COMPUTING PROBABILISTIC BISIMILARITY DISTANCES

QIYI TANG

A DISSERTATION SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO
AUGUST 2018

Abstract

Behavioural equivalences like probabilistic bisimilarity rely on the transition probabilities and, as a result, are sensitive to minuscule changes of those probabilities. Such behavioural equivalences are not robust, as first observed by Giacalone, Jou and Smolka. Probabilistic bisimilarity distances, a robust quantitative generalization of probabilistic bisimilarity, capture the similarity of the behaviour of states of a probabilistic model. The smaller the distance, the more alike the states behave. In particular, states are probabilistic bisimilar if and only if the distance between them is zero. In this dissertation, we focus on algorithms to compute probabilistic bisimilarity distances for two probabilistic models: labelled Markov chains and probabilistic automata.

In the late nineties, Desharnais, Gupta, Jagadeesan and Panangaden defined probabilistic bisimilarity distances on the states of a labelled Markov chain. This provided a quantitative generalization of probabilistic bisimilarity, which was introduced by Larsen and Skou a decade earlier. Several algorithms to approximate and compute these probabilistic bisimilarity distances have been put forward. In this

dissertation, we correct and generalize some of these policy iteration algorithms. Moreover, we develop several new algorithms which have better performance in practice and can handle much larger systems.

Similarly, Deng, Chothia, Palamidessi and Pang presented probabilistic bisimilarity distances on the states of a probabilistic automaton. This provided a robust quantitative generalization of probabilistic bisimilarity introduced by Segala and Lynch. Although the complexity of computing probabilistic bisimilarity distances for probabilistic automata has already been studied and shown to be in $\mathbf{NP} \cap \mathbf{coNP}$ and **PPAD**, we are not aware of any practical algorithms to compute those distances. In this dissertation, we provide several key results that may prove to be useful for the development of algorithms to compute probabilistic bisimilarity distances for probabilistic automata. In particular, we present a polynomial time algorithm that decides distance one. Furthermore, we give an alternative characterization of the probabilistic bisimilarity distances as a basis for a policy iteration algorithm.

Acknowledgements

I would like to express sincere appreciation to my dissertation supervisor, Franck van Breugel, for his continuous generous support, for his vision, wisdom, enthusiasm and patience, and for his always generosity with his time. I feel honoured to work with him. This journey, though with some ups and downs, has been a great pleasure.

I would like to thank Eric Ruppert for being a member of my supervisory committee, for his generosity with his time, advise and feedback.

I would like to thank Suprakash Datta for his support as a member of my supervisory committee.

I would like to express special thanks to Prakash Panangaden. His questions and comments as the external examiner have had a significant impact on the final version of this dissertation.

I would like to thank Jonathan Ostroff from whom I have picked up skills that impacted my research a lot.

I would like to thank Stephen Watson for being the internal member of my examining committee and asking thought provoking questions during the examination.

I would like to thank Giorgio Bacci and Giovanni Bacci, with whom I have discussed my research, for their time and good suggestions. I would also like to thank David Maclver for the interesting discussions of the research and his generosity with his time of reviewing this dissertation.

I would like to thank the anonymous referees of my papers for their constructive feedback that shaped parts of the dissertation.

Many thanks to my family, for always being there for me. Without their support and understanding, I would not have had the courage to pursue my dreams. Last but not least, I would like to thank my fiancé, Yulong Wu, for his love, inspiration, support and optimistic nature.

Table of Contents

List of Symbols

A	set of actions of Markov decision processes 78
α	available action function of Markov decision processes 78
A^*	optimal max policy for simple stochastic games 225
B	subset of set of state pairs with distance one 108
$\mathcal{C}_B$	set of couplings which excludes B 108
$\mathbf{c}$	cost function of Markov decision processes 78
D_1	set of state pairs with distance one 106
Distr	probability distribution 18
E	set of edges of simple stochastic games 218
ℓ	labelling function of labelled Markov chains and probabilistic automata 18, 44
Γ	a function used to define D_1 176
$\boldsymbol{\nu}$	greatest fixed point 22

H Hausdorff metric 48

I^* optimal min policy for simple stochastic games 225

K Kantorovich metric 33

L set of labels of labelled Markov chains and probabilistic automata 18, 44

Λ function to compute D_1 for probabilistic automata 249

$\boldsymbol{\mu}$ least fixed point 22

Δ distance function 21

Δ_1 distance function 42

Δ_B distance function parametrized by B 121

$\mathcal{A}$ set of max policies for simple stochastic games 214

$\mathcal{I}$ set of min policies for simple stochastic games 214

Ω set of couplings 19

ω coupling 19

$\mathbf{1}$ function that maps each element to one 31

P edge probability function for simple stochastic games 214

Δ_c distance function for probabilistic automata with discount factor c 220

Θ_B^P distance function parametrized by the partial policy P and B 156

Φ value function of Markov decision processes 83

π transition probability function for Markov decision processes 78

$\mathcal{P}$ set of partial policies 155

Ψ^T distance function used by Bacci et al. 114

$\mathcal{R}$ equivalence relation 20

S set of states of labelled Markov chains and probabilistic automata 18, 44

S^2 set of state pairs 18

S_0^2 set of state pairs which are probabilistic bisimilar 50

S_1^2 set of state pairs which have difference labels 50

$S_?^2$ set of state pairs which have the same labels but are not probabilistic bisimilar 50

τ transition function of labelled Markov chains 18

∞ terminal state of Markov decision processes 90

Θ_B^T distance function parametrized by the total policy T and B 110

$\mathcal{T}$	set of total policies 78
V	set of vertices of simple stochastic games 78
V_0	zero sink in simple stochastic games 214
V_1	one sink in simple stochastic games 214
$V_{\max}$	set of min vertices in simple stochastic games 214
$V_{\min}$	set of min vertices in simple stochastic games 213
V_{rnd}	set of random vertices in simple stochastic games 214
v^T	value function under the policy T 81
$\mathbf{0}$	function that maps each element to zero 31

1 Introduction

This chapter discusses the motivations behind this dissertation, the main objectives, and the contributions achieved in the dissertation.

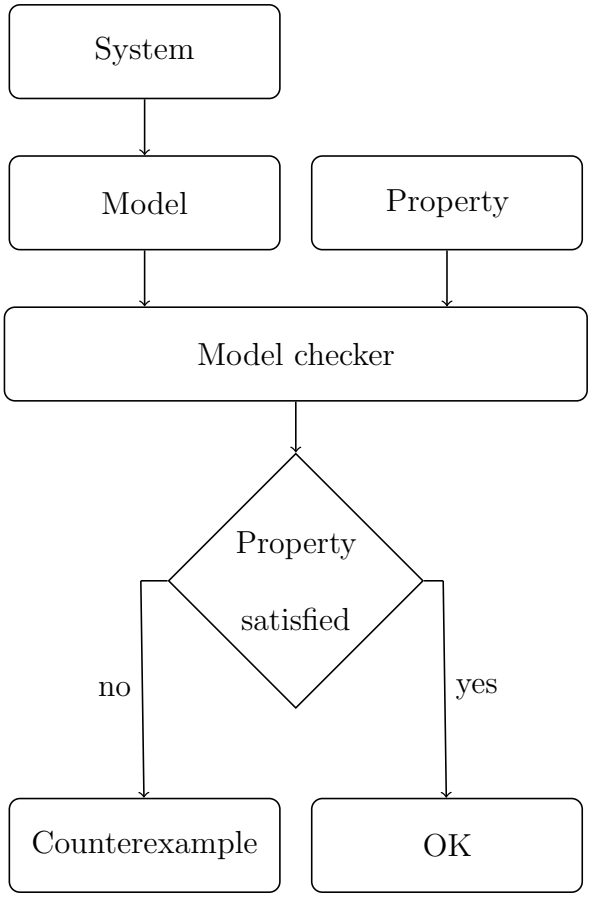
1.1 Behavioural Equivalences for Probabilistic Models

Checking for bugs in software is critical. *Software verification* is the process to verify that the designed system behaves the way it is supposed to. Though software verification is time consuming, failing to detect system errors can be costly and sometimes catastrophic (see, for example, [8, page 1-2]).

