Deciding Probabilistic Bisimilarity Distance One for Labelled Markov Chains

Qiyi Tang(✉) and Franck van Breugel

DisCoVeri Group, York University, Toronto, Canada
{qiyitang,franck}@eecs.yorku.ca

Abstract. Probabilistic bisimilarity is an equivalence relation that captures which states of a labelled Markov chain behave the same. Since this behavioural equivalence only identifies states that transition to states that behave exactly the same with exactly the same probability, this notion of equivalence is not robust. Probabilistic bisimilarity distances provide a quantitative generalization of probabilistic bisimilarity. The distance of states captures the similarity of their behaviour. The smaller the distance, the more alike the states behave. In particular, states are probabilistic bisimilar if and only if their distance is zero. This quantitative notion is robust in that small changes in the transition probabilities result in small changes in the distances.

During the last decade, several algorithms have been proposed to approximate and compute the probabilistic bisimilarity distances. The main result of this paper is an algorithm that decides distance one in $O(n^2 + m^2)$, where n is the number of states and m is the number of transitions of the labelled Markov chain. The algorithm is the key new ingredient of our algorithm to compute the distances. The state of the art algorithm can compute distances for labelled Markov chains up to 150 states. For one such labelled Markov chain, that algorithm takes more than 49 h. In contrast, our new algorithm only takes 13 ms. Furthermore, our algorithm can compute distances for labelled Markov chains with more than 10,000 states in less than 50 min.

Keywords: Labelled Markov chain · Probabilistic bisimilarity
Probabilistic bisimilarity distance

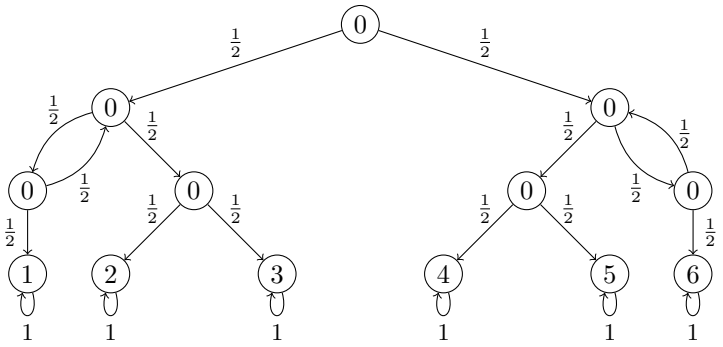
1 Introduction

A *behavioural equivalence* captures which states of a model give rise to the same behaviour. Bisimilarity, due to Milner [22] and Park [25], is one of the best known behavioural equivalences. Verifying that an implementation satisfies a specification boils down to checking that the model of the implementation gives rise to the same behaviour as the model of the specification, that is, the models are behavioural equivalent (see [1, Chap. 3]).

In this paper, we focus on models of probabilistic systems. These models can capture randomized algorithms, probabilistic protocols, biological systems and

H. Chockler and G. Weissenbacher (Eds.): CAV 2018, LNCS 10981, pp. 681–699, 2018.
https://doi.org/10.1007/978-3-319-96145-3_39

many other systems in which probabilities play a central role. In particular, we consider *labelled Markov chains*, that is, Markov chains the states of which are labelled.



The above example shows how the behaviour of rolling a die can be mimicked by flipping a coin, an example due to Knuth and Yao [19]. Six of the states are labelled with the values of a die and the other states are labelled zero. In this example, we are interested in the labels representing the value of a die. As the reader can easily verify, the states with these labels are each reached with probability $\frac{1}{6}$ from the initial, top most, state. In general, labels are used to identify particular states that have properties of interest. As a consequence, states with different labels are not behaviourally equivalent.

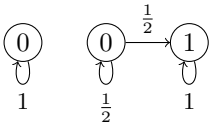
Probabilistic bisimilarity, due to Larsen and Skou [21], is a key behavioural equivalence for labelled Markov chains. As shown by Katoen et al. [16], minimizing a labelled Markov chain by identifying those states that are probabilistic bisimilar speeds up model checking. Probabilistic bisimilarity only identifies those states that behave exactly the same with exactly the same probability. If, for example, we replace the fair coin in the above example with a biased one, then none of the states labelled with zero in the original model with the fair coin are behaviourally equivalent to any of the states labelled with zero in the model with the biased coin. Behavioural equivalences like probabilistic bisimilarity rely on the transition probabilities and, as a result, are sensitive to minor changes of those probabilities. That is, such behavioural equivalences are not robust, as first observed by Giacalone et al. [12].

The *probabilistic bisimilarity distances* that we study in this paper were first defined by Desharnais et al. in [11]. Each pair of states of a labelled Markov chain is assigned a distance, a real number in the unit interval $[0, 1]$. This distance captures the similarity of the behaviour of the states. The smaller the distance, the more alike the states behave. In particular, states have distance zero if and only if they are probabilistic bisimilar. This provides a quantitative generalization of probabilistic bisimilarity that is robust in that small changes in the transition probabilities give rise to small changes in the distances. For example, we can model a biased die by using a biased coin instead of a fair coin in the above example. Let us assume that the odds of heads of the biased coin, that is, going to the left, is $\frac{51}{100}$. A state labelled zero in the model of the fair die

has a *non-trivial* distance, that is, a distance greater than zero and smaller than one, to the corresponding state in the model of the biased die. For example, the initial states have distance about 0.036. We refer the reader to [7] for a more detailed discussion of a similar example.

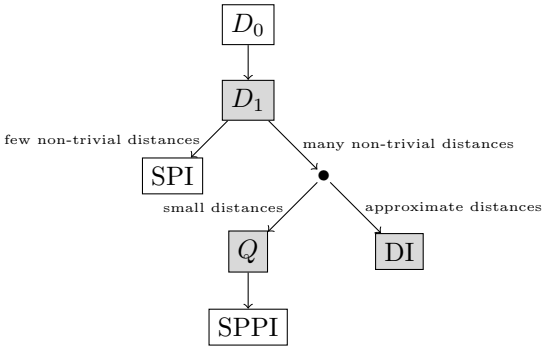
As we already mentioned earlier, behavioural equivalences can be used to verify that an implementation satisfies a specification. Similarly, the distances can be used to check how similar an implementation is to a specification. We also mentioned that probabilistic bisimilarity can be used to speed up model checking. The distances can be used in a similar way, by identifying those states that behave almost the same, that is, have a small distance (see [3,23,26]).

