Algorithms to Compute Probabilistic Bisimilarity Distances for Labelled Markov Chains*

Qiyi Tang[1] and Franck van Breugel[2]

1 **DisCoVeri Group, Department of Electrical Engineering and Computer Science, York University, Toronto**
2 **DisCoVeri Group, Department of Electrical Engineering and Computer Science, York University, Toronto**

Abstract

In the late nineties, Desharnais, Gupta, Jagadeesan and Panangaden presented probabilistic bisimilarity distances on the states of a labelled Markov chain. This provided a quantitative generalisation of probabilistic bisimilarity introduced by Larsen and Skou a decade earlier. In the last decade, several algorithms to approximate and compute these probabilistic bisimilarity distances have been put forward. In this paper, we correct, improve and generalise some of these algorithms. Furthermore, we compare their performance experimentally.

1998 ACM Subject Classification D.2.4 Software/Program Verification, F.1.1 Models of Computation, G.3 Probability and Statistics

Keywords and phrases labelled Markov chain, probabilistic bisimilarity, pseudometric, policy iteration

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2017.27

1 Introduction

For the last three and a half decades behavioural equivalences such as bisimilarity, due to Milner [22] and Park [25], have been a cornerstone of concurrency theory. As this theory matured, more detailed models of concurrent systems have been developed, such as models with probabilities. For these enriched models, behavioural equivalences were proposed, such as probabilistic bisimilarity due to Larsen and Skou [21].

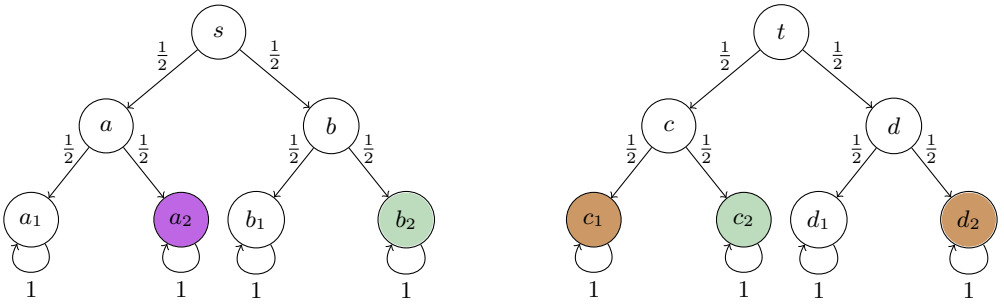
Although these behavioural equivalences allow us to reason about the behaviour of these models, they have one drawback: they are not robust. That is, small changes in the probabilities may cause equivalent states to becomes inequivalent or vice versa. Since the probabilities are usually obtained experimentally and, therefore are often just an approximation, this lack of robustness is a serious limitation of behavioural equivalences for these models with probabilities. This lack of robustness was first pointed out by Giacalone, Jou and Smolka [13]. They suggested generalising behavioural equivalences, which assign to each pair of states a Boolean, to behavioural pseudometrics, which assign to each pair of states a nonnegative real number.

The distance of a pair of states provides a quantitative measure of their behavioural similarity. The smaller this distance, the more alike the states behave. Distance zero captures that the states behave exactly the same, that is, they are behaviourally equivalent.

* Part of this work was done while the second author was visiting the Simons Institute for the Theory of Computing. This research was also supported by the Natural Sciences and Engineering Research Council of Canada.

For a more detailed discussion of the merits of behavioural pseudometrics, we refer the reader to, for example, [24, Chapter 8].

Labelled Markov chains are often used to model systems with probabilistic behaviour. An example of such a Markov chain is depicted below. In a labelled Markov chain, each state has a label. In the example below, the label is represented by the colour of the state. These labels are used to capture that particular properties of interest hold in some states and do not hold in other states.



Several behavioural pseudometrics for labelled Markov chains have been proposed, the probabilistic bisimilarity pseudometric due to Desharnais, Gupta, Jagadeesan and Panangaden [12] being the most notable one. In this paper, we focus on this probabilistic bisimilarity pseudometric. Some of the probabilistic bisimilarity distances for the above labelled Markov chain can be found in the table below. For some historical background on this behavioural pseudometric we refer the reader to [6, Section 1].

	s	t	a	b	c	d
s	0	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{3}{4}$	1	$\frac{3}{4}$
t	$\frac{1}{2}$	0	$\frac{3}{4}$	$\frac{3}{4}$	1	$\frac{3}{4}$
a	$\frac{3}{4}$	$\frac{3}{4}$	0	$\frac{1}{2}$	1	$\frac{1}{2}$
b	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$
c	1	1	1	$\frac{1}{2}$	0	$\frac{1}{2}$
d	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0

In order to exploit behavioural pseudometrics such as the probabilistic bisimilarity pseudometric, it is essential to be able to approximate or compute these behavioural distances. The first algorithm to approximate these distances was presented by Van Breugel, Sharma and Worrell in [5]. In their algorithm, the distance between states s and t, denoted $\delta(s,t)$, is computed as follows. Since $\delta(s,t) < q$, for some rational q, can be expressed in the existential fragment of the first order theory over the reals as shown by Van Breugel et al., and this theory is decidable as shown by Tarski [29], one can use binary search to approximate $\delta(s,t)$. The satisfiability problem for the existential fragment of the first order theory over the reals can be solved in polynomial space [7].

Subsequently, Chen, Van Breugel and Worrell [8] presented a polynomial time algorithm to compute the distances. They showed that the distances are rational and that those distances can be computed by means of Khachiyan's ellipsoid method [17]. In particular, they showed that the distance function can be expressed as the solution of a linear program. In this case, the separation algorithm, which is an integral part of the ellipsoid method, boils down to solving a minimum cost flow problem. The network simplex algorithm solves the latter problem in polynomial time [23].

It is well known that the ellipsoid method is simpler when the feasible region is bounded and full-dimensional (see, for example, [26, Chapter 13]). In Section 4 we show that the feasible region of the linear program used by Chen et al. to compute the probabilistic bisimilarity distances is bounded and full-dimensional. As a consequence, we can use the simpler version of the ellipsoid method to compute the probabilistic bisimilarity distances.

Bacci, Bacci, Larsen and Mardare [1] put forward yet another algorithm to compute the probabilistic bisimilarity distances. Their algorithm can be viewed as a basic algorithm, enhanced with an optimization. The key idea behind this optimization is to compute the distances on-the-fly. We will come back to this optimization later, but we will first focus on the basic algorithm.

In [28], we show that a slight modification of the basic algorithm of Bacci et al. can be seen as an incarnation of simple policy iteration, also known as sequential policy iteration. In particular, we construct a transformation mapping each labelled Markov chain to a simple stochastic game, a simplification of Shapley's stochastic games [27] due to Condon [9]. The vertices of the simple stochastic game represent pairs of states of the labelled Markov chain. Furthermore, the probabilistic bisimilarity distance of two states of the labelled Markov chain is shown to be equal to the value of the corresponding vertex of the simple stochastic game.