A *behavioural equivalence* captures which states of a model give rise to the same behaviour. In software verification, simulations and bisimulations [68, 73] are two important notions of behaviour equivalences which can be used to capture correctness criteria, such as, for example, linearizability [49] (see, for example, [39]). 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, Chapter 3]).

We refer the reader to, for example, [76, page 1–4], for an extensive discussion of the importance of behavioural equivalences.

Model checking is one of the most popular software verification techniques. It can show the absence of errors, which is its major advantage compared to other model-based verification techniques such as testing. The software tool which conducts model checking is called a model checker. The following graph shows the general process of model checking. There are three different phases.



- Firstly, a model is built for the software system. Usually, we model the

system's behaviour using a model description language of the model checker. Also, we need to determine the desired properties that the system should satisfy. These properties are expressed in a property specification language of the model checker.

- Secondly, the model checker is run to check the validity of the properties in the system model.

- Finally, the model checker terminates with three possible outcomes. It outputs OK if the system satisfies the desired properties or it outputs a counterexample otherwise. The model checker may not always terminate successfully if it has to explore too many states due to the so-called *state space explosion problem*. In such cases, it may run out of memory or time.

One of the techniques to tackle the state space explosion problem is to minimize the state space by collapsing those states which are bisimilar, that is, have equivalent behaviours (see [1, Chapter 3]). There are many other techniques to tackle the state space explosion problem, such as symbolic execution (see, for example, [59]) and partial order reduction (see, for example, [45]).

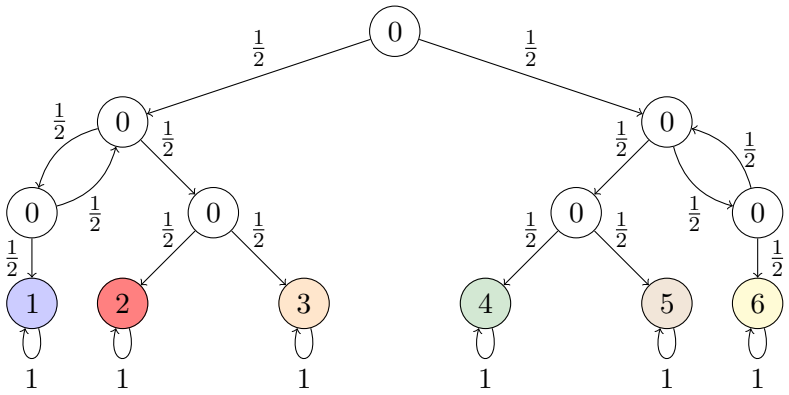
In this dissertation, we focus on quantitatively generalizations of behavioural equivalences for probabilistic models. These models can capture randomized algorithms, probabilistic protocols, biological systems and many other systems in which probabilities play a central role. In particular, we consider *labelled Markov chains*

and *probabilistic automata*.

1.2 Two Probabilistic Models and Their Distances

A labelled Markov chain is a Markov chain with labelled states. These labels provide a partition of the states so that states satisfying the same basic properties of interest are in the same partition. *Probabilistic bisimilarity* due to Larsen and Skou [64] is a key behavioural equivalence for labelled Markov chains. As shown by Katoen, Kemna, Zapreev and Jansen [57], minimizing a labelled Markov chain by identifying those states that are probabilistic bisimilar speeds up model checking. However, probabilistic bisimilarity only identifies those states that behave exactly the same with exactly the same probability.

The following example shows how the behaviour of rolling a die can be mimicked by flipping a coin, an example due to Knuth and Yao [62]. 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.



If 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, Jou and Smolka [43].

A *behavioural pseudometric* is a quantitative generalization of a behavioural equivalence. Such a pseudometric assigns to each pair of states a number in the unit interval $[0, 1]$. The smaller this number, the more alike the states behave. For the models that include quantitative information such as time and probabilities, behavioural pseudometrics are an essential complement to behavioural equivalences. For some historical background on this behavioural pseudometric we refer the reader to [21, Section 1].

The *probabilistic bisimilarity distances* for labelled Markov chains that we study

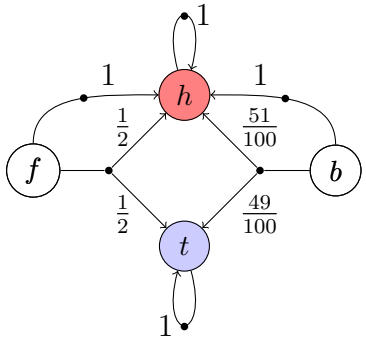
in this dissertation were first defined by Desharnais, Gupta, Jagadeesan and Panangaden in [31]. Their definition is based on a real-valued modal logic. This logic can be viewed as a function which maps a formula of the logic and a state of the labelled Markov chain to a real number in the unit interval of $[0, 1]$. The larger the number is, the more likely the state satisfies the formula. The distance between two states is defined as the difference of the formula which can distinguish them most. This pseudometric 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, that is, going to the left, for the biased coin, is 0.51. 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 [i]. We refer the reader to [15] for a more detailed discussion of a similar example.

Later, Van Breugel and Worrell [20] defined probabilistic bisimilarity distances

[i]The actual distance is $\frac{27251}{755000}$.

on labelled Markov chains as a fixed point of a function. The Kantorovich metric [56] is a key ingredient of this pseudometric. They also showed that their pseudometric coincides with the one defined by Desharnais *et al.* This pseudometric can also be characterized in terms of tests [17], as the values of a game (Chapter 5), in terms of a coalgebra [20] and a quantitative algebra [66]. It provides a natural generalization of probabilistic bisimilarity. It can be defined in terms of the Kantorovich metric, a natural distance function on probability distributions, and it can be elegantly characterized in terms of a logic, tests, a game, a coalgebra and a quantitative algebra. In this dissertation, we will use the definition of the probabilistic bisimilarity distances which is interpreted as a fixed point of a function.

The other model, probabilistic automata, was first studied by Segala in [78]. It captures not only probabilities but also nondeterminism (and, hence, concurrency). Let us consider a simple example.



The states of a probabilistic automaton are also labelled. In the above example, the labels are represented by colours. Each state has one or more probabilistic

transitions. For example, the state t has a single probabilistic transition that takes state t to itself with probability one. State f has two probabilistic transitions. One takes state f to state h with probability one. The other represents a fair coin toss. Also state b has two transitions, one of which represents a biased coin toss.