We focus in this paper on computing the probabilistic bisimilarity distances. In particular, we present a *decision procedure* for *distance one*. That is, we compute the set of pairs of states that have distance one. Recall that distance one is the maximal distance and, therefore, captures that states behave very differently. States with different labels have distance one. However, also states with the same label can have distance one, as the next example illustrates.



Instead of computing the set of state pairs that have distance one, we compute the complement, that is, the set of state pairs with distance smaller than one. Obviously, the set of state pairs with distance zero is included in this set. First, we decide distance zero. As we mentioned earlier, distance zero coincides with probabilistic bisimilarity. The first decision procedure for probabilistic bisimilarity was provided by Baier [4]. More efficient decision procedures were subsequently proposed by Derisavi et al. [10] and also by Valmari and Franceschinis [30]. The latter two both run in $O(m \log n)$, where n and m are the number of states and transitions of the labelled Markov chain. Subsequently, we use a traversal of a directed graph derived from the labelled Markov chain. This traversal takes $O(n^2 + m^2)$.

The decision procedures for distance zero and one can be used to compute or approximate probabilistic bisimilarity distances as indicated below.



Once we have computed the sets D_0 and D_1 of state pairs that have distance zero or one, we can easily compute the number of state pairs with non-trivial distances. If the number of non-trivial distances is small, then we can use the *simple policy iteration* (SPI) algorithm due to Bacci et al. [2] to compute those distances. Otherwise, we can either compute all distances smaller than a chosen $\varepsilon > 0$ or we can approximate the distances up to some chosen accuracy $\alpha > 0$. In the former case, we first compute a query set Q of state pairs that contains all state pairs the distances of which are at most ε. Subsequently, we apply the *simple partial policy iteration* (SPPI) algorithm due to Bacci et al. [2] to compute the distances for all state pairs in Q. In the latter case, we start with a pair of distance functions, one being a lower-bound and the other being an upper-bound of the probabilistic bisimilarity distances, and iteratively improve the accuracy of those until they are α close. We call this new approximation algorithm *distance iteration* (DI) as it is similar in spirit to Bellman's value iteration [5].

Chen et al. [8] presented an algorithm to compute the distances by means of Khachiyan's ellipsoid method [17]. Though the algorithm is polynomial time, in practice it is not as efficient as the policy iteration algorithms (see the examples in [28, Sect. 8]). The state of the art algorithm to compute the probabilistic bisimilarity distances consists of two components: D_0 and SPI. To compare this algorithm with our new algorithm consisting of the components D_0, D_1 and SPI, we implemented all the components in Java and ran both implementations on several labelled Markov chains. These labelled Markov chains model randomized algorithms and probabilistic protocols that are part of the distribution of probabilistic model checkers such as PRISM [20]. Whereas the original state of the art algorithm can handle labelled Markov chains with up to 150 states, our new algorithm can handle more than 10,000 states. Furthermore, for one such labelled Markov chain with 150 states, the original algorithm takes more than 49 h, whereas our new algorithm takes only 13 ms. Also, the new algorithm consisting of the components D_0, D_1, Q and SPPI to compute only small distances along with the new algorithm consisting of the components D_0, D_1 and DI to approximate the distances give rise to even less execution times for a number of the labelled Markov chains.

The main contributions of this paper are

- a polynomial decision procedure for distance one,
- an algorithm to compute the probabilistic bisimilarity distances,
- an algorithm to compute those probabilistic bisimilarity distances smaller than some given $\varepsilon > 0$, and
- an approximation algorithm to compute the probabilistic bisimilarity distances up to some given accuracy $\alpha > 0$.

Furthermore, by means of experiments we have shown that these three new algorithms are very effective, improving significantly on the state of the art.

2 Labelled Markov Chains and Probabilistic Bisimilarity Distances

We start by reviewing the model of interest, labelled Markov chains, its most well known behavioural equivalence, probabilistic bisimilarity due to Larsen and Skou [21], and the probabilistic bisimilarity pseudometric due to Desharnais et al. [11]. We denote the set of rational probability distributions on a set S by $\mathrm{Distr}(S)$. For $\mu \in \mathrm{Distr}(S)$, its support is defined by $\mathrm{support}(\mu) = \{\, s \in S \mid \mu(s) > 0 \,\}$. Instead of $S \times S$, we often write S^2.

Definition 1. *A labelled Markov chain is a tuple $\langle S, L, \tau, \ell \rangle$ consisting of*

- *a nonempty finite set S of states,*
- *a nonempty finite set L of labels,*
- *a transition function $\tau : S \to \mathrm{Distr}(S)$, and*
- *a labelling function $\ell : S \to L$.*

For the remainder of this section, we fix such a labelled Markov chain $\langle S, L, \tau, \ell \rangle$.

Definition 2. *Let μ, $\nu \in \mathrm{Distr}(S)$. The set $\Omega(\mu, \nu)$ of couplings of μ and ν is defined by*

$$\Omega(\mu,\nu) = \left\{ \omega \in \mathrm{Distr}(S^2) \;\middle|\; \begin{array}{l} \forall s \in S : \sum_{t \in S} \omega(s,t) = \mu(s) \wedge \\ \forall t \in S : \sum_{s \in S} \omega(s,t) = \nu(t) \end{array} \right\}.$$

Note that $\omega \in \Omega(\mu, \nu)$ is a joint probability distribution with marginals μ and ν. The following proposition will be used to prove Proposition 5.

Proposition 1. *For all μ, $\nu \in \mathrm{Distr}(S)$ and $X \subseteq S^2$,*

$$\forall \omega \in \Omega(\mu,\nu) : \mathrm{support}(\omega) \subseteq X \text{ if and only if } \mathrm{support}(\mu) \times \mathrm{support}(\nu) \subseteq X.$$

Definition 3. *An equivalence relation $R \subseteq S^2$ is a probabilistic bisimulation if for all $(s,t) \in R$, $\ell(s) = \ell(t)$ and there exists $\omega \in \Omega(\tau(s), \tau(t))$ such that $\mathrm{support}(\omega) \subseteq R$. Probabilistic bisimilarity, denoted $\sim$, is the largest probabilistic bisimulation.*

The probabilistic bisimilarity pseudometric of Desharnais et al. [11] maps each pair of states of a labelled Markov chain to a distance, an element of the unit interval $[0,1]$. Hence, the pseudometric is a function from S^2 to $[0,1]$, that is, an element of $[0,1]^{S^2}$. As we will discuss below, it can be defined as a fixed point of the following function.