A variety of algorithms has been developed to compute the value function of a simple stochastic game. Several of these algorithms use policy iteration. As long as there exists a choice in the strategy, also known as a policy, of a player that is not locally optimal, switch that choice for one that is locally optimal. Hoffman and Karp [15] introduced a policy iteration algorithm for stochastic games in which all non-optimal choices are switched in each iteration. Condon [10] presented a similar algorithm, known as simple policy iteration, that switches only one non-optimal choice per iteration.

In Section 5 we present our simple policy iteration algorithm. Furthermore, we modify this algorithm to mimic general policy iteration a la Karp and Hoffman. We prove both algorithms correct. In Section 6 we consider the on-the-fly optimization due to Bacci et al. [1]. This optimization boils down to considering partial policies. That is, a player only makes choices for an appropriate subset of the vertices. We call it partial policy iteration. As we will show, Bacci et al. do not always consider partial policies that are defined for sufficiently many vertices. We propose a modification of their algorithm and prove it correct. A partial variant that mimics general policy iteration can be developed and proved correct similarly.

In [28], we show that the number of iterations of the simple policy iteration algorithm is at least exponential in the number of states of the labelled Markov chain. In Section 7 we prove that this is also the case for the simple partial policy iteration algorithm, even if we are only interested in the distance of a single pair of states.

The algorithms considered in this paper are the one which applies the first order theory over the reals, the ellipsoid method, simple policy iteration, general policy iteration, simple partial policy iteration and general partial policy iteration. To compare the performance of these six algorithms to approximate and compute probabilistic bisimilarity distances for labelled Markov chain, we ran several experiments. All six algorithms were implemented in Java. These implementations were run on a number of labelled Markov chains, which model well-known randomized algorithms and were obtained from examples of probabilistic model checkers such as PRISM [20] and jpf-probabilistic [31]. These experiments and the results are discussed in Section 8.

The contributions of this paper are the following. In Section 4 we show that the feasible region of the linear programming of Chen et al. describing the probabilistic bisimilarity dis-

tances is bounded and full-dimensional. As a consequence, we can resort to a simpler version of the ellipsoid method to compute the distances. We study the on-the-fly optimization of the algorithm of Bacci et al. in Section 6 and show that it does not always consider sufficiently many states. We modify their optimization and prove our modification correct. We consider this our main contribution. In Section 7 we prove an exponential lower bound for the simple partial policy iteration algorithm. Finally, by means of experiments we compare the performance of the different algorithms to approximate and compute the probabilistic bisimilarity distances in Section 8.

2 Labelled Markov Chains and Probabilistic Bisimilarity

We start by reviewing the model of interest, labelled Markov chains, and its most well known behavioural equivalence, probabilistic bisimilarity due to Larsen and Skou [21]. We denote the set of probability distributions on a finite set S by $\mathit{Distr}(S)$.

▶ **Definition 1.** A labelled Markov chain is a tuple $\langle S, L, \tau, \ell \rangle$ consisting of
- a finite set S of states,
- a finite set L of labels,
- a transition function $\tau : S \to \mathit{Distr}(S)$, and
- a labelling function $\ell : S \to L$.

We restrict our attention to labelled Markov chains with rational transition probabilities. For the remainder of this paper, we fix such a labelled Markov chain. In order to characterize probabilistic bisimilarity, we first introduce the notion of a coupling of probability distributions.

▶ **Definition 2.** Let $\mu, \nu \in \mathit{Distr}(S)$. The set $\Omega(\mu, \nu)$ of couplings[1] of μ and ν is defined by

$$\Omega(\mu,\nu) = \left\{ \omega \in \mathit{Distr}(S \times S) \;\middle|\; \sum_{t \in S} \omega(s,t) = \mu(s) \wedge \sum_{s \in S} \omega(s,t) = \nu(t) \right\}.$$

The set $\Omega(\mu, \nu)$ is a convex polytope. We denote its vertices by $V(\Omega(\mu, \nu))$. Generally, the set $\Omega(\mu, \nu)$ is infinite, but the set $V(\Omega(\mu, \nu))$ is finite (see, for example, [18, page 259]). The following characterization of probabilistic bisimilarity, in terms of couplings, is due to Jonsson and Larsen [16, Theorem 4.6]. For a discussion of the notion of a coupling we refer the reader to, for example, [4].

▶ **Definition 3.** An equivalence relation $R \subseteq S \times S$ 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 $\mathrm{support}(\omega) \subseteq R$, where $\mathrm{support}(\omega) = \{ (u, v) \in S \times S \mid \omega(u, v) > 0 \}$. Probabilistic bisimilarity, denoted $\sim$, is the largest probabilistic bisimulation.

3 Probabilistic Bisimilarity Distances

The probabilistic bisimilarity pseudometric of Desharnais et al. [12] maps each pair of states of a labelled Markov chain to a distance, which an element of the unit interval $[0, 1]$. Hence, the pseudometric is a function from $S \times S$ to $[0, 1]$, that is, an element of $[0, 1]^{S \times S}$. Such a function is a pseudometric if it satisfies the following three properties: for all $s, t, u \in S$,

[1] In the literature, different terminology is used. For example, in [1] these are called matchings.

$d(s,s) = 0$, $d(s,t) = d(t,s)$ and $d(s,u) \leq d(s,t) + d(t,u)$. As we will discuss below, the probabilistic bisimilarity pseudometric can be defined as a fixed point of the following function.

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

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

To define the probabilistic bisimilarity pseudometric as a fixed point of Δ we allude to the Knaster-Tarski fixed point theorem [30].

▶ **Theorem 5.** *Let X be a complete lattice and $f : X \to X$ a monotone function.*
(a) *The least fixed point of f is $\bigsqcap\{\, x \in X \mid f(x) \sqsubseteq x \,\}$.*
(b) *The greatest fixed point of f is $\bigsqcup\{\, x \in X \mid x \sqsubseteq f(x) \,\}$.*

For background material on order theory we refer the reader to, for example, [11]. To apply the above theorem, we need to define an order on $[0,1]^{S\times S}$. For d, $e \in [0,1]^{S\times S}$ we write $d \sqsubseteq e$ if $d(s,t) \leq e(s,t)$ for all s, $t \in S$. The set $[0,1]^{S\times S}$ endowed with the order $\sqsubseteq$ forms a complete lattice. Since Δ is a monotone function, we can conclude from the above Knaster-Tarski fixed point theorem that Δ has a least fixed point. We denote this fixed point by δ, which is a pseudometric. This is the probabilistic bisimilarity pseudometric introduced by Desharnais et al.

The probabilistic bisimilarity pseudometric δ provides a quantitative generalisation of probabilistic bisimilarity as captured by the following result which can be found in [12, Theorem 1].