Segala and Lynch [79] introduced *probabilistic bisimilarity* for probabilistic automata. This behavioural equivalence for probabilistic automata generalizes the one introduced by Larsen and Skou [64]. States s and t of a probabilistic automaton are probabilistic bisimilar if for each outgoing probabilistic transition of state s there exists a matching outgoing probabilistic transition of state t, and vice versa. Two probabilistic transitions match if they both transition to each probabilistic bisimilarity equivalence class with the exact same probability. States f and b in the above example are not probabilistic bisimilar. Although the transition from state f to state h can be matched by the transition from state b to state h, the probabilistic transitions representing a fair and biased coin toss do not match since the probabilities are slightly different. Deng, Chothia, Palamidessi and Pang [29] introduced a behavioural pseudometric for probabilistic automata that generalizes probabilistic bisimilarity. The Hausdorff metric [47] and the Kantorovich metric [56] are key ingredients of this pseudometric. The former is used to capture nondeterminism. This idea dates back to the work of De Bakker and Zucker [10]. As we already mentioned earlier, the Kantorovich metric captures probabilistic behaviours.

On the one hand, the behaviours of the states h and t of the above example are very different since their labels are different. As a result, their probabilistic bisimilarity distance is one. On the other hand, the behaviours of the states f and b are very similar, which is reflected by the fact that these states have probabilistic bisimilarity distance $\frac{1}{100}$.

Tracol, Desharnais and Zhioua [91] also introduced a behavioural pseudometric for probabilistic automata. Their probabilistic bisimilarity distances generalize probabilistic bisimilarity as well, but are different from the ones introduced by Deng *et al.* An example showing the difference can be found in [91, Example 5]. To compute their probabilistic bisimilarity distances, they developed an iterative algorithm. In each iteration, a maximum flow problem needs to be solved. The resulting algorithm runs in polynomial time.

1.3 Algorithms to Compute the Probabilistic Bisimilarity Distances

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 tackle the state space explosion problem. Probabilistic bisimilarity distances can be used in a similar way, by identifying those

states that behave almost the same, that is, have a small distance (see [6, 69, 80]). Ferns, Panangaden, and Precup [36] show that the probabilistic bisimilarity distances can also be used in model approximation, that is, the probabilistic bisimilarity distances can provide error bounds between the correct and the approximate value function of some approximation schemes. Note that, Ferns and Precup [37] also show that the probabilistic bisimilarity distances are related to the values. They show that the probabilistic bisimilarity distances for a Markov decision process can be viewed as the optimal value function of an optimal coupling of two copies of the original Markov decision process.

In order to exploit the probabilistic bisimilarity distances, it is essential to be able to approximate or compute these distances. In this dissertation, we will first introduce definitions and theorems from the literature in Chapter 2. We will review the algorithms to compute the probabilistic bisimilarity distances for labelled Markov chains in the literature. We briefly introduce these algorithms below.

The first algorithm to approximate these distances due to Desharnais *et al.* [31] was presented by Van Breugel, Sharma and Worrell in [19]. This algorithm will be discussed in Chapter 3. Since the statement that the distance between states s and t is less than q, for some rational q, can be expressed in the existential fragment of the first order theory over the reals, and this theory is decidable as shown by Tarski [89], one can use binary search to approximate the distance between s and t. The

satisfiability problem for the existential fragment of the first order theory over the reals can be solved in polynomial space [22]. This algorithm can only handle very small examples.

Subsequently, Chen, Van Breugel and Worrell [23] presented a polynomial time algorithm to compute the distances, which will be discussed in Chapter 4. They showed that the distances are rational and that those distances can be computed by means of Khachiyan's ellipsoid method [58]. 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 [70].

In Chapter 6, we will present the algorithm by Bacci, Bacci, Larsen and Mardare [3]. In their paper, they showed that their algorithm, in contrast to the two algorithms mentioned above, can handle non-trivial labelled Markov chains. Their algorithm can be viewed as a basic algorithm, enhanced with an optimization. The key idea behind this optimization is not to compute all the distances but only the ones in which we are interested.

The above three algorithms are the main ones in the literature to approximate or compute probabilistic bisimilarity distances for labelled Markov chains. We will improve and correct some of these algorithms. Also, we will propose and present

some new algorithms.

In Chapter 5, we will construct a transformation mapping each labelled Markov chain to a Markov decision process. A state pair of a labelled Markov chain is mapped to a single vertex of the transformed Markov decision process and the distance of a state pair of the labelled Markov chain corresponds to the value of the corresponding vertex of the transformed Markov decision process. Thus, computing the distances of a labelled Markov chain is equivalent to computing the values of the corresponding Markov decision process. As shown by Van Breugel and Worrell [21], a probabilistic automaton can be transformed into a two player game. A similar transformation will be presented in Chapter 10. Since a labelled Markov chain can be viewed as a probabilistic automaton without nondeterminism, we adopt the game perspective of Markov decision processes in this dissertation, that is, a Markov decision process can be viewed as a one player game.

In Chapter 6, we will show that a small modification of the basic algorithm of Bacci *et al.* [3] can be seen as an instance of simple policy iteration, also known as sequential policy iteration. We will show that the basic algorithm by Bacci *et al.* [3], without the modification, does not always correctly compute the distances. We will prove an exponential lower bound of this simple policy iteration algorithm. Many similar lower bounds have been proved for closely related algorithms by showing that the algorithms can be viewed as binary counters. We refer the reader to, for

example, the thesis of Friedmann [40] for several such proofs. We will also present the general policy iteration algorithm in this chapter. The results in Chapter 5 and Chapter 6 can be found in [85].

In Chapter 7, we will correct the optimization part of the algorithm by Bacci *et al*. We will refer to the algorithm with the corrected optimization as the partial policy iteration. This algorithm has an input which is a query set of state pairs. We are only interested in the distances of the state pairs in this set. Thus, the partial policy iteration algorithm does not need to consider all the state pairs in the system. We will show that an exponential lower bound also holds for the simple partial policy iteration algorithm. We will also generalize the general policy iteration algorithm to use partial policies and present the general partial policy iteration algorithm. These results can be found in [86].

In Chapter 8, we will present three new algorithms. A polynomial time decision procedure for distance one will be presented first. This procedure is the key new ingredient of the three new algorithms to approximate or compute the distances. These new algorithms, as shown in Chapter 9, are much faster and can handle much larger models. The results in this chapter can be found in [87].

To compare the performance of the algorithms to approximate and compute probabilistic bisimilarity distances for labelled Markov chains, we ran several experiments. These algorithms include the algorithm which applies the first order theory

over the reals, the polynomial time algorithm which uses the ellipsoid method, the (partial) policy iteration algorithms due to Bacci *et al.* and our new algorithms. We have implemented the above algorithms in Java and have ran implementations on several labelled Markov chains. 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 [63] and jpf-probabilistic [93]. The experimental results can be found in Chapter 9.

