selected randomly. In this case, $\delta(s, \alpha)(s')$ determines the probability of a transition from s to s' by the action $\alpha \in Act(s)$. We extend the definition of *paths* for MDPs: A path in an MDP M is a non-empty (finite or infinite) sequence $\pi = s_0 \overset{\alpha_0}{\to} s_1 \overset{\alpha_1}{\to} ...$ where $s_i \in S$ and $\alpha_i \in Act(s_i)$ and $s_{i+1} \in Post(s_i, \alpha_i)$ for every $i \geq 0$. Similar to the case with DTMCs, for a state $s \in S$, we use $Paths_{M,s}$ to denote the set of all paths of M starting in s and $FPaths_{M,s}$ for all finite paths of it. For reasoning about the probabilistic behaviour of an MDP we use the notion of policy (also called adversary) [2, 3, 6].

Definition 3. A (deterministic) *policy* of an MDP M is a function $\sigma : FPath_M \to Act$ that for every finite path $\pi = s_0 \overset{\alpha_0}{\to} s_1 \overset{\alpha_1}{\to} ... \overset{\alpha_{i-1}}{\to} s_i$ selects an enabled action $\alpha_i \in Act(s_i)$. The policy σ is memoryless if it depends only on the last state of the path. In general, a policy is defined as function of $FPath_M$ to a distribution on Act. However, memoryless and deterministic policies are enough for computing the optimal unbounded reachability probabilities [2]. We use Pol_M for the set of all deterministic and memoryless policies of M. A policy $\sigma \in Pol_M$ resolves all non-deterministic choices in M and induces a DTMC M^σ for which every state is a finite path of M.

Definition 4 Induced DTMC. For an MDP $M = (S, \hat{s}, Act, \delta, R, G)$ and a policy σ, the induced DTMC is $M^\sigma = (FPath_M, \hat{s}, P^\sigma, G')$ where:

- Every path $\pi \in FPath_M$ is a state of M^σ
- For every $\pi = s_0 \overset{\alpha_0}{\to} s_1 \overset{\alpha_1}{\to} ... \overset{\alpha_{n-1}}{\to} s_n, P^\sigma(\pi, \pi \overset{\alpha(s_n)}{\to} s_{n+1}) = \delta(s_n, \sigma(s_n))(s_{n+1})$.
- $G' = \{s_0 s_1 ... s_n \in FPath_M | s_n \in G\}$

2.4 Probability Measure for MDPs

For a policy σ and any infinite path $s_0 \overset{\alpha_0}{\to} s_1 \overset{\alpha_1}{\to} ... \in Paths_M$, we define the following bijection function f to associate the infinite paths in M and M^σ [2]:

$$f(s_0 \overset{\alpha_0}{\to} s_1 \overset{\alpha_1}{\to} ...) \overset{def}{=} (s_0)(s_0 \alpha_0 s_1)...$$

where $\alpha_i = \sigma(s_i)$ for all $i \geq 0$. We can use this function and the probability measure Pr^{M^σ} for the induced DTMC M^σ to define the probability measure Pr^M_σ over $Paths_M$ [1, 3]. We use Pr^M_σ to capture the behaviour of M under σ.

Although the number of states of the induced DTMC D is (countably) infinite, for memoryless policies its state space is isomorphic to S and we can reduce M^σ to an $|S|$-state *quotient DTMC*. The quotient DTMC for an MDP $M = (S, \hat{s}, Act, \delta, R, G)$ and a deterministic finite-memory policy σ is the finite state DTMC $M^\sigma = (S, \hat{s}, \mathbf{P}, R', G)$ where S, G and $\hat{s}$ are the same as in M, and $\mathbf{P} : S \times S \to [0, 1]$ is a transition probability function defined as $\mathbf{P}(s, s') = \delta(s, \sigma(s))(s')$ and R' is a reward function where $R'(s, s') = R(s, \sigma(s))(s')$ [2].