Definition 4. *The function $\Delta : [0,1]^{S^2} \to [0,1]^{S^2}$ is defined by*

$$\Delta(d)(s,t) = \begin{cases} 1 & \text{if } \ell(s) \neq \ell(t) \\ \min\limits_{\omega \in \Omega(\tau(s),\tau(t))} \sum\limits_{u,v \in S} \omega(u,v)\, d(u,v) & \text{otherwise} \end{cases}$$

Since a concave function on a convex polytope attains its minimum (see [18, p. 260]), the above minimum exists. We will use this fact in Proposition 4, one of the key technical results in this paper. We endow the set $[0,1]^{S^2}$ of functions from S^2 to $[0,1]$ with the following partial order: $d \sqsubseteq e$ if $d(s,t) \leq e(s,t)$ for all $s, t \in S$. The set $[0,1]^{S^2}$ together with the order $\sqsubseteq$ form a complete lattice (see [9, Chap. 2]). The function Δ is monotone (see [6, Sect. 3]). According to the Knaster-Tarski fixed point theorem [29, Theorem 1], a monotone function on a complete lattice has a least fixed point. Hence, Δ has a least fixed point, which we denote by $\boldsymbol{\mu}(\Delta)$. This fixed point assigns to each pair of states their probabilistic bisimilarity distance.

Given that $\boldsymbol{\mu}(\Delta)$ captures the probabilistic bisimilarity distances, we define the following sets.

$$D_0 = \{(s,t) \in S^2 \mid \boldsymbol{\mu}(\Delta)(s,t) = 0\}$$
$$D_1 = \{(s,t) \in S^2 \mid \boldsymbol{\mu}(\Delta)(s,t) = 1\}$$

The probabilistic bisimilarity pseudometric $\boldsymbol{\mu}(\Delta)$ provides a quantitative generalization of probabilistic bisimilarity as captured by the following result by Desharnais et al. [11, Theorem 1].

Theorem 1. $D_0 = \{\, (s,t) \in S^2 \mid s \sim t \,\}$.

3 Distance One

We concluded the previous section with the characterization of D_0 as the set of state pairs that are probabilistic bisimilar. In this section we present a characterization of D_1 as a fixed point of the function introduced in Definition 5.

Let us consider the case that the probabilistic bisimilarity distance of states s and t is one, that is, $\boldsymbol{\mu}(\Delta)(s,t) = 1$. Then $\Delta(\boldsymbol{\mu}(\Delta))(s,t) = 1$. From the definition of Δ, we can conclude that either $\ell(s) \neq \ell(t)$, or for all couplings $\omega \in \Omega(\tau(s), \tau(t))$ we have $\mathrm{support}(\omega) \subseteq D_1$.

We partition the set S^2 of state pairs into

$$S^2_0 = \{(s,t) \in S^2 \mid s \sim t\}$$
$$S^2_1 = \{(s,t) \in S^2 \mid \ell(s) \neq \ell(t)\}$$
$$S^2_? = S^2 \setminus (S^2_0 \cup S^2_1)$$

Hence, if $\boldsymbol{\mu}(\Delta)(s,t) = 1$, then either $(s,t) \in S^2_1$, or $(s,t) \in S^2_?$ and for all couplings $\omega \in \Omega(\tau(s), \tau(t))$ we have $\mathrm{support}(\omega) \subseteq D_1$. This leads us to the following function.

Definition 5. *The function* $\Gamma : 2^{S^2} \to 2^{S^2}$ *is defined by*

$$\Gamma(X) = S^2_1 \cup \{\, (s,t) \in S^2_? \mid \forall \omega \in \Omega(\tau(s), \tau(t)) : \mathrm{support}(\omega) \subseteq X \,\}.$$

Proposition 2. *The function* Γ *is monotone.*

Since the set 2^{S^2} of subsets of S^2 endowed with the order $\subseteq$ is a complete lattice (see [9, Example 2.6(2)]) and the function Γ is monotone, we can conclude from the Knaster-Tarski fixed point theorem that Γ has a greatest fixed point, which we denote by $\boldsymbol{\nu}(\Gamma)$. Next, we show that D_1 is a fixed point of Γ.

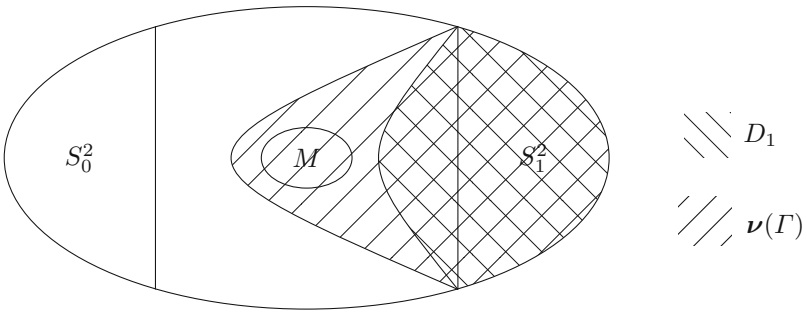
Proposition 3. $D_1 = \Gamma(D_1)$.

Since we have already seen that D_1 is a fixed point of Γ, we have that $D_1 \subseteq \boldsymbol{\nu}(\Gamma)$. To conclude that D_1 is the greatest fixed point of Γ, it remains to show that $\boldsymbol{\nu}(\Gamma) \subseteq D_1$, which is equivalent to the following.

Proposition 4. $\boldsymbol{\nu}(\Gamma) \setminus D_1 = \emptyset$.