For probabilistic automata, we are not aware of any practical algorithms to compute the probabilistic bisimilarity distances due to Deng *et al.* [29]. In Section 3.2, we will generalize the algorithm to approximate the probabilistic bisimilarity distances for labelled Markov chains which uses the first-order theory over the reals so that it can handle probabilistic automata. This work is very similar to [24, 25]. However, such an algorithm is not practical. In Chapter 10, we will present a transformation mapping each probabilistic automaton to a simple stochastic game, a simplification of Shapley's stochastic games [82] due to Condon [27]. This transformation was firstly presented by Van Breugel and Worrell [21]. We will discuss the possibility of developing a policy iteration algorithm to compute the distances for probabilistic automata. In particular, we will present an alternative characterization of the distances in terms of the corresponding simple stochastic game (Chapter 10) and

a procedure to decide distance one for probabilistic automata (Chapter 11). The results about probabilistic automata can be found in [88].

Chapter 12 will discuss the future work and conclude the dissertation.

1.4 Contributions and Publications

The dissertation explores algorithms to compute probabilistic bisimilarity distances for labelled Markov chains and probabilistic automata. Our major contributions to algorithms to compute the distances for labelled Markov chains are the following.

- We review and generalize the algorithms in literature. In particular, we focus on the policy iteration algorithms. We found an error in the basic algorithm by Bacci *et al.* [3]. We then correct the algorithm and show that the modified algorithm corresponds to a policy iteration algorithm. We prove an exponential lower bound for the simple policy iteration algorithm to compute the distances.

- We study the on-the-fly optimization of the algorithm of Bacci *et al.* and show that it does not always consider sufficiently many states. We modify their optimization and prove our modification correct. We prove an exponential lower bound for the simple partial policy iteration algorithm.

- We present a polynomial time decision procedure for distance one.

- We develop three new algorithms, in which the decision procedure for distance

one is a key step, to compute or approximate the probabilistic bisimilarity distances of labelled Markov chains. These new algorithms are much faster and can handle much larger models in practice.

For probabilistic automata, we provide several key results towards algorithms to compute probabilistic bisimilarity distances for probabilistic automata.

- We present a polynomial time decision procedure for distance one.
- We propose an alternative characterization of their distances in terms of a game. We believe this characterization forms the basis for a policy iteration algorithm to compute the probabilistic bisimilarity distances for probabilistic automata, just as the similar characterization forms the basis for the algorithm to compute the probabilistic bisimilarity distances for labelled Markov chains by Bacci *et al.* [3].

The publications related to the dissertation are the following.

- Qiyi Tang and Franck van Breugel. Deciding Probabilistic Bisimilarity Distance One for Probabilistic Automata. To appear in Proceedings of the 29th International Conference on Concurrency Theory, Beijing, China, September 2018.
- Qiyi Tang and Franck van Breugel. Deciding Probabilistic Bisimilarity Distance One for Labelled Markov Chains. In Proceedings of the 30th Interna-

tional Conference on Computer Aided Verification, Oxford, UK, July 2018.

- Qiyi Tang and Franck van Breugel. Algorithms to Compute Probabilistic Bisimilarity Distances for Labelled Markov Chains. In Proceedings of the 28th International Conference on Concurrency Theory, Berlin, Germany, September 2017.

- Qiyi Tang and Franck van Breugel. Computing Probabilistic Bisimilarity Distances via Policy Iteration. In Proceedings of the 27th International Conference on Concurrency Theory, Quebec City, Canada, August 2016.

2 Probabilistic Models and Probabilistic Bisimilarity Distances

In this chapter, we introduce the two basic probabilistic models, labelled Markov chains and probabilistic automata, that are studied in this dissertation. We present their most prominent behaviour equivalence: probabilistic bisimilarity. We also introduce the quantitative notion of probabilistic bisimilarity distances for these two models. The definitions and results in this chapter are collected from the literature.

2.1 Labelled Markov Chains

In this section, we review the model of interest, labelled Markov chains, its most well known behavioural equivalence, probabilistic bisimilarity due to Larsen and Skou [64] (see also [65]), and the probabilistic bisimilarity pseudometric due to Desharnais *et al.* [31] (see also [33]). Given a finite set S, a function $\mu : S \to [0, 1]$ is a rational probability distribution on S if $\mu(s)$ is rational for every $s \in S$ and $\sum_{s \in S} \mu(s) = 1$. We denote the set of 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 2.1.1. 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* $\ell : S \to L$.

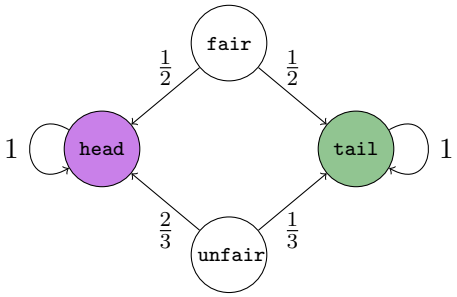
Note that we restrict our attention to labelled Markov chains with finitely many states and the transition probabilities of which are rationals. We denote the transition probability from s to t as $\tau(s)(t)$.

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. These labels are used to capture that particular properties of interest hold in some states and do not hold in other states.

Example 2.1.2. *We consider a labelled Markov chain with four states:* `fair`, `unfair`, `head`, *and* `tail`. *It models a fair coin flip and a biased coin flip. The following table contains the transition probabilities and, hence, captures* τ. *The labelled Markov chain can be depicted as the graph on the right. In this example,*

the label is represented by the colour of the state. The state ***fair*** *models a fair coin flip, which reaches* ***head*** *and* ***tail*** *with probability* $\frac{1}{2}$*, respectively. The state* ***unfair*** *models a biased coin flip, which reaches* ***head*** *with probability* $\frac{2}{3}$ *and* ***tail*** *with probability* $\frac{1}{3}$*.*

	fair	unfair	head	tail
fair	0	0	$\frac{1}{2}$	$\frac{1}{2}$
unfair	0	0	$\frac{2}{3}$	$\frac{1}{3}$
head	0	0	1	0
tail	0	0	0	1



□

For the remainder of this section, we fix a labelled Markov chain $\langle S, L, \tau, \ell \rangle$. In order to characterize probabilistic bisimilarity, we first introduce the notion of a coupling of probability distributions.

Definition 2.1.3. Let μ, $\nu \in \mathrm{Distr}(S)$. The set $\Omega(\mu, \nu)$ of *couplings* of μ and ν is defined by

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

The set $\Omega(\mu, \nu)$ is a convex polytope (see, for example, [72, Section 2.3.2]). We denote its vertices (see, for example, [72, page 36]) 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, [60, page 259]).

Note that $\omega \in \Omega(\mu, \nu)$ is a joint probability distribution with marginals μ and ν (see, for example, [13, page 260-262]).

Definition 2.1.4. The *lifting* of a relation $\mathcal{R} \subseteq S^2$ is the relation $\mathcal{R}\uparrow \subseteq \mathrm{Distr}(S)^2$ defined by $(\mu, \nu) \in \mathcal{R}\uparrow$ if there exists $\omega \in \Omega(\mu, \nu)$ such that $\mathrm{support}(\omega) \subseteq \mathcal{R}$.

This notion of lifting can be found, for example, in [55, Definition 4.3]. It can be used to define probabilistic bisimilarity as follows.