▶ **Theorem 6.** *For all s, $t \in S$, $s \sim t$ if and only if $\delta(s,t) = 0$.*

We conclude this section with an alternative characterization of the probabilistic bisimilarity pseudometric δ which is a small variation on the characterization that can be found in [8, Theorem 8]. It provides the basis for most of the algorithms we will discuss later in this paper.

Theorem 6 tells us that δ assigns to each state pair in $S^2_0 = \{\, (s,t) \in S \times S \mid s \sim t \,\}$ the distance zero. From the definition of Δ we can conclude that δ assigns to each state pair in $S^2_1 = \{\, (s,t) \in S \times S \mid \ell(s) \neq \ell(t) \,\}$ the distance one. Hence, it remains to compute the probabilistic bisimilarity distance for $S^2_? = \{\, (s,t) \in S \times S \mid \ell(s) = \ell(t) \wedge s \not\sim t \,\}$. Note that S^2_0, S^2_1 and $S^2_?$ form a partition of $S \times S$. As we mentioned in the introduction, our algorithms use policy iteration to compute these remaining distances.

▶ **Definition 7.** For a labelled Markov chain $\langle S, L, \tau, \ell \rangle$, the set $\mathcal{T}$ of total policies[2] is defined by

$$\mathcal{T} = \{\, T \in S^2_? \to \mathit{Distr}(S \times S) \mid \forall (s,t) \in S^2_? : T(s,t) \in V(\Omega(\tau(s), \tau(t))) \,\}.$$

Restricting to vertices in the above definition amounts to no loss of generality, since Theorem 9 holds also for total policies that map to arbitrary elements of the convex polytope $\Omega(\tau(s), \tau(t))$, rather than just its vertices (see [1, Remark 13]). For each $T \in \mathcal{T}$, the pair $\langle S \times S, T \rangle$ can be viewed as a Markov chain, by extending T such that

$$T(s,t)(u,v) = \begin{cases} 1 & \text{if } (u,v) = (s,t) \\ 0 & \text{otherwise} \end{cases}$$

for all $(s,t) \in (S \times S) \setminus S^2_?$.

[2] In the literature, different terminology is used. For example, in [1] these are called couplings.

▶ **Definition 8.** Let $T \in \mathcal{T}$. The function $\Gamma^T : [0,1]^{S\times S} \to [0,1]^{S\times S}$ is defined by

$$\Gamma^T(d)(s,t) = \begin{cases} 0 & \text{if } s \sim t \\ 1 & \text{if } \ell(s) \neq \ell(t) \\ \sum_{u,v\in S} T(s,t)(u,v)\, d(u,v) & \text{otherwise} \end{cases}$$

Since $[0,1]^{S\times S}$ is a complete lattice and Γ^T is a monotone function [1], we can conclude from the Knaster-Tarski fixed point theorem that Γ^T has a least fixed point. We denote this fixed point by γ^T. Note that $\gamma^T(s,t)$ captures the probability of reaching any (u,v) with $\ell(u) \neq \ell(v)$ from (s,t) in the Markov chain $\langle S \times S, T\rangle$ (see, for example, [3, Section 10.1.1] for a discussion of reachability probabilities). The probabilistic bisimilarity pseudometric δ can be characterized as follows.

▶ **Theorem 9.** $\delta = \min_{T\in\mathcal{T}} \gamma^T$.

4 The Ellipsoid Method

As shown by Chen, Van Breugel and Worrell [8], the probabilistic bisimilarity distances can be computed in polynomial time. In particular, they show that the distances can be obtained by solving a linear programming problem by means of Khachiyan's ellipsoid method [17]. It is well known that the method is simpler when the feasible region is bounded and full-dimensional (see, for example, [26, Chapter 13]). Below, we will show that the feasible region is bounded and full-dimensional in the setting of Chen et al. This feasible region is defined by

$$d(s,t) \geq 0 \tag{1}$$
$$d(s,t) \leq 1 \tag{2}$$
$$d(s,t) \leq \sum_{(u,v)\in S_?^2} d(u,v)\,\pi(u,v) + \sum_{(u,v)\in S_1^2} \pi(u,v) \tag{3}$$

for all $(s,t) \in S_?^2$ and $\pi \in V(\Omega(\tau(s),\tau(t)))$. Note that we need to decide probabilistic bisimilarity in order to define $S_?^2$ and, hence, the feasible region.

We restrict our attention to the case that the set $S_?^2$ is nonempty. Otherwise, there are no distances to compute. Obviously, the feasible region is bounded by $[0,1]^{S_?^2}$. A feasible region is full-dimensional if and only if there are no implicit inequalities (see, for example, [26, page 101]). An inequality defining the feasible region is implicit if the inequality is actually an equality for each point of the feasible region (see, for example, [26, page 99] for a formal definition).

▶ **Proposition 10.** *1. For all $(s,t) \in S_?^2$, there exists $d \in S_?^2 \to \mathbb{R}$ satisfying (1)–(3) such that $d(s,t) > 0$.*
2. For all $(s,t) \in S_?^2$, there exists $d \in S_?^2 \to \mathbb{R}$ satisfying (1)–(3) such that $d(s,t) < 1$.
3. For all $(s,t) \in S_?^2$ and $\pi \in V(\Omega(\tau(s),\tau(t)))$, there exists $d \in S_?^2 \to \mathbb{R}$ satisfying (1)–(3) such that

$$d(s,t) < \sum_{(u,v)\in S_?^2} \pi(u,v)\, d(u,v) + \sum_{(u,v)\in S_1^2} \pi(u,v)$$

▶ **Theorem 11.** *The feasible region defined by (1)–(3) is full-dimensional.*

Recall that we need to precompute probabilistic bisimilarity to define the feasible region. If, instead, we were not to do so, then we would have to replace (3) with

$$d(s,t) \quad \leq \quad \sum_{(u,v)\in S^2\setminus S_1^2} d(u,v)\,\pi(u,v) + \sum_{(u,v)\in S_1^2} \pi(u,v) \tag{4}$$

Now consider the labelled Markov chain consisting of a single state s. In that case the above inequality amounts to $d(s,s) \leq d(s,s)$ which is implicit and, hence, the feasible region is not full-dimensional.

5 Policy Iteration

In this section we present our modification of the basic algorithm of Bacci et al. [1]. Our optimization will be discussed in Section 6. Before we can present our algorithm, we need one more ingredient.

▶ **Definition 12.** The function $\Lambda : [0,1]^{S\times S} \to [0,1]^{S\times S}$ is defined by

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

The following result is proved in [8, Proposition 17]. This result will be instrumental in several proofs later in this paper.

▶ **Lemma 13.** *δ is the unique fixed point of Λ.*

Now that we have all the ingredients, we can present our modification of the basic algorithm by Bacci et al. to compute δ. This modification was first presented in [28].