Proof. Towards a contradiction, assume that $\boldsymbol{\nu}(\Gamma) \setminus D_1 \neq \emptyset$. Let

$$m = \min\{\boldsymbol{\mu}(\Delta)(s,t) \mid (s,t) \in \boldsymbol{\nu}(\Gamma) \setminus D_1\}$$
$$M = \{(s,t) \in \boldsymbol{\nu}(\Gamma) \setminus D_1 \mid \boldsymbol{\mu}(\Delta)(s,t) = m\}$$



Since $\boldsymbol{\nu}(\Gamma) \setminus D_1 \neq \emptyset$, we have that $M \neq \emptyset$. Furthermore,

$$M \subseteq \boldsymbol{\nu}(\Gamma) \setminus D_1. \tag{1}$$

Since $\boldsymbol{\nu}(\Gamma) \setminus D_1 \subseteq \boldsymbol{\nu}(\Gamma)$, we have

$$M \subseteq \boldsymbol{\nu}(\Gamma) = \Gamma(\boldsymbol{\nu}(\Gamma)) \subseteq S_1^2 \cup S_?^2. \tag{2}$$

For all $(s,t) \in M$,

$$\begin{aligned} &(s,t) \in \boldsymbol{\nu}(\Gamma) \wedge (s,t) \notin D_1 \quad [(1)] \\ &\Rightarrow (s,t) \in \Gamma(\boldsymbol{\nu}(\Gamma)) \wedge (s,t) \notin S_1^2 \\ &\Rightarrow \forall \omega \in \Omega(\tau(s), \tau(t)) : \mathrm{support}(\omega) \subseteq \boldsymbol{\nu}(\Gamma). \end{aligned} \tag{3}$$

For each $(s,t) \in M$, let

$$\omega_{s,t} = \operatorname*{argmin}_{\omega \in \Omega(\tau(s),\tau(t))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v). \tag{4}$$

We distinguish the following two cases.

– Assume that there exists $(s,t) \in M$ such that $\mathrm{support}(\omega_{s,t}) \cap D_1 \neq \emptyset$. Let

$$p = \sum_{(u,v)\in \boldsymbol{\nu}(\Gamma)\cap D_1} \omega_{s,t}(u,v).$$

By (3), we have that $\mathrm{support}(\omega_{s,t}) \subseteq \boldsymbol{\nu}(\Gamma)$. Since $\mathrm{support}(\omega_{s,t}) \cap D_1 \neq \emptyset$ by assumption, we can conclude that $p > 0$. Again using the fact that $\mathrm{support}(\omega_{s,t}) \subseteq \boldsymbol{\nu}(\Gamma)$, we have that

$$\sum_{(u,v)\in \boldsymbol{\nu}(\Gamma)\setminus D_1} \omega_{s,t}(u,v) = 1 - p. \tag{5}$$

Furthermore,

$$\begin{aligned}
m &= \boldsymbol{\mu}(\Delta)(s,t)\\
&= \Delta(\boldsymbol{\mu}(\Delta))(s,t)\\
&= \min_{\omega\in\Omega(\tau(s),\tau(t))} \sum_{u,v\in S} \omega(u,v)\,\boldsymbol{\mu}(\Delta)(u,v)\\
&= \sum_{u,v\in S} \omega_{s,t}(u,v)\,\boldsymbol{\mu}(\Delta)(u,v) \quad [(4)]\\
&= \sum_{(u,v)\in \boldsymbol{\nu}(\Gamma)} \omega_{s,t}(u,v)\,\boldsymbol{\mu}(\Delta)(u,v) \quad [(3)]\\
&= \sum_{(u,v)\in \boldsymbol{\nu}(\Gamma)\cap D_1} \omega_{s,t}(u,v)\,\boldsymbol{\mu}(\Delta)(u,v) + \sum_{(u,v)\in \boldsymbol{\nu}(\Gamma)\setminus D_1} \omega_{s,t}(u,v)\,\boldsymbol{\mu}(\Delta)(u,v)\\
&= p + \sum_{(u,v)\in \boldsymbol{\nu}(\Gamma)\setminus D_1} \omega_{s,t}(u,v)\,\boldsymbol{\mu}(\Delta)(u,v)\\
&\geq p + (1-p)m.
\end{aligned}$$

The last step follows from (5) and the fact that $\boldsymbol{\mu}(\Delta)(u,v) \geq m$ for all $(u,v) \in \boldsymbol{\nu}(\Gamma) \setminus D_1$. From the facts that $p > 0$ and $m \geq p + (1-p)m$ we can conclude that $m \geq 1$. This contradicts (1).

– Otherwise, $\mathrm{support}(\omega_{s,t}) \cap D_1 = \emptyset$ for all $(s,t) \in M$. Next, we will show that M is a probabilistic bisimulation under this assumption. From the fact that M is a probabilistic bisimulation, we can conclude from Theorem 1 that $\boldsymbol{\mu}(\Delta)(s,t) = 0$ for all $(s,t) \in M$. Hence, since $M \neq \emptyset$ we have that $M \cap S_0^2 \neq \emptyset$ which contradicts (2).
Next, we prove that M is a probabilistic bisimulation. Let $(s,t) \in M$. Since $M \subseteq \boldsymbol{\nu}(\Gamma) \setminus D_1$ by (1), we have that $(s,t) \notin D_1$ and, hence, $\Delta(\boldsymbol{\mu}(\Delta))(s,t) = \boldsymbol{\mu}(\Delta)(s,t) < 1$. From the definition of Δ, we can conclude that $\ell(s) = \ell(t)$. Since

$$\begin{aligned}
m &= \boldsymbol{\mu}(\Delta)(s,t)\\
&= \sum_{(u,v)\in \boldsymbol{\nu}(\Gamma)\setminus D_1} \omega_{s,t}(u,v)\,\boldsymbol{\mu}(\Delta)(u,v) \quad [\text{as above}]
\end{aligned}$$

and $\boldsymbol{\mu}(\Delta)(u,v) \geq m$ for all $(u,v) \in \boldsymbol{\nu}(\Gamma) \setminus D_1$, we can conclude that $\boldsymbol{\mu}(\Delta)(u,v) = m$ for all $(u,v) \in \text{support}(\omega_{s,t})$. Hence, $\text{support}(\omega_{s,t}) \subseteq M$. Therefore, M is a probabilistic bisimulation. □

Theorem 2. $D_1 = \boldsymbol{\nu}(\Gamma)$.

Proof. Immediate consequence of Proposition 3 and 4. □

We have shown that D_1 can be characterized as the greatest fixed point of Γ. Next, we will show that D_1 can be decided in polynomial time.

Theorem 3. *Distance one can be decided in* $O(n^2 + m^2)$.