Definition 2.1.5. An equivalence relation $\mathcal{R} \subseteq S^2$ is a *probabilistic bisimulation* if for all s, $t \in S$, if $(s, t) \in \mathcal{R}$ then

- $\ell(s) = \ell(t)$, and
- $(\tau(s), \tau(t)) \in \mathcal{R}\uparrow$.

Probabilistic bisimilarity, denoted $\sim$, is defined as the largest probabilistic bisimulation. For a proof that such a largest probabilistic bisimulation exists, we refer the reader to, for example, [15, Proposition 4.3]. In [55, Theorem 4.6] it is shown that this definition is equivalent to the standard definition given in [64].

Definition 2.1.6. *Probabilistic bisimilarity* is defined by

$$\sim = \bigcup \{\, \mathcal{R} \subseteq S^2 \mid \mathcal{R} \text{ is a probabilistic bisimulation} \,\}.$$

The probabilistic bisimilarity pseudometric of Desharnais *et al.* [31] maps each pair of states of a labelled Markov chain to a distance, which is 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 $[0,1]^{S^2}$. Such a function is a pseudometric if it satisfies the following three properties: for all $s,t,u \in S$, $d(s,s) = 0$, $d(s,t) = d(t,s)$ and $d(s,u) \leq d(s,t) + d(t,u)$. As we will discuss below, it can be defined in terms of the following function.

Definition 2.1.7. The function $\Delta : [0,1]^{S^2} \to [0,1]^{S^2}$ is defined by

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

To define the probabilistic bisimilarity pseudometric as a fixed point of Δ we use the Knaster-Tarski fixed point theorem [90, 61] which is presented next.

Let X be a nonempty set. Consider a function $f : X \to X$. Then $x \in X$ is a fixed point of f if $f(x) = x$. Let $\sqsubseteq \subseteq X^2$. Then $x \in X$ is a pre-fixed point of f if $f(x) \sqsubseteq x$ and it is a post-fixed point of f if $x \sqsubseteq f(x)$. The function f is monotone if for all x, $y \in X$, if $x \sqsubseteq y$ then $f(x) \sqsubseteq f(y)$.

Theorem 2.1.8. [90, Theorem 1] *Let $\langle X, \sqsubseteq \rangle$ be a complete lattice and $f : X \to X$ be a monotone function.*

(a) f has a least fixed point.

(b) f has a greatest fixed point.

(c) The least fixed point of f is the least pre-fixed point of f.

(d) The greatest fixed point of f is the greatest post-fixed point of f.

Although we will not use the definition of complete lattice, interested readers can find it, for example, in [28, Chapter 2]. We denote the least fixed point and the greatest fixed point of f by $\boldsymbol{\mu}(f)$ and $\boldsymbol{\nu}(f)$, respectively.

We present the following theorem which will be used in the proofs of Chapter 11.

Theorem 2.1.9. *Let S be a nonempty finite set and let $\Phi : 2^{S^2} \to 2^{S^2}$ be a monotone function.*

(a) $\boldsymbol{\mu}(\Phi) = \Phi^n(\emptyset)$ for some $n \in \mathbb{N}$.

(b) $\boldsymbol{\nu}(\Phi) = \Phi^n(S^2)$ for some $n \in \mathbb{N}$.

(c) If $X \subseteq \boldsymbol{\mu}(\Phi)$ then $\boldsymbol{\mu}(\Phi) = \Phi^n(X)$ for some $n \in \mathbb{N}$.

Proof. For proofs of part (a) and (b), see, for example, [26, Lemma 8]. Part (c) is proved in the appendix. □

To apply the above theorem, we need to define an order on $[0,1]^{S^2}$.

Definition 2.1.10. The relation $\sqsubseteq \subseteq [0,1]^{S^2} \times [0,1]^{S^2}$ is defined by

$$d \sqsubseteq e \text{ if } d(s,t) \leq e(s,t) \text{ for all } s,t \in S.$$

Proposition 2.1.11. *$\langle [0,1]^{S^2}, \sqsubseteq \rangle$ is a complete lattice.*

Proof. See, for example, [32, Lemma 3.2]. □

Next, we show that $\Omega(\tau(s), \tau(t))$ in Definition 2.1.7 can be replaced by $V(\Omega(\tau(s), \tau(t)))$, and in the future, we use them interchangeably.

Proposition 2.1.12. *For all $d \in [0,1]^{S^2}$ and s, $t \in S$, if $\ell(s) = \ell(t)$ then*

$$\Delta(d)(s,t) = \min_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{u,v \in S} \omega(u,v)\, d(u,v).$$

Proof. Let $d \in [0,1]^{S^2}$ and $s, t \in S$ with $\ell(s) = \ell(t)$. The function $\sigma : \Omega(\tau(s), \tau(t)) \to [0,1]$ is defined by

$$\sigma(\omega) = \sum_{u,v \in S} \omega(u,v)\, d(u,v).$$

For all ω, $\pi \in \Omega(\tau(s), \tau(t))$ and $r \in [0,1]$,

$$\sigma(r\,\omega + (1-r)\,\pi) = r\,\sigma(\omega) + (1-r)\,\sigma(\pi).$$

As a consequence, σ is a concave function (see, for example, [72, page 13]). Since $\Omega(\tau(s), \tau(t))$ is a convex polytope and a concave function on a convex polytope attains its minimum at a vertex of the polytope (see, for example, [60, page 260]), we can conclude that there exists $\omega \in V(\Omega(\tau(s), \tau(t)))$ such that σ attains its minimum at ω. As a consequence, the desired result holds. □

The next proposition proves that Δ is monotone.

Proposition 2.1.13. [16, Proposition 38] *For all d, $e \in [0,1]^{S^2}$, if $d \sqsubseteq e$ then $\Delta(d) \sqsubseteq \Delta(e)$.*

Proof. Let d, $e \in [0,1]^{S^2}$ with $d \sqsubseteq e$. Let s, $t \in S$. We distinguish two cases.

- If $\ell(s) \neq \ell(t)$ then

$$\Delta(d)(s,t) = 1 = \Delta(e)(s,t).$$

- Otherwise,

$$\begin{aligned}
\Delta(d)(s,t) &= \min_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{u,v \in S} \omega(u,v)\, d(u,v) \\
&\leq \min_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{u,v \in S} \omega(u,v)\, e(u,v) \qquad [d \sqsubseteq e] \\
&= \Delta(e)(s,t).
\end{aligned}$$

□

According to Proposition 2.1.11, the set $[0,1]^{S^2}$ endowed with the order $\sqsubseteq$ forms a complete lattice. According to Proposition 2.1.13, Δ is a monotone function. We can conclude from Theorem 2.1.8(a), the Knaster-Tarski fixed point theorem, that Δ has a least fixed point. We denote this fixed point by $\boldsymbol{\mu}(\Delta)$. As we will prove later in this chapter, $\boldsymbol{\mu}(\Delta)$ is a pseudometric. This is the probabilistic bisimilarity pseudometric introduced by Desharnais *et al.*

The probabilistic bisimilarity pseudometric $\boldsymbol{\mu}(\Delta)$ provides a quantitative generalization of probabilistic bisimilarity as captured by the following result which can be found in [31, Theorem 1].

Theorem 2.1.14. [31, Theorem 1] *For all $s, t \in S$, $s \sim t$ if and only if $\boldsymbol{\mu}(\Delta)(s,t) = 0$.*

Proof. We split the proof into two parts.