3 Probabilistic Model Checking

The aim of probabilistic model checking is to verify a desired quantitative or qualitative property of the system. A main class of PCTL properties includes reachability probabilities and expected rewards. For DTMCs, a reachability probability is the probability of reaching the set of goal states G and for MDPs, it is the extremal probability of reaching G. Reward-based properties are defined as the expected accumulated reward (extremal expected reward in the case of MDPs) until reaching a goal state [1]. To formally reason about reachability probabilities, we need to define a probability measure on the set of paths that reach to some states in G. For a state $s_0 \in S$, let $reach_{s_0}(G)$ be the set of all paths that start from s_0 and have a state from G:

$$reach_{s_0}(G) \stackrel{def}{=} \{\pi \in Paths_{D,s_0} \mid \pi[i] \in G \text{ for some } i \geq 0\}.$$

The probability measure on the set $reach_{s_0}(G)$ is defined as [1]:

$$Pr^D(reach_{s_0}(G)) = \sum_{s_0 s_1 \ldots s_n \in reach_{s_0}(G)} Pr^D(Cyl(s_0 s_1 \ldots s_n)).$$

For MDPs, reachability probabilities are defined as the extremal probabilities of reaching goal states [13, 16]:

$$Pr^M_{max}(reach_{s_0}(G)) \stackrel{def}{=} sup_{\sigma \in Pol_M} Pr^M_\sigma(reach_{s_0}(G))$$

Reachability probability properties are divided into qualitative and quantitative properties. A qualitative property of a probabilistic system means the probability of reaching the set of goal states is either 0 or 1 [1]. Qualitative verification is a method to find the set of states for which this reachability probability is exactly 0 or 1. We denote these sets by S^0 and S^1, respectively. For the case of MDPs, we are interested to find those states for which the maximum or minimum reachability probability is 0 or 1. These sets are denoted by S^0_{min},S^0_{max},S^1_{min} and S^1_{max} and can be computed by graph-based methods [2].

3.1 Quantitative Properties

In probabilistic model checking, we consider a PCTL property to be quantitative if the probability of reaching goal states is not exactly 0 or 1. Verification of quantitative properties usually reduces to solving a linear time equation system (for DTMCs) or solving a *Bellman* equation for MDPs [3, 5]. For an arbitrary state $s \in S$ in a DTMC D, let x_s be the probability of reaching G from s, i.e., $x_s = Pr^D(reach_s(G))$. The values of x_s for all $s \in S$ are obtained as the unique solution of the *linear equation system* [1, 2]:

$$x_s = \begin{cases} 0 & \text{if} \quad s \in S^0 \\ 1 & \text{if} \quad s \in S^1 \\ \sum_{s' \in S} \mathbf{P}(s, s').x_{s'} & \text{if} \quad s \in S^? \end{cases}$$

where $S^? = S \backslash (S^0 \cup S^1)$. A model checker can use any direct method (e.g. Gaussian elimination) or iterative method (e.g. Jacobi, Gauss-Seidel) to compute the solution of this linear equation system.

For the case of MDPs, we consider x_s as the maximum (or minimum) probability of reaching G, i.e., $x_s = Pr^M_{max}(reach_s(G))$. In this case, the values of x_s are obtained as the solution of the *Bellman equation* system:

$$x_s = \begin{cases} 0 & \text{if} \quad s \in S^0_{max} \\ 1 & \text{if} \quad s \in S^1_{max} \\ \max\limits_{\alpha \in Act(s)} \sum_{s' \in S} \delta(s, \alpha)(s').x_{s'} & \text{if} \quad s \in S^?_{max} \end{cases}$$

where $S^?_{max} = S \backslash (S^0_{max} \cup S^1_{max})$. Using x_s for the maximal expected accumulated reward, we have [1, 2]:

$$x_s = \begin{cases} 0 & \texttt{if } s \in G \\ \infty & \texttt{if } s \notin S^1_{min} \\ \max\limits_{\alpha \in Act(s)} \sum_{s' \in S} (R(s, \alpha, s') + \delta(s, \alpha)(s').x_{s'}) & \texttt{otherwise} \end{cases}$$

Some standard direct algorithms (like Simplex algorithm [11]) are able to compute the precise values for the *Bellman equation*. The main drawback of direct methods is their scalability that limits them to relatively small models [2]. Iterative methods are other alternatives to approximate the values of x_s.

3.2 Iterative Methods for Quantitative Reachability Probabilities

Value iteration (VI) and policy iteration (PI) are two standard iterative methods that are used to compute the quantitative properties in probabilistic systems. VI and its Gauss-Seidel extension are widely studied in the previous works [1, 16, 15, 2, 13]. We review PI, which is used in the remaining of this paper.