Proof. As we will show in Theorem 5, distance smaller than one can be decided in $O(n^2 + m^2)$. Hence, distance one can be decided in $O(n^2 + m^2)$ as well. □

4 Distance Smaller Than One

To compute the set of state pairs which have distance one, we can first compute the set of state pairs which have distance less than one. The latter set we denote by $D_{<1}$. We can then obtain D_1 by taking the complement of $D_{<1}$. As we will discuss below, $D_{<1}$ can be characterized as the least fixed point of the following function.

Definition 6. *The function* $\Upsilon : 2^{S^2} \to 2^{S^2}$ *is defined by*

$$\Upsilon(X) = S^2 \setminus \Gamma(S^2 \setminus X).$$

The next theorem follows from Theorem 2.

Theorem 4. $D_{<1} = \boldsymbol{\mu}(\Upsilon)$.

Next, we show that the computation of $D_{<1}$ can be formulated as a reachability problem on a directed graph which is induced by the labelled Markov chain. Thus, we can use standard search algorithms, for example, breadth-first search, on the induced graph.

Next, we present the graph induced by the labelled Markov chain.

Definition 7. *The directed graph* $G = (V, E)$ *is defined by*

$$V = S_0^2 \cup S_?^2$$
$$E = \{\, \langle (u,v),(s,t)\rangle \mid \tau(s)(u) > 0 \wedge \tau(t)(v) > 0 \,\}$$

We are left to show that in the graph G defined above, a vertex (s,t) is reachable from some vertex in S_0^2 if and only if the state pair (s,t) in the labelled Markov chain has distance less than one.

As we have discussed earlier, if a state pair (s,t) has distance one, either s and t have different labels, or for all couplings $\omega \in \Omega(\tau(s),\tau(t))$ we have that $\text{support}(\omega) \subseteq D_1$. To avoid the universal quantification over couplings, we will use Proposition 1 in the proof of following proposition.

Proposition 5. $\boldsymbol{\mu}(\Upsilon) = \{\, (s,t) \mid (s,t) \text{ is reachable from some } (u,v) \in S_0^2 \,\}$.

Theorem 5. *Distance smaller than one can be decided in* $O(n^2 + m^2)$.

Proof. Distance smaller than one can be decided as follows.

1. Decide distance zero.
2. Breadth-first search of G, with the queue initially containing the pairs of states that have distance zero.

By Theorem 4 and Proposition 5, we have that s and t have distance smaller than one if and only if (s,t) is reachable in the directed graph G from some (u,v) such that u and v have distance zero. These reachable state pairs can be computed using breadth-first search, with the queue initially containing S_0^2.

Distance zero, that is, probabilistic bisimilarity, can be decided in $O(m \log n)$ as shown by Derisavi et al. in [10]. The directed graph G has n^2 vertices and m^2 edges. Hence, breadth-first search takes $O(n^2 + m^2)$. □

5 Number of Non-trivial Distances

As we have already discussed earlier, distance zero captures that states behave exactly the same, that is, they are probabilistic bisimilar, and distance one indicates that states behave very differently. The remaining distances, that is, those greater than zero and smaller than one, we call non-trivial. Being able to determine quickly the number of non-trivial distances of a labelled Markov chain allows us to decide whether computing all these non-trivial distances (using some policy iteration algorithm) is feasible.

To determine the number of non-trivial distances of a labelled Markov chain, we use the following algorithm.

1. Decide distance zero.
2. Decide distance one.

As first proved by Baier [4], distance zero, that is, probabilistic bisimilarity, can be decided in polynomial time. As we proved in Theorem 3, distance one can be decided in polynomial time as well. Hence, we can compute the number of non-trivial distances in polynomial time.

To decide distance zero, we implemented the algorithm to decide probabilistic bisimilarity due to Derisavi et al. [10] in Java. We also implemented our algorithm to decide distance one, described in the proof of Theorems 3 and 5.

We applied our implementation to labelled Markov chains that model randomized algorithms and probabilistic protocols. These labelled Markov chains have been obtained from the verification tool PRISM [20]. We compute the number of non-trivial distances for two models: the randomized self-stabilising algorithm due to Herman [14] and the bounded retransmission protocol by Helmink et al. [13].

For the randomized self-stabilising algorithm, the size of the labelled Markov chain grows exponentially in the numbers of processes, N. The results for the randomized self-stabilising algorithm are shown in the table below. As we can see from the table, for systems up to 128 states, the algorithm runs for less than a second. For the system with 512 states, the algorithm terminates within seven minutes. For the case $N = 3$, there are only 12 non-trivial distances. The size is so small that we can easily compute all the non-trivial distances. Section 6 will use the simple policy iteration algorithm as the next step to compute them. The same applies to the case $N = 5$. For $N = 7$ or 9, the number of non-trivial distances is around 11,000 and 200,000, respectively. This makes computing all of them infeasible. Thus, instead of computing all of them, we need to find alternative ways to handle systems with a large number of non-trivial distances. We will discuss two alternative ways in Sects. 7 and 8. Moreover, in this example, as $|D_1| = |S_1^2|$, we know that all the state pairs with distance one are those that have different labels.

N	$\lvert S\rvert$	$D_0 + D_1$	Non-trivial	$\lvert D_0\rvert$	$\lvert D_1\rvert$	$\lvert S_1^2\rvert$
3	8	1.00 ms	12	38	14	14
5	32	6.06 ms	280	304	440	440
7	128	0.77 s	11,032	2,160	3,192	3,192
9	512	378.42 s	230,712	13,648	17,784	17,784

In the bounded retransmission protocol, there are two parameters: N denotes the number of chunks and M the maximum allowed number of retransmissions of each chunk. The results are shown in the table below. The algorithm can handle systems up to 3,526 states within 11 min. In this example, there are no non-trivial distances. As a consequence, deciding distance zero and one suffices to compute all the distances in this case.