- ($\Longrightarrow$) For all s, $t \in S$, if $s \sim t$ then $\boldsymbol{\mu}(\Delta)(s,t) = 0$.

  Define $d \in [0,1]^{S^2}$ by

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

  We prove $\boldsymbol{\mu}(\Delta)(s,t) = 0$ for all $s \sim t$ by proving that $\boldsymbol{\mu}(\Delta) \sqsubseteq d$. To do that, we show that d is a pre-fixed point of Δ, that is, $\Delta(d) \sqsubseteq d$. As $\boldsymbol{\mu}(\Delta)$ is the least pre-fixed point of Δ by Theorem 2.1.8(c), we can conclude that $\boldsymbol{\mu}(\Delta) \sqsubseteq d$.

  Let $s, t \in S$. We distinguish two cases.

  - If $s \not\sim t$ then

$$\Delta(d)(s,t) \leq 1 = d(s,t).$$

  - Otherwise, $s \sim t$. According to Definition 2.1.5 and Definition 2.1.6, we can conclude that $\ell(s) = \ell(t)$, and by Definition 2.1.4 there must exist an $\pi \in \Omega(\tau(s), \tau(t))$ such that support$(\pi) \subseteq \sim$.

$$\begin{aligned} \Delta(d)(s,t) &= \min_{\omega \in \Omega(\tau(s),\tau(t))} \sum_{u,v \in S} \omega(u,v)\, d(u,v) \\ &\leq \sum_{u,v \in S} \pi(u,v)\, d(u,v) \\ &= \sum_{(u,v) \in \sim} \pi(u,v)\, d(u,v) \qquad [\text{support}(\pi) \subseteq \sim] \\ &= 0 \qquad [d(u,v) = 0 \text{ for all } u \sim v] \end{aligned}$$

$$= \quad d(s,t). \qquad [s \sim t]$$

- ($\Longleftarrow$) For all s, $t \in S$, if $\boldsymbol{\mu}(\Delta)(s,t) = 0$ then $s \sim t$.

  Let $R \subseteq S^2$ such that $(s,t) \in R$ if and only if $\boldsymbol{\mu}(\Delta)(s,t) = 0$. By Definition 2.1.6, it suffices to show that R is a probabilistic bisimulation.

  Let $(s,t) \in R$. By definition $\boldsymbol{\mu}(\Delta)(s,t) = 0$. Let

  $$\pi = \underset{\omega \in V(\Omega(\tau(s),\tau(t)))}{\operatorname{argmin}} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) \tag{2.1}$$

  According to the definition of Δ, we have $\ell(s) = \ell(t)$ and

  $$\begin{aligned} 0 &= \boldsymbol{\mu}(\Delta)(s,t) \\ &= \Delta(\boldsymbol{\mu}(\Delta))(s,t) \qquad [\boldsymbol{\mu}(\Delta) \text{ is a fixed point of } \Delta] \\ &= \min_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) \\ &= \sum_{u,v \in S} \pi(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) \qquad [2.1]. \end{aligned}$$

  From the above equations, we can deduce that for all $(u,v) \in \text{support}(\pi)$, $\boldsymbol{\mu}(\Delta)(u,v) = 0$, that is, $\text{support}(\pi) \subseteq R$. According to Definition 2.1.5 and Definition 2.1.6, we conclude that R is a probabilistic bisimulation.

□

Example 2.1.15. *The probabilistic bisimilarity distances for the labelled Markov chain in Example 2.1.2 can be found in the table below.*

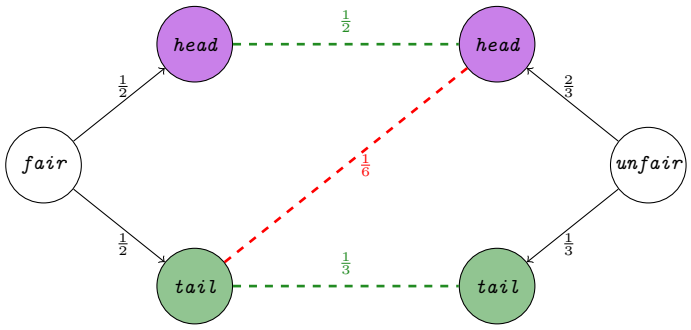
	fair	*unfair*	*head*	*tail*
fair	0	$\frac{1}{6}$	1	1
unfair	$\frac{1}{6}$	0	1	1
head	1	1	0	1
tail	1	1	1	0

The distance for states with different labels is one. For example, the distance between ***head*** *and any other state is* 1*. As the distance function is symmetric, the table above is symmetric. As the distance between a state with itself is zero, the diagonal contains all zeros.*

The states ***fair*** *and* ***unfair*** *have the same label. According to the Definition 2.1.7, we have to find a coupling* $\pi \in \Omega(\tau(\textbf{\textit{fair}}), \tau(\textbf{\textit{unfair}}))$ *such that*

$$\pi = \underset{\omega \in \Omega(\tau(\textbf{\textit{fair}}), \tau(\textbf{\textit{unfair}}))}{\operatorname{argmin}} \sum_{(u,v) \in S^2} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v).$$

Next, we provide a visual representation of a coupling in $\Omega(\tau(\textbf{\textit{fair}}), \tau(\textbf{\textit{unfair}}))$.



A coupling in $\Omega(\tau(\texttt{fair}), \tau(\texttt{unfair}))$ *has to satisfy the following equations.*

$$\begin{aligned}
\omega(\texttt{head}, \texttt{head}) + \omega(\texttt{head}, \texttt{tail}) &= \tfrac{1}{2} \\
\omega(\texttt{tail}, \texttt{head}) + \omega(\texttt{tail}, \texttt{tail}) &= \tfrac{1}{2} \\
\omega(\texttt{head}, \texttt{head}) + \omega(\texttt{tail}, \texttt{head}) &= \tfrac{2}{3} \\
\omega(\texttt{head}, \texttt{tail}) + \omega(\texttt{tail}, \texttt{tail}) &= \tfrac{1}{3}
\end{aligned}$$

Such a coupling can be viewed as a transportation plan of moving one unit from state **fair** *to state* **unfair** *in the network. In the above figure, the dashed lines show a possible coupling.*

*Since the distance between a state and itself is zero, transporting between the tail states contributes zero to the distance of the state pair (***fair***,* **unfair***). Similarly, transporting between the head states contributes zero to the distance of the state pair (***fair***,* **unfair***). Since the distance between* **head** *and* **tail** *is one, transporting between these pairs contributes* $\frac{1}{6}$ *to the distance of the state pair (***fair***,* **unfair***). Hence, the distance of state* **fair** *and state* **unfair** *can be viewed as the cost of transporting one unit from* **fair** *to* **unfair***. As the reader can easily verify, the above transportation plan gives rise to the minimal cost of transporting one unit from state* **fair** *to state* **unfair***, which is* $\frac{1}{6}$*.* □

We partition the set S^2 of state pairs into

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

The following example shows the partition of state pairs of the labelled Markov chain in Example 2.1.2.