Policy Iteration This method iterates over policies in order to find the optimal policy that maximizes reachability probabilities of all states. Starting from an arbitrary policy, it improves policies until reaching no change in them [2].

For each policy, the method uses an iterative method to compute the reachability probability values of quotient DTMCs and updates the value of states of the original MDP. The termination of this method is guaranteed for every finite MDP [2]. Modified policy iteration (MPI) [5] performs a limited number of iterations for each quotient DTMC (100 iterations for example) and updates the policy after doing this number of iterations. Algorithm 1 shows the standard policy iteration to compute $Pr^M_{max}(reach_s(G))$. We consider a policy as a mapping σ from states to actions. In lines 7-9, the algorithm uses a greedy approach to update the policy σ. More details about this algorithm are available in [2].

Algorithm 1 Policy iteration for $P^{max}_s(G)$.

input: an MDP $M = (S, \hat{s}, Act, \delta, G)$
output: Approximation of $Pr^M_{max}(reach_s(G))$ for all $s \in S$
for all $s \in S$ **do**
$x_s \leftarrow \begin{cases} 1, & \text{if } s \in S^1_{max} \\ 0, & \text{otherwise} \end{cases}$
end for
select an arbitrary policy σ;
repeat
Compute $x_s = Pr^M_\sigma(reach_s(G))$ for all $s \in S$;
for all $s \in S^?$ **do**
$\sigma(s) \leftarrow \text{argmax}_{a \in Act(s)} \sum_{s' \in S} \delta(s, a)(s') \times p_{s'}$;
end for
until σ has not changed;

4 Improving Iterative Methods for Quantitative Reachability Probabilities

We use a heuristic to improve the performance of iterative methods for DTMCs. It considers transitions with Dirac distributions. For this class of transitions, the reachability probability values of the source and destination states of the transition are the same. The idea of our heuristic is to avoid useless updates in order to reduce the total number of updates before termination. We use it to improve the performance of policy iteration method for computing reachability probabilities in MDPs. In this case, Dirac transitions are considered to reduce the number of updates in the computations of each quotient DTMCs. We apply this technique on the SCC-based method to improve the performance of this approach. Although the idea of considering Dirac transitions has been used as a reduction technique in [8], it has been only proposed for the case of deterministic Dirac transitions. The main contribution of our work is to use this heuristic for improving the MPI method for both reachability probabilities and expected rewards to cover nondeterministic action selections. To the bet of our knowledge, none of state of the art model checkers support this technique.

4.1 Using Transitions with Dirac Probability

In our work, we consider transitions with Dirac distribution (which we call Dirac transitions) to avoid redundant updates of state values. This type of transitions is also used in statistical model checking [6] but we use it to accelerate iterative methods. For a DTMC D, a Dirac transition is a pair of states s and s' with $\mathbf{P}(s, s') = 1$. Based on the definition, for this pair of states, we have $Pr^D(reach_s(G)) = Pr^D(reach_{s'}(G))$. As a result, we can ignore this transition and postpone the update of the value of s until the convergence of the value of s'. Our approach updates the value of s only once and avoids redundant updates. However, there may be some incoming transitions to s that need the value of s in every iteration. Consider Fig. 1 and suppose s is the target state of a transition of the form (t, s). In this case the value of s affects the value of t and we should modify the incoming transitions to s to point to s'. Algorithm 2 shows

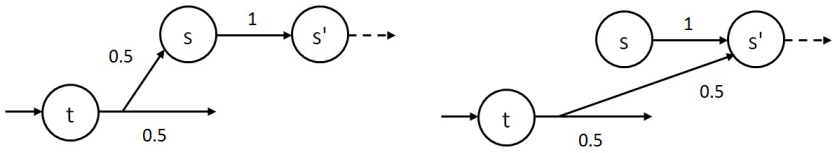

Fig. 1. An example of a Dirac transition (left) and the modified DTMC (right)