N	M	S	$D_0 + D_1$	$\lvert D_0\rvert$	$\lvert D_1\rvert$	$\lvert S_1^2\rvert$
16	2	677	3.0 s	456,977	1,352	1,352
16	3	886	8.6 s	783,226	1,770	1,770
16	4	1,095	17.5 s	1,196,837	2,188	2,188
16	5	1,304	22.8 s	1,697,810	2,606	2,606
32	2	1,349	24.7 s	1,817,105	2,696	2,696
32	3	1,766	69.7 s	3,115,226	3,530	3,530
32	4	2,183	141.0 s	4,761,125	4,364	4,364
32	5	2,600	208.6 s	6,754,802	5,198	5,198
64	2	2,693	235.2 s	7,246,865	5,384	5,384
64	3	3,526	616.4 s	12,425,626	7,050	7,050

6 All Distances

To compute all distances of a labelled Markov chain, we augment the existing state of the art algorithm, which is based on algorithms due to Derisavi et al. [10] (step 1) and Bacci et al. [2] (step 3), by incorporating our decision procedure (step 2) as follows.

1. Decide distance zero.
2. Decide distance one.
3. Simple policy iteration.

Given that we not only decide distance zero, but also distance one, before running simple policy iteration, the correctness of the simple policy iteration algorithm in the augmented setting needs an adjusted proof.

As we already discussed in the previous section, step 1 and 2 are polynomial time. However, step 3 may take at least exponential time in the worst case, as we have shown in [27]. Hence, the overall algorithm is exponential time.

The first example we consider here is the synchronous leader election protocol of Itai and Rodeh [15] which is taken from PRISM. The protocol takes the number of processors, N, and a constant K as parameters. We compare the running time of our new algorithm with the state of the art algorithm, that combines algorithms due to Derisavi et al. and due to Bacci et al. The results are shown in the table below. In this protocol, the number of non-trivial distances is zero. Thus, our new algorithm terminates without running step 3 which is the simple policy iteration algorithm. On the other hand, the original simple policy iteration algorithm computes the distances of all the elements in the set $D_1 \setminus S_1^2$, the size of which is huge as can be seen from the last two columns of the table.

N	K	$\lvert S\rvert$	$D_0 + \text{SPI}$	$D_0 + D_1 + \text{SPI}$	Speed-up	$\lvert D_0\rvert$	$\lvert D_1\rvert$	$\lvert S_1^2\rvert$
3	2	26	4 s	1 ms	4,281	122	554	50
3	4	147	49 h	13 ms	13,800,000	7,419	14,190	292
3	6	459	-	214 ms	-	88,671	122,010	916
3	8	1,059	-	3 s	-	508,851	612,630	2,116
4	2	61	812 s	3 ms	305,000	459	3, 262	120
4	4	812	-	388 ms	-	145,780	513,564	1,622
4	6	3,962	-	82 s	-	4,350,292	11,347,152	7,922
4	8	12,400	-	2,971 s	-	46, 198,188	107,561,812	24,798
5	2	141	-	6 ms	-	2,399	17,482	280
5	4	4,244	-	33 s	-	3,318,662	14,692,874	8,486
6	2	335	-	25 ms	-	14,327	97,898	668

The simple policy iteration algorithm can only handle a limited number of states. For the labelled Markov chain with 26 states ($N = 3$ and $K = 2$) the simple policy iteration algorithm takes four seconds, while our new algorithm

takes one millisecond. The speed-up is more than 4,000 times. For the labelled Markov chain with 61 states ($N = 4$ and $K = 2$), the simple policy iteration algorithm runs in 812 s, while our new algorithm takes three milliseconds. The speed-up of the new algorithm is 30,000 times. The biggest system the simple policy iteration algorithm can handle is the one with 147 states ($N = 3$ and $K = 4$) and it takes more than 49 h. In contrast, our new algorithm terminates within 13 ms. That makes the new algorithm seven orders of magnitude faster than the state of the art algorithm. This example also shows that the new algorithm can handle systems with at least 12,400 states.

In the second example, we model two dies, one using a fair coin and the other one using a biased coin. The goal is to compute the probabilistic bisimilarity distance between these two dies. An implementation of the die algorithm is part of PRISM. The resulting labelled Markov chain has 20 states.

As there are only 30 non-trivial distances, we run the simple policy iteration algorithm as step 3. The new algorithm is about 46 times faster than the original algorithm.

$\|S\|$	D_0+SPI	$D_0 + D_1 +$ SPI	Speed-up	Non-trivial	$\|D_0\|$	$\|D_1\|$	$\|S_1^2\|$
20	5.55 s	0.12 s	46.25	30	20	350	198

7 Small Distances

As we have discussed in Sect. 5, for systems of which the number of non-trivial distances is so large that computing all of them is infeasible, we have to find alternative ways. In practice, as we only identify the state pairs with small distances, we can cut down the number of non-trivial distances by only computing those with small distances.

To compute the non-trivial distances smaller than a positive number, ε, we use the following algorithm.

1. Decide distance zero.
2. Decide distance one.
3. Compute the query set

$$Q = \{\, (s,t) \in S^2 \setminus (D_0 \cup D_1) \mid \Delta(d)(s,t) \leq \varepsilon \,\}$$

where

$$d(s,t) = \begin{cases} 1 \text{ if } (s,t) \in D_1 \\ 0 \text{ otherwise} \end{cases}$$

4. Simple partial policy iteration for Q.

The first two steps remain the same. In step 3, we compute a query set Q that contains all state pairs with distances no greater than ε, as shown in Proposition 6. In step 4, we use this set as the query set to run the simple partial policy iteration algorithm by Bacci et al. [2].

Proposition 6. *Let d be the distance function defined in step 3. For all $(s,t) \in S^2 \setminus (D_0 \cup D_1)$, if $\boldsymbol{\mu}(\Delta)(s,t) \leq \varepsilon$, then $\Delta(d)(s,t) \leq \varepsilon$.*

Given that we not only decide distance zero, but also distance one, before running simple partial policy iteration, the correctness of the simple partial policy iteration algorithm in the augmented setting needs an adjusted proof.

As we have seen before, step 1 and 2 take polynomial time. In step 3, computing $\Delta(d)$ corresponds to solving a minimum cost network flow problem. Such a problem can be solved in polynomial time using, for example, Orlin's network simplex algorithm [24]. As we have shown in [28], step 4 takes at least exponential time in the worst case. Therefore, the overall algorithm is exponential time.