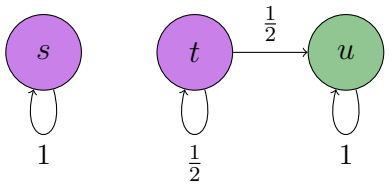
Example 2.1.16. *There are 16 state pairs of states in the labelled Markov chain in Example 2.1.2 and*

$$\begin{aligned} S_0^2 &= \{(\mathit{fair}, \mathit{fair}), (\mathit{unfair}, \mathit{unfair}), (\mathit{head}, \mathit{head}), (\mathit{tail}, \mathit{tail})\} \\ S_1^2 &= \{(\mathit{fair}, \mathit{head}), (\mathit{head}, \mathit{fair}), (\mathit{fair}, \mathit{tail}), (\mathit{tail}, \mathit{fair}), \\ &\quad (\mathit{unfair}, \mathit{head}), (\mathit{head}, \mathit{unfair}), (\mathit{unfair}, \mathit{tail}), (\mathit{tail}, \mathit{unfair}), \\ &\quad (\mathit{head}, \mathit{tail}), (\mathit{tail}, \mathit{head})\} \\ S_?^2 &= \{(\mathit{fair}, \mathit{unfair}), (\mathit{unfair}, \mathit{fair})\}. \end{aligned}$$

□

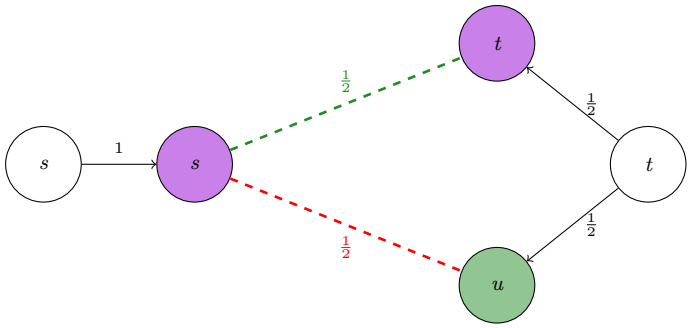
According to Theorem 2.1.14, the state pairs in S_0^2 have distances zero. From the definition of Δ, we can deduce that the state pairs in S_1^2 have distance one. Note that there may be state pairs with the same label that have distance one (see Example 2.1.17). The set $S_?^2$ contains the remaining state pairs.

Example 2.1.17. *We consider a labelled Markov chain with three states: s, t and u. The label is represented by the colour of the state. The transition probabilities are denoted in the graph below. We are interested in the distance of s and t.*



Since s and u have different labels, the distance of s and u is one, that is, $\boldsymbol{\mu}(\Delta)(s,u) = 1$.

State s and state t have the same label. The reader can verify that there is only one coupling π in the set $\Omega(\tau(s),\tau(t))$, where $\pi(s,t) = \pi(s,u) = \frac{1}{2}$. Next, we provide a visual representation of the coupling π.



Recall that a coupling can be viewed as a transportation plan of moving one unit from state s to state t. In the above figure, the dashed lines represent the coupling π. Thus, we have

$$
\begin{aligned}
\boldsymbol{\mu}(\Delta)(s,t) &= \Delta(\boldsymbol{\mu}(\Delta))(s,t) && [\boldsymbol{\mu}(\Delta) \textit{ is a fixed point of } \Delta] \\
&= \min_{\omega \in \Omega(\tau(s),\tau(t))} \sum_{(u,v)\in S^2} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) && [\textit{definition of } \Delta] \\
&= \sum_{(u,v)\in S^2} \pi(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) && [\pi \textit{ is the only coupling in } \Omega(\tau(s),\tau(t))] \\
&= \tfrac{1}{2} \times \boldsymbol{\mu}(\Delta)(s,t) + \tfrac{1}{2} \times \boldsymbol{\mu}(\Delta)(s,u) \\
&= \tfrac{1}{2} \times \boldsymbol{\mu}(\Delta)(s,t) + \tfrac{1}{2} \times 1 && [\boldsymbol{\mu}(\Delta)(s,u) = 1] \\
&= \tfrac{1}{2} \times \boldsymbol{\mu}(\Delta)(s,t) + \tfrac{1}{2}
\end{aligned}
$$

We can conclude that the distance of s and t is one by solving the above equation. This example shows that two states with the same label can have distance one too.

□

The least fixed point of a monotone function on a complete lattice can be obtained iteratively. Starting from the least element of the lattice the function is applied repeatedly. In general, one may have to iterate a transfinite number of times. Here, we restrict our attention to monotone functions on $[0,1]^{S^m}$, where $m \in \mathbb{N}$ and S is a set. We endow the set $[0,1]^{S^m}$ with the following order.

Definition 2.1.18. Let $m \in \mathbb{N}$ and let S be a set. The relation $\sqsubseteq \subseteq [0,1]^{S^m} \times [0,1]^{S^m}$ is defined by

$$f \sqsubseteq g \text{ if } f(s_1, \ldots, s_m) \leq g(s_1, \ldots, s_m) \text{ for all } s_1, \ldots, s_m \in S.$$

Note that the above definition generalizes Definition 2.1.10. Also Proposition 2.1.11 can be generalized as follows.

Proposition 2.1.19. *$\langle [0,1]^{S^m}, \sqsubseteq \rangle$ is a complete lattice.*

Proof. Similar to the proof of Proposition 2.1.11. □

The least element of $[0,1]^{S^m}$ is the function $\mathbf{0} : S^m \to [0,1]$ which maps each tuple to zero. The greatest element of $[0,1]^{S^m}$ is the function $\mathbf{1} : S^m \to [0,1]$ which maps each tuple to one. The set $[0,1]^{S^m}$ not only carries a natural order, as we defined in Definition 2.1.18, but also a natural metric, which we define next.

Definition 2.1.20. The function $\| \cdot - \cdot \| : [0,1]^{S^m} \times [0,1]^{S^m} \to [0,1]$ is defined by

$$\|f - g\| = \sup_{s_1,\ldots,s_m \in S} |f(s_1, \ldots, s_m) - g(s_1, \ldots, s_m)|.$$

A function $\Phi : [0,1]^{S^m} \to [0,1]^{S^m}$ is *c-Lipschitz* if $\|\Phi(f) - \Phi(g)\| \leq c \|f - g\|$ for all f, $g \in [0,1]^{S^m}$. A 1-Lipschitz function is also called *nonexpansive*. A function is *contractive* if it is c-Lipschitz for some $c \in (0,1)$. Now we have all the ingredients to express the iterative characterization of the least fixed point.

Theorem 2.1.21. *Let $m \in \mathbb{N}$ and let S be a finite set. If $\Phi : [0,1]^{S^m} \to [0,1]^{S^m}$ is monotone and nonexpansive then*

$$\boldsymbol{\mu}(\Phi) = \sup_{n \in \mathbb{N}} \Phi^n(\mathbf{0}).$$

Proof. Here we only sketch a proof. The details can be found in [14, Corollary 1].