1 **for each** $(s,t) \in S_?^2$
2 $\quad T(s,t) \leftarrow$ **an element of** $V(\Omega(\tau(s),\tau(t)))$
3 **while** $\exists (s,t) \in S_?^2 : \Lambda(\gamma^T)(s,t) < \gamma^T(s,t)$
4 $\quad T(s,t) \leftarrow \underset{\omega\in V(\Omega(\tau(s),\tau(t)))}{\arg\min} \sum_{u,v\in S} \omega(u,v)\,\gamma^T(u,v)$

The only difference with the algorithm of Bacci et al. is that we use Λ instead of Δ in line 3. In line 2, for each $(s,t) \in S_?^2$, a vertex of $\Omega(\tau(s),\tau(t))$ can be easily obtained in polynomial time by using, for example, Hitchcock's north west corner rule [14]. As we already mentioned, the fixed point γ^T, used in line 3 and 4, captures reachability probabilities and it is well known that these can be computed in polynomial time by solving a system of linear equations (see, for example, [3, Section 10.1.1]). Let $(s,t) \in S_?^2$. To compute $\Lambda(\gamma^T)(s,t)$ in line 3 we first have to decide probabilistic bisimilarity. This can be done in polynomial time as has been shown by Baier in [2]. Computing $\Lambda(\gamma^T)(s,t)$ boils down to solving a minimum cost network flow problem, where ω represents the flow and γ^T captures the cost. This problem can be solved in polynomial time using, for example, Orlin's network simplex algorithm [23]. This algorithm not only computes the minimum cost but also a vertex ω of $\Omega(\tau(s),\tau(t))$ at which that minimum cost is attained, which we use in line 4.

First, we prove the partial correctness of the above algorithm, that is, if the algorithm terminates then it computes the probabilistic bisimilarity distances. Hence, we have to show that at termination γ^T captures δ. Our proofs here are different from the ones presented in [1, 28] as we will use them as a stepping stone towards proving the partial policy version correct.

▶ **Theorem 14.** *For all $T \in \mathcal{T}$, if $\gamma^T(s,t) \leq \Lambda(\gamma^T)(s,t)$ for all $(s,t) \in S_?^2$, then $\gamma^T = \delta$.*

As we already mentioned earlier, for each $(s,t) \in S_?^2$, the set $V(\Omega(\tau(s),\tau(t)))$ is finite. Since we restrict our attention to labelled Markov chains with finitely many states, the set $S_?^2$ is finite as well. Therefore, the set $\mathcal{T}$ is finite. To prove that the loop terminates we show that T becomes smaller in every iteration.

▶ **Definition 15.** The order $\prec$ on $\mathcal{T}$ is defined by $T \prec U$ if $\gamma^T \sqsubset \gamma^U$.

To relate the value of T at the beginning of the loop with its value at the end of the loop, we introduce the following notation.

▶ **Definition 16.** Let $T \in \mathcal{T}$, $(s,t) \in S_?^2$ and $\omega \in V(\Omega(\tau(s),\tau(t)))$. The function $T[(s,t)/\omega] : S_?^2 \to \mathit{Distr}(S \times S)$ is defined by

$$T[(s,t)/\omega](u,v) = \begin{cases} \omega & \text{if } (u,v) = (s,t) \\ T(u,v) & \text{otherwise} \end{cases}$$

Clearly, $T[(s,t)/\omega] \in \mathcal{T}$. Next, we show that T indeed becomes smaller in every iteration of the loop, and as a consequence the loop terminates.

▶ **Theorem 17.** *For all $T \in \mathcal{T}$ and $(s,t) \in S_?^2$, if $\Lambda(\gamma^T)(s,t) < \gamma^T(s,t)$, then $T[(s,t)/\pi] \prec T$, where $\pi = \operatorname{arg\,min}_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{u,v \in S} \omega(u,v)\, \gamma^T(u,v)$.*

In the above simple policy iteration algorithm, in each iteration of the loop the policy is adjusted for a single state pair (s,t) which is locally non-optimal, that is, $\Lambda(\gamma^T)(s,t) < \gamma^T(s,t)$. In our general policy iteration algorithm, the policy is updated for all state pairs which are locally non-optimal.

```
1 for each (s,t) ∈ S_?^2
2     T(s,t) ← an element of V(Ω(τ(s),τ(t)))
3 while ∃(s,t) ∈ S_?^2 : Λ(γ^T)(s,t) < γ^T(s,t)
4     U ← T
5     for each (s,t) ∈ S_?^2 such that Λ(γ^U)(s,t) < γ^U(s,t)
6         T(s,t) ← arg min_{ω ∈ V(Ω(τ(s),τ(t)))} Σ_{u,v∈S} ω(u,v) γ^U(u,v)
```

The proof of partial correctness is the same as for the simple policy iteration algorithm. To prove termination, we slightly generalise Theorem 17.

6 Partial Policy Iteration

As proposed by Bacci et al. in [1], if we are only interested in the probabilistic bisimilarity distances of some of the state pairs, then we may be able to cut down on the number of state pairs for which we need to compute the probabilistic bisimilarity distance. Instead of total policies, we use partial ones. Hence, we generalise the set of total policies $\mathcal{T}$ of Definition 7 as follows. We denote the set of partial functions from $S_?^2$ to $\mathit{Distr}(S \times S)$ by $S_?^2 \nrightarrow \mathit{Distr}(S \times S)$ and the domain of such a function P by $\mathrm{dom}(P)$.

▶ **Definition 18.** For a labelled Markov chain $\langle S, L, \tau, \ell \rangle$, the set $\mathcal{P}$ of partial policies is defined by

$$\mathcal{P} = \{\, P \in S_?^2 \nrightarrow \mathit{Distr}(S \times S) \mid \forall (s,t) \in \mathrm{dom}(P) : P(s,t) \in V(\Omega(\tau(s),\tau(t))) \,\}.$$

Recall that S^2_0, S^2_1 and $S^2_?$ form a partition of $S \times S$. For a given $P \in \mathcal{P}$, S^2_0, S^2_1, $S^2_? \setminus \mathrm{dom}(P)$ and $S^2_? \cap \mathrm{dom}(P)$ form a partition of $S \times S$ as well. This partition is used to generalise the function Γ^T of Definition 8 to the partial setting.

▶ **Definition 19.** Let $P \in \mathcal{P}$. The function $\Theta^P : [0,1]^{S\times S} \to [0,1]^{S\times S}$ is defined by

$$\Theta^P(d)(s,t) = \begin{cases} 1 & \text{if } \ell(s) \neq \ell(t) \\ 0 & \text{if } s \sim t \vee (\ell(s) = \ell(t) \wedge s \not\sim t \wedge (s,t) \notin \mathrm{dom}(P)) \\ \sum\limits_{u,v\in S} P(s,t)(u,v)\, d(u,v) & \text{otherwise} \end{cases}$$

Since $[0,1]^{S\times S}$ is a complete lattice and Θ^P is a monotone function, we can conclude from the Knaster-Tarski fixed point theorem that Θ^P has a least fixed point. We denote this fixed point by θ^P.