We consider the randomized quicksort algorithm, an implementation of which is part of jpf-probabilistic [31]. The input of the algorithm is the list to be sorted. The list of size 6 gives rise to a labelled Markov chain with 82 states. We compare the running time of the new algorithm for small distances ($D_0 + D_1 + Q$ + SPPI) to the original algorithm (D_0 + SPI) and the new algorithm presented in Sect. 6 ($D_0 + D_1$ + SPI). The original algorithm (D_0 + SPI) takes about 14 h, the new algorithm which incorporates the decision procedure of distance one takes less than 7 h. For $\varepsilon = 0.1$, the new algorithm for small distances takes 57 min. This makes it about 7 times faster than the algorithm presented in Sect. 6 and about 15 times faster than the original simple policy iteration algorithm. For $\varepsilon = 0.01$, the new algorithm for small distances takes even less time, namely 41 min. As can be seen in the table below, the total number of non-trivial distances is 2,300. The simple partial policy iteration algorithm starts with the query set Q but may have to compute the distances of other state pairs as well. The total number of state pairs considered by the simple partial policy iteration algorithm can be found in the column labelled Total.

ε	$D_0 + D_1 + Q$ + SPPI	$\lvert Q \rvert$	Total	Non-trivial
0.1	57 min	96	1,002	2,300
0.01	41 min	84	842	2,300

8 Approximation Algorithm

We propose another solution to deal with a large number of non-trivial distances by approximating the distances rather than computing the exact values. To approximate the distances such that the approximate values differ from the exact ones by at most α, a positive number, we use the following algorithm.

1. Decide distance zero.
2. Decide distance one.

3. $l(s,t) = \begin{cases} 1 \text{ if } (s,t) \in D_1 \\ 0 \text{ otherwise} \end{cases}$
$u(s,t) = \begin{cases} 0 \text{ if } (s,t) \in D_0 \\ 1 \text{ otherwise} \end{cases}$
repeat
for each $(s,t) \in S^2 \setminus (D_0 \cup D_1)$
if $l(s,t) \neq u(s,t)$
$l(s,t) = \Delta(l)(s,t)$
$u(s,t) = \Delta(u)(s,t)$
until $\|l - u\| \leq \alpha$

Again, the first two steps remain the same. Step 3 contains the new approximation algorithm called *distance iteration* (DI). In this step, we define two distance functions, a lower-bound l and an upper-bound u. We repeatedly apply Δ to these two functions until the difference of the non-trivial distances in these two functions is smaller than the threshold α. For each state pair we end up with an interval of at most size α in which their distance lies. To prove the algorithm correct, we modify the function Δ defining the probabilistic bisimilarity distances slightly as follows.

Definition 8. *The function* $\Delta_0 : [0,1]^{S^2} \to [0,1]^{S^2}$ *is defined by*

$$\Delta_0(d)(s,t) = \begin{cases} 0 & \text{if } (s,t) \in D_0 \\ \Delta(d)(s,t) & \text{otherwise} \end{cases}$$

Some properties of Δ_0, which are key to the correctness proof of the above algorithm, are collected in the following theorem.

Theorem 6.

(a) The function Δ_0 *is monotone.*
(b) The function Δ_0 *is nonexpansive.*
(c) $\boldsymbol{\mu}(\Delta_0) = \boldsymbol{\mu}(\Delta)$.
(d) $\boldsymbol{\mu}(\Delta_0) = \boldsymbol{\nu}(\Delta_0)$.
(e) $\boldsymbol{\mu}(\Delta_0) = \sup_{m \in \mathbb{N}} \Delta_0^m(d_0)$, *where* $d_0(s,t) = 0$ *for all* $s,t \in S$.
(f) $\boldsymbol{\nu}(\Delta_0) = \inf_{n \in \mathbb{N}} \Delta_0^n(d_1)$, *where* $d_1(s,t) = 1$ *for all* $s,t \in S$.

Let us use randomized quicksort introduced in Sect. 7 and the randomized self-stabilising algorithm due to Herman [14] introduced in Sect. 5 as examples. Recall that for the randomized self-stabilising algorithm, when $N = 7$, the number of non-trivial distances is 11,032, which we are not able to handle using the simple policy iteration algorithm. We apply the approximation algorithm to this model and the randomized quicksort example with 82 states and present the results below. The accuracy α is set to be 0.01.

The approximation algorithm for randomized quicksort runs for about 14 min, which is about 3 to 4 times faster than the algorithm for small distances in Sect. 7. For the randomized self-stabilising algorithm with 128 states, the approximation algorithm terminates in about 54 h. Although the number of non-trivial

distances for the randomized self-stabilising algorithm is about 5 times of that of the randomized quicksort, the running time is more than 200 times slower. It is unknown whether this approximation algorithm has exponential running time.

Model	$\|S\|$	Non-trivial	$D_0 + D_1$+DI
Randomized quicksort	82	2,300	14 min
Randomized self-stabilising algorithm	128	11,032	54 h

9 Conclusion

In this paper, we have presented a decision procedure for probabilistic bisimilarity distance one. This decision procedure provides the basis for three new algorithms to compute and approximate the probabilistic bisimilarity distances of a labelled Markov chain. The first algorithm decides distance zero, then decides distance one, and finally uses simple policy iteration to compute the remaining distances. As shown experimentally, this new algorithm significantly improves the state of the art algorithm that only decides distance zero and then uses simple policy iteration. The second algorithm computes all probabilistic bisimilarity distances that are smaller than some given upper bound, by deciding distance zero, deciding distance one, computing a query set, and running simple partial policy iteration for that query set. This second algorithm can handle labelled Markov chains that have considerably more non-trivial distances than our first algorithm. The third algorithm approximates the probabilistic bisimilarity distances up to a given accuracy, deciding distance zero, deciding distance one and running distance iteration. Also this third algorithm can handle labelled Markov chains that have considerably more non-trivial distances than our first algorithm. Whereas we know that the first two algorithms take at least exponential time in the worst case, the analysis of the running time of the third algorithm has not yet been determined. Moreover, if we are only interested in the probabilistic bisimilarity distances for a few state pairs, with pre-computation of distance zero and one we can exclude the state pairs with trivial distances. We can add the remaining state pairs to a query set and run simple partial policy iteration to get the distances. Alternatively, we can modify the distance iteration algorithm to approximate the distances for the predefined state pairs. The details of these new algorithms will be studied in the future.

Acknowledgements. The authors would like to thank Daniela Petrisan, Eric Ruppert and Dana Scott for discussions related to this research. The authors are also grateful to the referees for their constructive feedback.