details of our approach to remove Dirac transitions from a DTMC D and reduce it to smaller DTMC D'. The main idea of this algorithm is to partition the state space of D to *Dirac-based* classes. A *Dirac-based* class is the set of states that are connected by Dirac transitions. The algorithm uses the DBC array to partition $S^?$ to the related classes. Consider for example a sequence of states of the form $s_i, s_j, ..., s_{k-1}, s_k$ where there exists Dirac transitions between each two consecutive states and the outgoing transition of s_k is not a Dirac one or s_k is one of $s_i, ..., s_{k-1}$. We put these states into one class and set their DBC to s_k. The reachability probability of all states of this class are equal to the reachability probability of s_k. Formally, for each state $s_i \in S$, we define $DBC[s_i]$ as the index of the last state in the sequence of states that are connected with $Dirac$ transitions. For every state $s \in S$ we have $x_s = x_{DBC[s]}$. Algorithm 2 initiates the values of this array in lines 3-5. For every state $s_i \in S$ that $DBC[s_i]$ is not determined previously, the algorithm calls the $Find\_Dirac\_Index$ function to determine its DBC. After determining the DBC value of all states of S, the algorithm creates the reduced DTMC D' (lines 9-13). The states of D' are those states $s_i \in S$ for which $DBC[s_i] = i$. According to the definition of S^1, for each state $s \in G$ and each $s' \in S$ if $DBC[s'] = DBC[s]$ then we have $s' \in G$.

Algorithm 3 shows details of the $Find\_Dirac\_Index$ function to determine the DBC value of a state s_i. It first checks the number of outgoing transitions of s_i. If there is more than one outgoing transition, the state should be considered

Algorithm 2 Dirac transition reduction in a DTMC D

1: **input:** a DTMC $D = (S, \hat{s}, \mathbf{P}, G)$
2: **output:** $P_s^{max}(F)$ for all $s \in S^?$
3: **for all** $s_i \in S^?$ **do**
4: $DBC[s_i] = -1$;
5: **end for**
6: **for all** $s_i \in S^?$ **do**
7: **if** $DBC[s_i] = -1$ **then**
8: $DBC[s_i] = Find\_Dirac\_Index(D, DBC, s_i)$;
9: **end if**
10: **end for**
11: Create reduced DTMC $D' = \{S', \hat{s}, \mathbf{P}', G\}$ where:
12: $S' = \{s_i \in S | DBC[s_i] = i\}$;
13: **for all** $(s, s') \in S'$ **do**
14: $\mathbf{P}'(s, s') = \sum_{\{s'' \in S | DBC[s'] = DBC[s']\}} \mathbf{P}(s, s'')$
15: **end for**
16: return D';

in the reduced DTMC D' and its DBC value is set to the index of the state (line 11). If there is only one outgoing transition (a Dirac one of the form (s_i, s_j)), the algorithm checks the DBC value of $s_j \in Post(s_i)$ and recursively computes the DBC values of the sequence of states that are connected via Dirac transitions.

Algorithm 3 Computing $Find\_Dirac\_Index(s_i)$

1: **input:** a DTMC $D = (S, \hat{s}, \mathbf{P}, G)$, a DBC array and a state $s_i \in S$
2: **output:** $DBC[s_i]$;
3: **if** there is only one state s_j in $Post(s_i)$ and s_j is unmarked **then**
4: **if** $DBC[s_j] = -1$ **then**
5: Mark s_j;
6: $DBC[s_i] = Find\_Dirac\_Index(D, DBC, s_j)$;
7: **else**
8: $DBC[s_i] = DBC[s_j]$;
9: **end if**
10: **else**
11: $DBC[s_i] = i$;
12: **end if**
13: return $DBC[s_i]$;

The following lemma shows that the reachability probabilities of the states of D' are the same as the values of the corresponding states of D. As a result, the iterative method can be applied on D' which may be smaller than D.

Lemma 1: For each state $s_i \in S^?$ with $DBC[s_i] = i$ we have: $Pr^{D'}(reach_{s_i}(G)) = Pr^{D}(reach_{s_i}(G))$ where D' is the Dirac-based reduced DTMC.

Proof: Recall that for each $s_i \in S^?$ we have

$$x_{s_i} = Pr^D(reach_{s_i}(G)) = \sum_{s' \in Post(s_i)} P(s_i, s').x_{s'} \tag{3}$$

Suppose $Post(s_i) = \{s'_1, s'_2, ..., s'_k\}$. First consider a case where for each two states $s'_i, s'_j \in Post(s_i)$ we have $DBC[s_i] \neq DBC[s_j]$ and as a result, for each $s'_j \in Post(s_i)$ we have $P'(s_i, DBC[s'_j]) = P(s_i, s'_j)$. We can rewrite equation (4) as:

$$x_{s_i} = P(s_i, s'_1).x_{s'_1} + ... + P(s_i, s'_k).x_{s'_k} = P(s_i, s'_1).x_{DBC[s'_1]} + ... + P(s_i, s'_k).x_{DBC[s'_k]}$$

$$= P'(s_i, DBC[s'_1]).x_{DBC[s'_1]} + ... + P'(s_i, DBC[s'_k]).x_{DBC[s'_k]} = Pr^{D'}(reach_{s_i}(G))$$