The set $Q \subseteq S^2_?$ contains those pairs of states for which we want to compute their distances. Recall that for pairs of states in $(S \times S) \setminus S^2_? = S^2_0 \cup S^2_1$ we can obtain their distances by deciding probabilistic bisimilarity and comparing their labels.

```
1  P ← the partial function with empty domain
2  for each (s,t) ∈ Q
3      P(s,t) ← an element of V(Ω(τ(s),τ(t)))
4      expand(P,s,t)
5  while ∃(s,t) ∈ dom(P) : Λ(θ^P)(s,t) < θ^P(s,t)
6      P(s,t) ← arg min_{ω∈V(Ω(τ(s),τ(t)))} Σ_{u,v∈S} ω(u,v) θ^P(u,v)
7      expand(P,s,t)
```

Let $P \in \mathcal{P}$ and $(s,t) \in S^2_?$. The recursive function expand(P,s,t) is defined as follows.

```
8  while ∃(u,v) ∈ support(P(s,t)) ∩ S²_? : (u,v) ∉ dom(P)
9      P(u,v) ← an element of V(Ω(τ(u),τ(v)))
10     expand(P,u,v)
```

To prove properties of this recursive function, we introduce the following predicate.

▶ **Definition 20.** Let $P \in \mathcal{P}$ and $X \subseteq S^2_?$. The predicate $F(P,X)$ is defined by

$$F(P,X) = \forall (s,t) \in \mathrm{dom}(P) \setminus X : \mathrm{support}(P(s,t)) \cap S^2_? \subseteq \mathrm{dom}(P).$$

Roughly, this predicate $F(P,X)$ captures that P is fully defined when we exclude X from its domain. Let $P \in \mathcal{P}$ and $(s,t) \in S^2_?$. Next, we prove that for expand(P,s,t) the precondition $F(P,X)$ implies the postcondition $F(P, X \setminus \{(s,t)\})$. First, we observe that $F(P,X)$ is a loop invariant. At the start of line 10, we have that $F(P, X \cup \{(u,v)\})$. Hence, at the end of line 10 we have $F(P,X)$. To conclude that the loop terminates, we observe that the finite set $\mathrm{dom}(P) \setminus \mathrm{support}(P(s,t))$ becomes smaller in every iteration. Note that expand does not give rise to infinite recursion since for each recursive call the finite set $S^2_? \setminus \mathrm{dom}(P)$ becomes smaller. At the end of the loop we have $F(P,X)$ and $(u,v) \in \mathrm{dom}(P)$ for all $(u,v) \in \mathrm{support}(P(s,t)) \cap S^2_?$, that is, $\mathrm{support}(P(s,t)) \cap S^2_? \subseteq \mathrm{dom}(P)$. Therefore, $F(P, X \setminus \{(s,t)\})$.

The proof of partial correctness of this simple partial policy iteration algorithm is similar to the partial correctness proof provided in Section 5. If the above algorithm terminates, then we have that

$$F(P,\emptyset) \wedge Q \subseteq \mathrm{dom}(P) \wedge \forall (s,t) \in \mathrm{dom}(P) : \theta^P(s,t) \leq \Lambda(\theta^P)(s,t)$$

at termination. As we will show next, from the above we can conclude that θ^P and δ coincide on Q and, hence, θ^P contains the probabilistic bisimilarity distances of Q.

▶ **Theorem 21.** *For all $P \in \mathcal{P}$, if $F(P, \emptyset)$ and $\theta^P(s,t) \leq \Lambda(\theta^P)(s,t)$ for all $(s,t) \in \mathrm{dom}(P)$, then $\theta^P(s,t) = \delta(s,t)$ for all $(s,t) \in \mathrm{dom}(P)$.*

It remains to prove that the simple partial policy iteration algorithm terminates. As we already discussed above, the recursive function expand terminates. Hence, we are left to show that the loop of line 5–7 terminates as well. We prove this by showing that in each iteration of the loop $\langle S^2_? \setminus \mathrm{dom}(P), P\rangle$ becomes smaller. These pairs are ordered lexicographically, with the first component ordered by $\subset$ and the second component ordered by $\prec$, as introduced in Definition 15. Assume that P is updated for (s,t) in line 6 of the current iteration of the loop. We distinguish two cases. If $(u,v) \notin \mathrm{dom}(P)$ for some $(u,v) \in \mathrm{support}(P(s,t)) \cap S^2_?$, then expand$(P,s,t)$ in line 7 will assign a value to $P(u,v)$ in line 9 of the expand function. As a consequence, $\mathrm{dom}(P)$ becomes bigger and, hence, $S^2_? \setminus \mathrm{dom}(P)$ becomes smaller, that is, the first component becomes smaller. Note that in this case P may not become smaller as $\theta^P(u,v)$ was zero and may have become positive. Otherwise, $\mathrm{support}(P(s,t)) \cap S^2_? \subseteq \mathrm{dom}(P)$. In that case, the expand function does not perform any assignments to P and, therefore, $\mathrm{dom}(P)$ stays the same. Thus, the first component stays the same. Furthermore, the iteration changes the partial policy from P to $P[(s,t)/\pi]$ (cf. Definition 16). As we show next, in this case the second component, that is, the partial policy, becomes smaller.

▶ **Theorem 22.** *For all $P \in \mathcal{P}$ and $(s,t) \in \mathrm{dom}(P)$, if $\Lambda(\theta^P)(s,t) < \theta^P(s,t)$, then $P[(s,t)/\pi] \prec P$, where $\pi = \underset{\omega \in V(\Omega(\tau(s),\tau(t)))}{\arg\min} \sum_{u,v \in S} \omega(u,v)\, \theta^P(u,v)$.*

The algorithm of Bacci et al. differs in three major ways from our simple partial policy iteration algorithm. First of all, as we already mentioned in Section 5, they use Δ instead of Λ in line 5. In [28, Theorem 8] we give an example different from the one presented below which shows that Λ is essential for computing the distances correctly. Secondly, in line 5 they consider only those state pairs (s,t) that are reachable from state pairs in Q in the Markov chain $\langle S \times S, P\rangle$, instead of those in $\mathrm{dom}(P)$. But, as we show below, as a result they do not always correctly compute the distances. Thirdly, they assign one to $\theta^P(s,t)$ when $(s,t) \notin \mathrm{dom}(P)$ whereas we assign zero. An example similar to the one presented below can be used to show that this also gives to computing the distances incorrectly.