References

1. Aceto, L., Ingolfsdottir, A., Larsen, K., Srba, J.: Reactive Systems: Modelling, Specification and Verification. Cambridge University Press, Cambridge (2003)
2. Bacci, G., Bacci, G., Larsen, K.G., Mardare, R.: On-the-fly exact computation of bisimilarity distances. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013. LNCS, vol. 7795, pp. 1–15. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36742-7_1
3. Bacci, G., Bacci, G., Larsen, K.G., Mardare, R.: On the metric-based approximate minimization of Markov chains. In: Chatzigiannakis, I., Indyk, P., Kuhn, F., Muscholl, A. (eds.) Proceedings of the 44th International Colloquium on Automata, Languages, and Programming, Warsaw, Poland, July 2017. Leibniz International Proceedings in Informatics, vol. 80, pp. 104:1–104:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
4. Baier, C.: Polynomial time algorithms for testing probabilistic bisimulation and simulation. In: Alur, R., Henzinger, T.A. (eds.) CAV 1996. LNCS, vol. 1102, pp. 50–61. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61474-5_57
5. Bellman, R.: A Markovian decision process. J. Math. Mech. **6**(5), 679–684 (1957)
6. van Breugel, F.: On behavioural pseudometrics and closure ordinals. Inf. Process. Lett. **112**(18), 715–718 (2012)
7. van Breugel, F.: Probabilistic bisimilarity distances. ACM SIGLOG News **4**(4), 33–51 (2017)
8. Chen, D., van Breugel, F., Worrell, J.: On the complexity of computing probabilistic bisimilarity. In: Birkedal, L. (ed.) FoSSaCS 2012. LNCS, vol. 7213, pp. 437–451. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28729-9_29
9. Davey, B., Priestley, H.: Introduction to Lattices and Order. Cambridge University Press, Cambridge (2002)
10. Derisavi, S., Hermanns, H., Sanders, W.: Optimal state-space lumping in Markov chains. In. Process. Lett. **87**(6), 309–315 (2003)
11. Desharnais, J., Gupta, V., Jagadeesan, R., Panangaden, P.: Metrics for labeled Markov systems. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR 1999. LNCS, vol. 1664, pp. 258–273. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48320-9_19
12. Giacalone, A., Jou, C.-C., Smolka, S.: Algebraic reasoning for probabilistic concurrent systems. In: Proceedings of the IFIP WG 2.2/2.3 Working Conference on Programming Concepts and Methods, Sea of Gallilee, Israel, April 1990, pp. 443–458. North-Holland (1990)
13. Helmink, L., Sellink, M.P.A., Vaandrager, F.W.: Proof-checking a data link protocol. In: Barendregt, H., Nipkow, T. (eds.) TYPES 1993. LNCS, vol. 806, pp. 127–165. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58085-9_75
14. Herman, T.: Probabilistic self-stabilization. Inf. Process. Lett. **35**(2), 63–67 (1990)
15. Itai, A., Rodeh, M.: Symmetry breaking in distributed networks. Inf. Comput. **88**(1), 60–87 (1990)
16. Katoen, J.-P., Kemna, T., Zapreev, I., Jansen, D.N.: Bisimulation minimisation mostly speeds up probabilistic model checking. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 87–101. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71209-1_9
17. Khachiyan, L.: A polynomial algorithm in linear programming. Sov. Math. Dokl. **20**(1), 191–194 (1979)

18. Klee, V., Witzgall, C.: Facets and vertices of transportation polytopes. In: Dantzig, G., Veinott, A. (eds.) Proceedings of 5th Summer Seminar on the Mathematis of the Decision Sciences, Stanford, CA, USA, July/August 1967. Lectures in Applied Mathematics, vol. 11, pp. 257–282. AMS (1967)
19. Knuth, D., Yao, A.: The complexity of nonuniform random number generation. In: Traub, J. (ed.) Proceedings of a Symposium on New Directions and Recent Results in Algorithms and Complexity, Pittsburgh, PA, USA, April 1976, pp. 375–428. Academic Press (1976)
20. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
21. Larsen, K., Skou, A.: Bisimulation through probabilistic testing. In: Proceedings of the 16th Annual ACM Symposium on Principles of Programming Languages, Austin, TX, USA, January 1989, pp. 344–352. ACM (1989)
22. Milner, R. (ed.): A Calculus of Communicating Systems. LNCS, vol. 92. Springer, Heidelberg (1980). https://doi.org/10.1007/3-540-10235-3
23. Murthy, A., et al.: Approximate bisimulations for sodium channel dynamics. In: Gilbert, D., Heiner, M. (eds.) CMSB 2012. LNCS, pp. 267–287. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33636-2_16
24. Orlin, J.: A polynomial time primal network simplex algorithm for minimum cost flows. Math. Program. **78**(2), 109–129 (1997)
25. Park, D.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) GI-TCS 1981. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981). https://doi.org/10.1007/BFb0017309
26. Sen, P., Deshpande, A., Getoor, L.: Bisimulation-based approximate lifted inference. In: Bilmes, J., Ng, A. (eds.) Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, pp. 496–505. AUAI Press (2009)
27. Tang, Q., van Breugel, F.: Computing probabilistic bisimilarity distances via policy iteration. In: Desharnais, J., Jagadeesan, R. (eds.) Proceedings of the 27th International Conference on Concurrency Theory, Quebec City, QC, Canada, August 2016. Leibniz International Proceedings in Informatics, vol. 59, pp. 22:1–22:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2016)
28. Tang, Q., van Breugel, F.: Algorithms to compute probabilistic bisimilarity distances for labelled Markov chains. In: Meyer, R., Nestmann, U. (eds.) Proceedings of the 28th International Conference on Concurrency Theory, Berlin, Germany, September 2017. Leibniz International Proceedings in Informatics, vol. 85, pp. 27:1–27:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
29. Tarski, A.: A lattice-theoretic fixed point theorem and its applications. Pac. J. Math. **5**(2), 285–309 (1955)
30. Valmari, A., Franceschinis, G.: Simple $O(m \log n)$ time Markov chain lumping. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 38–52. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12002-2_4
31. Zhang, X., van Breugel, F.: Model checking randomized algorithms with Java PathFinder. In: Proceedings of the 7th International Conference on the Quantitative Evaluation of Systems, Williamsburg, VA, USA, September 2010, pp. 157–158. IEEE (2010)