Now consider a case that $Post(s_i) = \{s'_{11}, s'_{12}, ..., s'_{1k}, ..., s'_{l1}, ..., s'_{lm}\}$ where the indexes are so that for each s'_{ij} and s'_{pq} we have $DBC[s'_{ij}] = DBC[s'_{pq}]$ if $i = p$ and $DBC[s'_{ij}] \neq DBC[s'_{pq}]$ if $i \neq p$. Using the definition of P' we have:

$$x_{s_i} = P(s_i, s'_{11}).x_{s'_{11}} + ... + P(s_i, s'_{1k}).x_{s'_{1k}} + ... +$$

$$P(s_i, s'_{l1}).x_{s'_{l1}} + ... + P(s_i, s'_{lm}).x_{s'_{lm}} =$$

$$P(s_i, s'_{11}).x_{DBC[s'_{11}]} + ... + P(s_i, s'_{1k}).x_{DBC[s'_{1k}]} + ... +$$

$$P(s_i, s'_{l1}).x_{DBC[s'_{l1}]} + ... + P(s_i, s'_{lk}).x_{DBC[s'_{lk}]} =$$

$$[P(s_i, s'_{11}) + ... + P(s_i, s'_{1k})].x_{DBC[s'_{11}]} + ... +$$

$$[P(s_i, s'_{l1}) + ... + P(s_i, s'_{lk})].x_{DBC[s'_{l1}]} =$$

$$P'(s_i, DBC[s'_{11}]).x_{DBC[s'_{11}]} + ... + P'(s_i, DBC[s'_{l1}]).x_{DBC[s'_{l1}]} = Pr^{D'}(reach_{s_i}(G)).$$

■

The time complexity of algorithm 2 is linear in the size of the DTMC D. For every state $s \in S^?$, the $Find\_Dirac\_Index$ function is called once and performs a check on the number of outgoing transitions of s and its DBC value. For creating the reduced DTMC D', Algorithm 2 checks every state and transitions of D only once. As a result the time complexity of Algorithm 2 is in $O(size(D))$.

The iterative method (Gauss-Seidel for example) can then compute the probability values of the states of D'. Algorithm 4 shows the way that the Dirac-based DTMC-reduction technique is used to accelerate the computations of reachability probabilities.

It first computes the reduced DTMC D' and then computes reachability probabilities of states of D'. Finally, in lines 3-5, it updates the value of the remaining states of D (those states that are not in D'). For the correctness of Algorithm 4, we compare the precision of the computed values of this algorithm with the computed values of Gauss-Seidel algorithm for DTMCs. We suppose

Algorithm 4 Dirac-Based Reduction Technique for Computing the Reachability Probabilities of DTMCs

1: **input:** a DTMC $D = (S, \hat{s}, \mathbf{P}, G)$
2: **output:** Reachability probabilities of the states of D;
3: Apply Algorithm 2 on D and compute the reduced DTMC D';
4: Apply Gauss-Seidel method for approximating reachability probabilities of S' in D';
5: **for all** $s \in S^?$ **do**
6: $\quad x_s = x_{DBC[s]}$;
7: **end for**
8: return $(x_s)_{s \in S}$;

that both algorithms use the same state ordering, i.e., for each i, j they update s_i before s_j if $i < j$. In this case we have

$$x^k_{s_i} = \sum_{s_j \in Post(s_i), j<i} \mathbf{P}(s_i, s_j) \times x^k_{s_j} + \sum_{s_j \in Post(s_i), j>i} \mathbf{P}(s_i, s_j) \times x^{k-1}_{s_j} \quad (4)$$

Lemma 2: Consider a DTMC D, its Dirac-based reduced DTMC D' and a state $s_i \in S$. Let $x^k_{s_i}$ be the computed reachability probability of s_i after k iteration of Gauss-Seidel method on D and $x'^k_{s_i}$ be the computed value of $s_{DBC[s_i]}$ after k iteration of Gauss-Seidel method on D'. For every $k \geq 0$ we have $x'^k_{s_i} \geq x^k_{s_i}$. *Proof Sketch:* By induction on k and the fact that for each state $s_j \in S$ we have $x^k_{s_j} \leq x^k_{DBC[s_j]}$. ■

According to Lemma 2, the precision of the computed values of Algorithm 4 is the same or better than the precision of the computed values of standard Gauss-Seidel if we use the same number of iterations and the same state ordering in both methods. Depending to the structure of the DTMC D, either of two algorithms may converge faster than the other algorithm. Faster convergence may cause non-precise results for some values. In general, standard iterative methods does not guarantee the convergence of values [16].