We conclude this section with an example that shows that Bacci et al. do not always consider partial policies that are defined for sufficiently many state pairs. Consider the labelled Markov chain presented in the introduction. Assume that we are only interested in the probabilistic bisimilarity distance between the states s and t. That is, $Q = \{(s,t)\}$. After executing line 1–4 of the simple partial policy iteration algorithm, we may end up with the partial policy P defined by

$$P(s,t) = \tfrac{1}{2}\mathrm{Dir}_{(a,d)} + \tfrac{1}{2}\mathrm{Dir}_{(b,c)} \quad P(a,d) = \tfrac{1}{2}\mathrm{Dir}_{(a_1,d_2)} + \tfrac{1}{2}\mathrm{Dir}_{(a_2,d_1)}$$
$$P(b,c) = \tfrac{1}{2}\mathrm{Dir}_{(b_1,c_2)} + \tfrac{1}{2}\mathrm{Dir}_{(b_2,c_1)}$$

where the Dirac distribution $\mathrm{Dir}_{(u,v)}$ is defined by

$$\mathrm{Dir}_{(u,v)}(x,y) = \begin{cases} 1 & \text{if } (x,y) = (u,v) \\ 0 & \text{otherwise} \end{cases}$$

At this point, we have $\theta^P(s,t) = 1$, $\theta^P(a,d) = 1$ and $\theta^P(b,c) = 1$. Note that (s,t) is not locally optimal, that is, $\Lambda(\theta^P)(s,t) < \theta^P(s,t)$. We update P by setting $P(s,t) = \frac{1}{2}\mathrm{Dir}_{(a,c)} +$

$\frac{1}{2}\mathrm{Dir}_{(b,d)}$. The expand function on line 7 of the simple partial policy iteration algorithm may give rise to

$$P(a,c) = \tfrac{1}{2}\mathrm{Dir}_{(a_1,c_1)} + \tfrac{1}{2}\mathrm{Dir}_{(a_2,c_2)} \quad P(b,d) = \tfrac{1}{2}\mathrm{Dir}_{(b_1,d_1)} + \tfrac{1}{2}\mathrm{Dir}_{(b_2,d_2)}$$

At this point, we have $\theta^P(s,t) = \frac{3}{4}$, $\theta^P(a,c) = 1$ and $\theta^P(b,d) = \frac{1}{2}$. Since in their algorithm, Bacci et al. only check local optimality for all state pairs reachable from (s,t) in the Markov chain $\langle S \times S, P\rangle$, that is, for (s,t), (a,c) and (b,d), and all three are locally optimal, their algorithm terminates at this point. Our algorithm checks for local optimality for all state pairs in $\mathrm{dom}(P)$. Since neither (a,d) nor (b,c) are locally optimal, our algorithm continues. We update P by setting

$$P(a,d) = \tfrac{1}{2}\mathrm{Dir}_{(a_1,d_1)} + \tfrac{1}{2}\mathrm{Dir}_{(a_2,d_2)} \quad P(b,c) = \tfrac{1}{2}\mathrm{Dir}_{(b_1,c_1)} + \tfrac{1}{2}\mathrm{Dir}_{(b_2,c_2)}$$

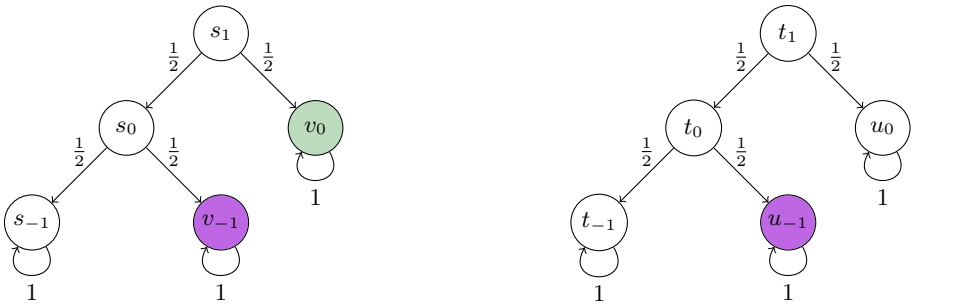
At this point, we have $\theta^P(s,t) = \frac{3}{4}$, $\theta^P(a,d) = \frac{1}{2}$ and $\theta^P(b,c) = \frac{1}{2}$. Since (s,t) is not locally optimal any more, we update P by setting $P(s,t) = \frac{1}{2}\mathrm{Dir}_{(a,d)} + \frac{1}{2}\mathrm{Dir}_{(b,c)}$. This results in $\theta^P(s,t) = \frac{1}{2}$ which is the probabilistic bisimilarity distance of (s,t).

Also the general policy iteration algorithm, which we presented at the end of Section 5, can be generalised to use partial policies instead of total ones.

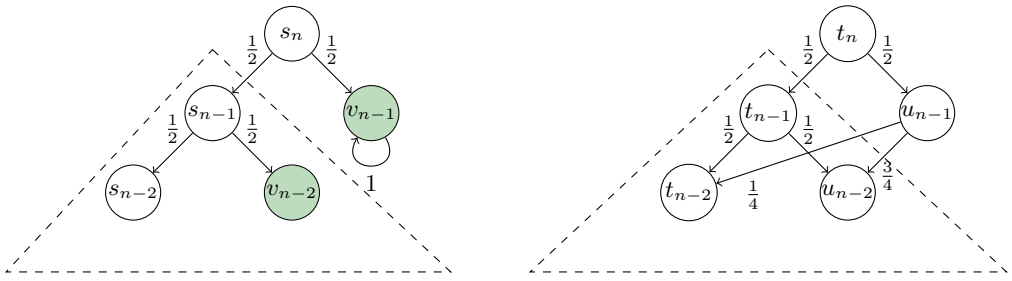
7 An Exponential Lower Bound

Below, we analyze the worst case running time of the simple partial policy iteration algorithm. We show that it is at least exponential in the number of states of the labelled Markov chain.

▶ **Definition 23.** For $n \in \mathbb{N}$, the labelled Markov chain $\mathcal{M}_n$ is defined as follows by induction on n. The labelled Markov chain $\mathcal{M}_0$ is defined as



If $n > 0$ then the labelled Markov chain $\mathcal{M}_n$ is defined as



where the two dashed triangles together represent the labelled Markov chain $\mathcal{M}_{n-1}$.

The above labelled Markov chain $\mathcal{M}_n$ has $4n+10$ states and $7n+14$ transitions. Next, we show that it may take at least $2^{n+1}-1$ iterations of the simple partial policy iteration algorithm to compute the distance of $s_n$and t_n in $\mathcal{M}_n$.

The partial policy iteration algorithm contains some nondeterminism. In particular, in line 3 and 9, an element of $V(\Omega(\tau(s),\tau(t)))$ and $V(\Omega(\tau(u),\tau(v)))$ is chosen. Furthermore, in line 5 a state pair $(s,t) \in \mathrm{dom}(P)$ with $\Lambda(\theta^P)(s,t) < \theta^P(s,t)$ is selected. Note that, for all $1 \leq i \leq n$,

$$V(\Omega(\tau(s_i),\tau(t_i))) = \{\tfrac{1}{2}\mathrm{Dir}_{(s_{i-1},t_{i-1})} + \tfrac{1}{2}\mathrm{Dir}_{(v_{i-1},u_{i-1})}, \tfrac{1}{2}\mathrm{Dir}_{(s_{i-1},u_{i-1})} + \tfrac{1}{2}\mathrm{Dir}_{(v_{i-1},t_{i-1})}\}.$$

Also,

$$V(\Omega(\tau(s_0),\tau(t_0))) = \{\tfrac{1}{2}\mathrm{Dir}_{(s_{-1},u_{-1})} + \tfrac{1}{2}\mathrm{Dir}_{(v_{-1},t_{-1})}, \tfrac{1}{2}\mathrm{Dir}_{(s_{-1},t_{-1})} + \tfrac{1}{2}\mathrm{Dir}_{(v_{-1},u_{-1})}\}.$$

Furthermore, for all $1 \leq i < n$,

$$\begin{aligned} V(\Omega(\tau(s_i),\tau(u_i))) \quad = \quad & \{\tfrac{1}{2}\mathrm{Dir}_{(v_{i-1},u_{i-1})} + \tfrac{1}{4}\mathrm{Dir}_{(s_{i-1},t_{i-1})} + \tfrac{1}{4}\mathrm{Dir}_{(s_{i-1},u_{i-1})}, \\ & \tfrac{1}{2}\mathrm{Dir}_{(s_{i-1},u_{i-1})} + \tfrac{1}{4}\mathrm{Dir}_{(v_{i-1},u_{i-1})} + \tfrac{1}{4}\mathrm{Dir}_{(v_{i-1},t_{i-1})}\}. \end{aligned}$$

To realize the exponential lower bound, in line 3 and 9 we choose the first element of the above sets and in line 5 we select the (s_i,t_i) with maximal index i.

▶ **Theorem 24.** *For each $n \in \mathbb{N}$, there exists a labelled Markov chain of size $O(n)$ and a singleton set Q such that simple partial policy iteration takes $\Omega(2^n)$ iterations to compute the distances for the state pair in Q.*

8 Experimental Results

Next we present results from experiments comparing the performance of six different algorithms: the algorithm described in the introduction by Van Breugel et al. [5] based on the first order theory over the reals, the algorithm by Chen et al. [8] based on the ellipsoid method, our simple policy iteration algorithm, our general policy iteration algorithm, our simple partial policy iteration algorithm and our general partial policy iteration algorithm. We implemented all the algorithms in Java. In our implementations we do not use arbitrary precision arithmetic. These implementations were run on a number of labelled Markov chains. These labelled Markov chains model well-known randomized algorithms and were obtained from examples of probabilistic model checkers such as PRISM [20] and jpf-probabilistic [31].

For each labelled Markov chain, we executed the code ten times. The first few executions were discarded to account for the "warm-up" time that the Java virtual machine needs to perform just-in-time compilation and optimization. For the remaining runs the average running time and the standard deviation were computed for each labelled Markov chain.

Since the distance function is symmetric and the distance from a state to itself is zero, we only need to compute the distance of $\frac{n^2-n}{2}$ state pairs for a labelled Markov chain with n states. Recall that the policy iteration algorithms precompute distances for state pairs that are probabilistic bisimilar or have different labels. Hence, only state pairs with the same label that are not probabilistic bisimilar are considered. In general, the partial policy iteration algorithms compute the distances for even fewer state pairs. For our experiments, we report for each policy iteration algorithm how many state pairs it considers and how many iterations it takes.

The first example we consider is a version of quicksort in which the pivot is chosen randomly. An implementation of this algorithm is part of jpf-probabilistic and this tool can be used to obtain the corresponding labelled Markov chain. The size of the labelled Markov chain grows exponentially in the size of the input, which is the list to be sorted. For example, lists of size 4, 5 and 6 give rise to labelled Markov chains with 10, 28 and 82 states, respectively.

The first order theory over the reals algorithm can only handle labelled Markov chains with a handful of states. We ran the algorithm on the labelled Markov chain with 10 states. It did not terminate within three days. The ellipsoid method takes on average 73 seconds. For the partial algorithms we compute the distance for a single pair of states. Of the 45 state pairs, the policy iteration algorithms consider 8 state pairs and the partial policy iteration algorithms only 3 state pairs. All policy iteration algorithms take less than 45 milliseconds. That makes the ellipsoid method three orders of magnitude slower than our policy iteration algorithms for this small example.

We did not run the first order theory over the reals algorithm on the randomized quicksort example with 28 states. The ellipsoid method takes more than 43 hours, making it five orders of magnitude slower than the policy iteration algorithms, which take less than a dozen seconds. For the example with 82 states, of the 3,321 state pairs the policy iteration algorithms consider 870 states and the partial policy iteration algorithms only 13 states.

Algorithm	List size	Running time	Standard deviation	State pairs	Iterations
Simple	4	42.75 ms	5.96 ms	8	2
General	4	27.93 ms	4.48 ms	8	2
Simple partial	4	12.23 ms	2.84 ms	3	2
General partial	4	8.84 ms	2.25 ms	3	1
Simple	5	12.00 s	19.38 ms	99	13
General	5	4.58 s	20.39 ms	99	13
Simple partial	5	51.58 ms	2.51 ms	3	0
General partial	5	81.62 ms	5.67 ms	3	1
Simple	6	15.61 h	4.00 m	870	159
General	6	47.30 m	6.03 s	870	154
Simple partial	6	24.48 s	162.91 ms	13	3
General partial	6	53.63 s	203.58 ms	13	4

In [19], Knuth and Yao show how to model a die by means of a fair coin. An implementation of their algorithm is part of PRISM. The resulting labelled Markov chain has 13 states.

Algorithm	Running time	Standard deviation	State pairs	Iterations
Simple	266.45 ms	22.40 ms	29	2
General	155.05 ms	6.59 ms	29	2
Simple partial	52.73 ms	3.69 ms	7	2
General partial	39.57 ms	0.83 ms	7	3

In the next experiment, we model two dice, one using only a fair coin and the other one using a biased coin with probability 0.6 for heads and 0.4 for tails. The goal is to compute the probabilistic bisimilarity distance between the two dice. The resulting labelled Markov chain has 20 states.

Algorithm	Running time	Standard deviation	State pairs	Iterations
Simple	10.99 s	69.67 ms	91	51
General	1.51 s	20.45 ms	91	51
Simple partial	1.28 s	12.00 ms	21	17
General partial	0.55 s	19.83 ms	21	18

It can be seen from the above experiments that the simple policy iteration algorithm is often slower than the general policy iteration algorithm. Moreover, if we are only interested in the probabilistic bisimilarity distances between a few states, the partial policy iteration algorithms are much more efficient as only part of the labelled Markov chain is considered and only the distances of related pairs of states are computed.