4.2 Improving Iterative methods for Computing Reachability Probabilities in MDPs

The Dirac-based reduction technique can be also used for MDPs to avoid redundant updates. However, it can not be directly applied for states with multiple enabled actions. To cover this case and as a novelty of our approach, we use the MPI method. We apply our heuristic for every quotient DTMC to reduce the number of states that should be updated. We propose our it as an extension of SCC-based techniques. We use MPI to compute reachability values of the states of each SCC and apply our Dirac-based heuristic to accelerate the iterative computations of each quotient DTMC. Algorithm 5 presents the overall idea of our method for accelerating SCC-based methods for MDPs. The correctness of this approach relies on the correctness of SCC decomposition for MDPs [13] and the correctness of our Dirac-based DTMC reduction method (Lemma 1).

Algorithm 5 SCC-based method with Dirac-based reductions for computing reachability probabilities.

1: **input:** an MDP $M = (S, \hat{s}, Act, \delta, G)$
2: **output:** $Pr^M_{max}(reach_s(G))$ for all $s \in S$;
3: Compute Strongly Connected Components of M and order them by the reverse topological order $C_1, ..., C_k$
4: **for** `i = 1 to k do` **do**
5: Select a policy σ of C_i
6: **repeat**
7: Use σ and compute the quotient DTMC D_i of C_i ;
8: Compute Dirac-based reduced DTMC D'_i;
9: Use an iterative method to update the value of states of D'_i.
10: Update the remaining states of D_i (like lines 3-5 of algorithm 5).
11: Update σ according to a greedy approach.
12: **until** σ has changed;
13: **end for**
14: return $(x_{s_i})_{s_i \in S^?}$;

4.3 Accelerating Iterative methods for Computing Expected Rewards in MDPs

The standard iterative methods can be used to approximate the Bellman equation for extremal expected rewards. In this case, the initial vector of values is set to zero for all states. Value iteration (or policy iteration) should also consider the defined reward of each action in the update of values of each state. More details about iterative methods for expected rewards are available in [1, 2, 16]. To use our heuristic for expected rewards, we use the fact that for every Dirac transition of the form (s_i, s_j) we have $x_{s_i} = x_{s_j} + R(x_i)$ where x_i and x_j are the expected values for these two states and $R(x_i)$ is the reward for x_i. In this case, an iterative method does not need to update the value of x_i in every iteration. Similar to the case for the reachability probabilities, the incoming transitions to x_i should be modified to point to x_j. In addition the reward of a modified transition is modified by adding the reward of the related Dirac transition.

5 Implementation and Experimental Results

We implemented our heuristics as a package in the PRISM model checker. Our implementations are based on sparse engine of PRISM [9] which is developed in C and are available in [18]. We used several standard case studies which have been used in previous works [2, 4, 10, 13, 15, 16]. We compare the running time of our heuristics with the running time of well-known previous methods. We only focus on the running time of the iterative computations for quantitative properties. We exclude the running times for model constructions that are the same for all methods. The running time of the graph-based pre-computation are negligible in appropriate implementations [7]. For SCC-based methods and

our Dirac-based ones, the reported times include the running times for SCC decomposition and Dirac-based reductions. All benchmarks have been run on a machine with Corei7 CPU (2.8GHz , 4 main cores) and 8GB RAM running Ubuntu 18. We use *Consensus*, *Zeroconf*, *firewire_abstract*, brp, *nand* and *Crowds* case studies for comparing the performance of iterative methods for reachability probabilities and use *Wlan*, *CSMA*, *Israeli-jalefon* and *Leader* cases for expected rewards. These case studies are explained in [4, 15, 16]. More details about our experiments and model paratmeters are available as log-files in [18]. Although there are several other standard case studies, their graphical structure do not have any cycle and a topological backward iterative method can be used to computed their underlying properties in linear time [14, 13]. Hence, we focus on the case studies of Table 1, where have several non-trivial cycles.