9 Conclusion

Although behavioural equivalences like probabilistic bisimilarity are not robust, they still play a pivotal role when computing their quantitative generalisations. For example, as we have seen in Section 4, the feasible region is defined in terms of probabilistic bisimilarity. The policies used in the policy iteration algorithms are only defined for states that are not probabilistic bisimilar. Hence, in both cases one first has to decide probabilistic bisimilarity. Deciding probabilistic bisimilarity takes only a fraction of the time to compute the probabilistic bisimilarity distances. For all the examples discussed in Section 8, deciding probabilistic bisimilarity takes less than 50 milliseconds.

The probabilistic bisimilarity pseudometric of Desharnais et al. also has discounted variants (see [12] for details). Bacci et al. consider the undiscounted version, as we do in this paper, as well as the discounted variants in [1]. All the results presented in this paper carry over to the discounted setting.

In [28] we prove an exponential lower bound for the simple policy iteration algorithm. In this paper, we present an exponential lower bound for our simple partial policy iteration algorithm. It is still open whether there exists an exponential lower bound for our general policy iteration algorithm.

Acknowledgements. The authors are thankful to all the referees of this paper for their constructive feedback.

References

1. Giorgio Bacci, Giovanni Bacci, Kim Larsen, and Radu Mardare. On-the-fly exact computation of bisimilarity distances. In Nir Piterman and Scott Smolka, editors, *Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 7795 of *Lecture Notes in Computer Science*, pages 1–15, Rome, Italy, March 2013. Springer-Verlag.
2. Christel Baier. Polynomial time algorithms for testing probabilistic bisimulation and simulation. In Rajeev Alur and Thomas Henzinger, editors, *Proceedings of the 8th International Conference on Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 50–61, New Brunswick, NJ, USA, July/August 1996. Springer-Verlag.
3. Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, Cambridge, MA, USA, 2008.
4. Franck van Breugel. Probabilistic bisimilarity distances. To appear in ACM SIGLOG News, 2017.
5. Franck van Breugel, Babita Sharma, and James Worrell. Approximating a behavioural pseudometric without discount. *Logical Methods in Computer Science*, 4(2), April 2008.
6. Franck van Breugel and James Worrell. The complexity of computing a bisimilarity pseudometric on probabilistic automata. In Franck van Breugel, Elham Kashefi, Catuscia Palamidessi, and Jan Rutten, editors, *Horizons of the Mind – A Tribute to Prakash Panangaden*, volume 8464 of *Lecture Notes in Computer Science*, pages 191–213. Springer-Verlag, 2014.
7. John Canny. Some algebraic and geometric computations in PSPACE. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 460–467, Chicago, IL, USA, May 1988. ACM.
8. Di Chen, Franck van Breugel, and James Worrell. On the complexity of computing probabilistic bisimilarity. In Lars Birkedal, editor, *Proceedings of the 15th International Conference on Foundations of Software Science and Computational Structures*, volume 7213 of *Lecture Notes in Computer Science*, pages 437–451, Tallinn, Estonia, March/April 2012. Springer-Verlag.
9. Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, February 1992.
10. Anne Condon. On algorithms for simple stochastic games. In Jin-Yi Cai, editor, *Advances in Computational Complexity Theory*, volume 13 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 51–73. American Mathematical Society, 1993.
11. Brian Davey and Hilary Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, United Kingdom, 2002.
12. Josée Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Metrics for labeled Markov systems. In Jos Baeten and Sjouke Mauw, editors, *Proceedings of the 10th International Conference on Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 258–273, Eindhoven, The Netherlands, August 1999. Springer-Verlag.
13. Alessandro Giacalone, Chi-Chang Jou, and Scott Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of the IFIP WG 2.2/2.3 Working Conference on Programming Concepts and Methods*, pages 443–458, Sea of Gallilee, Israel, April 1990. North-Holland.
14. Frank Hitchcock. The distribution of a product from several sources to numerous localities. *Studies in Applied Mathematics*, 20(1/4):224–230, April 1941.
15. Alan Hoffman and Richard Karp. On nonterminating stochastic games. *Management Science*, 12(5):359–370, January 1966.

16 Bengt Jonsson and Kim Larsen. Specification and refinement of probabilistic processes. In *Proceedings of the 6th Annual Symposium on Logic in Computer Science*, pages 266–277, Amsterdam, The Netherlands, July 1991. IEEE.

17 Leonid Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20(1):191–194, 1979.

18 Viktor Klee and Christoph Witzgall. Facets and vertices of transportation polytopes. In George B. Dantzig and Arthur F. Veinott, editors, *Proceedings of 5th Summer Seminar on the Mathematis of the Decision Sciences*, volume 11 of *Lectures in Applied Mathematics*, pages 257–282, Stanford, CA, USA, July/August 1967. AMS.

19 Donald E. Knuth and Andrew C. Yao. The complexity of nonuniform random number generation. In J.F. Traub, editor, *Proceedings of a Symposium on New Directions and Recent Results in Algorithms and Complexity*, pages 375–428, Pittsburgh, April 1976. Academic Press.

20 Marta Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Proceedings of the 23rd International Conference on Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591, Snowbird, UT, USA, July 2011. Springer-Verlag.

21 Kim Larsen and Arne Skou:. Bisimulation through probabilistic testing. In *Proceedings of the 16th Annual ACM Symposium on Principles of Programming Languages*, pages 344–352, Austin, TX, USA, January 1989. ACM.

22 Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Germany, 1980.

23 James Orlin. A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming*, 78(2):109–129, August 1997.

24 Prakash Panangaden. *Labelled Markov Processes*. Imperial College Press, London, United Kingdom, 2009.

25 David Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *Proceedings of 5th GI-Conference on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183, Karlsruhe, Germany, March 1981. Springer-Verlag.

26 Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester, UK, 1986.

27 Lloyd Shapley. Stochastic games. *Proceedings of the Academy of Sciences*, 39(10):1095–1100, October 1953.

28 Qiyi Tang and Franck van Breugel. Computing probabilistic bisimilarity distances via policy iteration. In Josée Desharnais and Radha Jagadeesan, editors, *Proceedings of the 27th International Conference on Concurrency Theory*, volume 59 of *Leibniz International Proceedings in Informatics*, pages 22:1–22:15, Quebec City, QC, Canada, August 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

29 Alfred Tarski. *A decision method for elementary algebra and geometry*. University of California Press, Berkeley, CA, USA, 1951.

30 Alfred Tarski. A lattice-theoretic fixed point theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, June 1955.

31 Xin Zhang and Franck van Breugel. Model checking randomized algorithms with Java PathFinder. In *Proceedings of the 7th International Conference on the Quantitative Evaluation of Systems*, pages 157–158, Williamsburg, VA, USA, September 2010. IEEE.