We consider the running time of the standard iterative methods and well known improved techniques from previous works. To perform a fair comparison, we use sparse engine of PRISM for all experiments and we also implemented topological (SCC-based) method for this engine. In this case, we use MPI to solve each SCC. For learning-based methods, we use the implementation that is proposed in [15] and for backward value iteration, we implement the proposed method from [14]. For all case studies, we consider $\epsilon = 10^{-8}$ as the threshold.

Table 1 shows the running time of the iterative methods for MDP models. For the SCC-based method, we use MPI to approximate the reachability probability values of each SCCs. All times are in seconds. We use * where a method does not terminate after an hour. For *consensus*, *Israeli-jalefon*, *Leader* and *Wlan* models, the running time of SCC-based method is less than the others. In these cases, we use our Dirac-based method to reduce the running time of the computations for each SCC. The results show that for these two classes, our technique is faster than the other methods. The learning-based method is faster than other methods for *zeroconf* cases with N=20. For *firewire* case studies, SCC based and backward value iteration methods are much faster than the standard iterative methods. In this case Dirac-based method for MPI (without SCC decomposition) works better than the other methods.

Table 2 shows the results of our experiments for DTMC case studies. We present the running time of Jacobi and Gauss-Seidel as the standard methods and SCC-based and backward Gauss-Seidel as the improved methods from the previous works. We also use the STORM model checker with bisimulation minimization technique to compare its impact on the running time of computations. The results of Table 1 and Table 2 show that our proposed method outperforms all previous standard methods and most improved ones.

6 Conclusion

In this paper, we proposed Dirac-based heuristics to accelerate iterative methods for computing reachability probabilities and expected rewards. These heuristics have been proposed to avoid redundant updates and multiplications of state values. They can be used for the Gauss-Seidel method for DTMCs or MPI for

Table 1. Running times of the iterative methods for quantitative properties of MDPs.

Model	GS -VI	MPI	Learning-based	Backward-VI	SCC-based GS	SCC-based MPI	Dirac-based
Consensus 4, 20	354.7	190.2	*	273	27.7	38.25	13.12
Consensus 4, 36	1812	968.6	*	1376	111.7	226.5	76.55
Consensus 5, 15	1966	441.3	*	1421	183.3	101.1	35.61
Consensus 5, 32	*	*	*	*	1327	2048	706
Consensus 6, 8	2680	1082	*	1783	430.3	281.6	101.2
Zeroconf 14 (N=20)	16.77	41.16	2.23	14.45	12.67	51.91	5.2
Zeroconf 16 (N=20)	20.11	41.84	2.3	16.11	13.8	71.54	6.92
Zeroconf 18 (N=20)	22.6	52.63	2.47	17.7	14.12	81.89	8.07
Zeroconf 14 (N=5000)	27.9	58.52	731.2	23.71	12.73	69.9	6.54
Zeroconf 16 (N=5000)	32.74	62.70	854.3	25.9	13.85	79.0	6.83
Zeroconf 18 (N=5000)	36.46	64.31	878.4	27.54	14.57	104.46	12.74
firewire_abst 7000,128	473.3	260.6	742.9	1.17	17.14	5.34	0.52
firewire_abst 9500,64	585.5	323.6	574.4	1.23	22.4	4.82	0.44
firewire_abst 9500,128	887.7	487.5	917.6	1.42	27.39	O.M	0.52
Wlan 5,1500	56.76	42.99	*	27.18	6.76	1.61	1.15
Wlan 5,3000	99.07	74.75	*	53.82	16.62	2.62	2.06
Wlan 6,250	160.1	121.5	*	97.5	12.17	3.65	1.44
Wlan 6,1200	257.3	195.9	*	182.4	22.9	4.82	2.67
CSMA 4,3	32.1	32.5	*	15.4	10.6	4.66	4.17
Israeli-jalefon-20	117	47.2	*	39.3	13.6	2.88	2.26
Israeli-jalefon-22	203	71.8	*	55.7	24.9	13.1	11.7
Leader 7	3.03	1.11	*	2.45	0.95	0.67	0.13
Leader 8	22.4	6.49	*	13.7	8.41	6.59	2.13

MDPs. Experiments show promising results. In most cases, our heuristics reduce the running time of iterative methods to less than 50% of the running time of the best previous methods. The extension of the proposed heuristics for other classes of properties (LTL properties for example) is an interesting direction for future works. The Dirac-based method reduces data dependency and can be used for parallel version of the Gauss-Seidel and MPI methods.

References

1. Baier, C., Katoen, J.P.: Principles of model checking. MIT press, 2008.
2. Forejt V., Kwiatkowska MZ., Norman G., Parker D.,: Automated Verification Techniques for Probabilistic Systems, In: SFM, Vol. 11, pp. 53-113. (2011)
3. Baier C., de Alfaro L., Forejt V., Kwiatkowska M.: Probabilistic Model Checking, Dependable Software Systems Engineering, Vol.45 pp. 1-23. (2016)
4. Kwiatkowska M., Norman G., Parker D.: The PRISM benchmark suite, In: 9th International Conference on Quantitative Evaluation of SysTems, IEEE CS press, pp. 203-204. (2012)

Table 2. Running Times of computations for reachability probabilities of DTMCs.

Model	Jacobi	Gauss-Seidel	SCC-based	Backward	Dirac-based	Bisimulation
brp 4,8192	296	137	13.2	32.7	6.14	453
brp 6,4096	104	47.3	7.5	19	4.1	349
brp 6,8192	415	233	28.5	36.2	11.7	O.M
crowds 10, 10	32.07	11.67	21.8	7.48	2.46	5.58
crowds 12, 10	150.89	49.26	*	41.8	11.49	27.3
crowds 9, 15	61.53	19.66	37.14	12.55	5.00	O.M
nand 5, 60	620.7	554.9	8.7	23.72	143.25	244.6
nand 6, 50	364.8	326.2	4.83	11.22	83.60	246.1
nand 6, 60	885.3	797.2	*	38.56	204.1	247.2

5. Puterman ML. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons. (2014)
6. Hartmanns, A.: On the analysis of stochastic timed systems. PhD thesis, Saarland University (2015)
7. Dehnert, C., Junges, S., Katoen, J.P., Volk, M.: A Storm is coming: A modern probabilistic model checker. In Proc. 29th International Conference on Computer Aided Verification (CAV) (2017).
8. D'Argenio, R., Jeannet, B., Jensen, H.E., Larsen, K.G.: Reduction and Refinement Strategies for Probabilistic Analysis. PAPM-PROBMIV pp. 57-76. (2002)
9. Parker, D.A.: Implementation of symbolic model checking for probabilistic systems. PhD thesis, University of Birmingham. (2003)
10. Feng, L.: On learning assumptions for Compositional verification of probabilistic systems. PhD thesis, University of Oxford. (2013)
11. Kwiatkowska. M., Norman. G., Parker. D.: Symmetry reduction for probabilistic model checking. Computer Aided Verification (CAV), pp. 234-248. (2006)
12. Ujma, M.: On Verification and Controller Synthesis for Probabilistic Systems at Runtime. PhD thesis, University of Oxford, (2015)
13. Kwiatkowska, M., Parker, D., Qu, H.: Incremental quantitative verification for Markov decision processes. In Dependable Systems & Networks (DSN), IEEE/IFIP 41st International Conference on, pp.359-370. IEEE. (2011)
14. Ciesinski, F., Baier, C., Groesser, M., Klein, J. Reduction techniques for model checking Markov decision processes. In: Quantitative Evaluation of Systems. Fifth International Conference on, IEEE. pp. 45-54. (2008)
15. Brázdil, T., Chatterjee, K., Chmelik, M., Forejt, V., Kretínský, J., Kwiatkowska, M., Parker, D., Ujma, M. Verification of Markov decision processes using learning algorithms. In 12th International Symposium on Automated Technology for Verification and Analysis (ATVA 14), volume 8837 of LNCS. pp. 98-114. (2014)
16. Baier, C., Klein, J., Leuschner, L., Parker, D., Wunderlich, S. Ensuring the reliability of your model checker: Interval iteration for Markov Decision Processes. In International Conference on Computer Aided Verification, Springer, Cham, pp.160-180. (2017)
17. Kamaleson, N.: Model reduction techniques for probabilistic verification of Markov chains. PhD thesis, University of Oxford, (2018)
18. https://github.com/sadeghrk/prism/tree/DiracBased-Improving