In the proof, a series of functions $d_n \in [0,1]^{S^m}$ for $n \in \mathbb{N}$ are defined as

$$d_n = \begin{cases} \mathbf{0} & \text{if } n = 0 \\ \Phi(d_{n-1}) & \text{otherwise.} \end{cases}$$

The fact that Φ is nonexpansive implies that Φ is continuous. Since Φ is monotone and continuous, one can show that $d_\omega \sqsubseteq \Phi(d_\omega)$ and $\Phi(d_\omega) \sqsubseteq d_\omega$. As the least fixed point of a monotone function on a complete lattice can be obtained iteratively, we have

$$\boldsymbol{\mu}(\Phi) = d_\omega = \sup_{n \in \omega} d_n = \sup_{n \in \omega} \Phi^n(\mathbf{0}).$$

□

The above theorem will allows us to prove properties of least fixed points by inductive arguments as we will see numerous times.

We present Banach's fixed point theorem [11], which will be used in Chapter 10.

Proposition 2.1.22. $\langle [0,1]^{S^2}, \|\cdot - \cdot\| \rangle$ *is a nonempty complete metric space.*

Proof. See, for example, [9, Section 1.1.2]. □

Theorem 2.1.23. *Let X be a nonempty complete metric space and $f : X \to X$ a contractive function. Then f has a unique fixed point.*

Proof. See, for example, [9, Theorem 1.34]. □

We have already shown that Δ is monotone. According to Theorem 2.1.21, if Δ is nonexpansive, $\boldsymbol{\mu}(\Delta)$ can be characterized as $\sup_{n\in\mathbb{N}} \Delta^n(\mathbf{0})$. We will show that Δ is indeed nonexpansive.

We define the Kantorovich metric [56] in the next definition and will use the fact that the function K is nonexpansive to prove that Δ is nonexpansive.

Definition 2.1.24. The function $K : [0,1]^{X^2} \to [0,1]^{\mathrm{Distr}(X)^2}$ is defined by

$$K(d)(\mu,\nu) = \min_{\omega \in V(\Omega(\nu,\mu))} \sum_{u,v\in S} \omega(u,v)\, d(u,v).$$

The function K is nonexpansive.

Proposition 2.1.25. *For all $d, e \in [0,1]^{X^2}$, $||K(d) - K(e)|| \leq ||d - e||$.*

Proof. See, for example, [14, Section 3]. □

Proposition 2.1.26. *For all* $d, e \in [0,1]^{S^2}$, $||\Delta(d) - \Delta(e)|| \leq ||d - e||$.

Proof. Let $d, e \in [0,1]^{S^2}$. Let $s, t \in S$. We distinguish two cases.

- Let $\ell(s) \neq \ell(t)$. According to the definition of Δ, we have

$$|\Delta(d)(s,t) - \Delta(e)(s,t)| = |1 - 1| = 0 \leq ||d - e||.$$

- If $\ell(s) = \ell(t)$, then

$$\begin{aligned} |\Delta(d)(s,t) - \Delta(e)(s,t)| &= |K(d)(\tau(s),\tau(t)) - K(e)(\tau(s),\tau(t))| \\ &\leq ||K(d) - K(e)|| \\ &\leq ||d - e|| \qquad \text{[Proposition 2.1.25]} \end{aligned}$$

□

Recall that a function $d : S^2 \to [0,1]$ is a pseudometric if for all $s, t, u \in S$:

$$\begin{aligned} d(s,s) &= 0 \\ d(s,t) &= d(t,s) \\ d(s,u) &\leq d(s,t) + d(t,u) \end{aligned}$$

Next, we show that the probabilistic bisimilarity distances form a pseudometric.

Proposition 2.1.27. *If* $d \in [0,1]^{S^2}$ *satisfies* $d(s,t) = d(t,s)$ *for all* $s, t \in S$, *then* $\Delta(d)(s,t) = \Delta(d)(t,s)$ *for all* $s, t \in S$.

Proof. Let $s, t \in S$. We distinguish two cases.

- If $\ell(s) \neq \ell(t)$, then

$$\Delta(d)(s,t) = 1 = \Delta(d)(t,s).$$

- Otherwise, $\ell(s) = \ell(t)$. Towards a contradiction, we assume that

$$\Delta(d)(s,t) > \Delta(d)(t,s) \tag{2.2}$$

without loss of generality. Let $\pi = \operatorname*{argmin}_{\omega \in \Omega(\tau(t),\tau(s))} \sum_{u,v \in S} \omega(u,v)\, d(u,v)$. Let $\rho(u,v) = \pi(v,u)$ for all $u, v \in S$. Since

$$\sum_{v \in S} \rho(u,v) = \sum_{v \in S} \pi(v,u) = \tau(t)(u)$$

and

$$\sum_{u \in S} \rho(u,v) = \sum_{u \in S} \pi(v,u) = \tau(s)(v),$$

we conclude that $\rho \in \Omega(\tau(s), \tau(t))$.

We have

$$\begin{aligned}
\Delta(d)(s,t) &= \min_{\omega \in \Omega(\tau(s),\tau(t))} \sum_{u,v \in S} \omega(u,v)\, d(u,v) \\
&\leq \sum_{u,v \in S} \rho(u,v)\, d(u,v) \\
&= \sum_{u,v \in S} \pi(v,u)\, d(u,v) \qquad \text{[definition of } \rho\text{]} \\
&= \sum_{u,v \in S} \pi(v,u)\, d(v,u) \qquad [d(u,v) = d(v,u) \text{ for all } u, v \in S] \\
&= \min_{\omega \in \Omega(\tau(t),\tau(s))} \sum_{u,v \in S} \omega(u,v)\, d(u,v) \\
&= \Delta(d)(t,s).
\end{aligned}$$

The above result contradicts (2.2). Thus, $\Delta(d)(s,t) = \Delta(d)(t,s)$. $\square$

Proposition 2.1.28. *If $d \in [0,1]^{S^2}$ satisfies $d(s,u) \leq d(s,t) + d(t,u)$ for all $s,t,u \in S$, then $\Delta(d)(s,u) \leq \Delta(d)(s,t) + \Delta(d)(t,u)$ for all $s,t,u \in S$.*

Proof. Let $s,t,u \in S$. We distinguish the following cases.

- If $\ell(s) \neq \ell(u)$, then either $\ell(s) \neq \ell(t)$ or $\ell(t) \neq \ell(u)$. Hence,

$$\Delta(d)(s,u) = 1 \leq \Delta(d)(s,t) + \Delta(d)(t,u).$$

- Otherwise, $\ell(s) = \ell(u)$.

  (a) If $\ell(s) \neq \ell(t)$, then

$$\Delta(d)(s,u) \leq 1 \leq 1 + \Delta(d)(t,u) = \Delta(d)(s,t) + \Delta(d)(t,u).$$

  (b) Otherwise, $\ell(s) = \ell(t)$.

  Let

$$\pi_{s,t} = \operatorname*{argmin}_{\omega \in \Omega(\tau(s),\tau(t))} \sum_{x,y \in S} \omega(x,y)\, d(x,y)$$

  and

$$\pi_{t,u} = \operatorname*{argmin}_{\omega \in \Omega(\tau(t),\tau(u))} \sum_{x,y \in S} \omega(x,y)\, d(x,y).$$

  We define the function $\pi_{s,u} : S^2 \to [0,1]$ by

$$\pi_{s,u}(x,z) = \sum_{y \in \operatorname{support}(\tau(t))} \frac{\pi_{s,t}(x,y)\, \pi_{t,u}(y,z)}{\tau(t)(y)}.$$

First, we show that for all x, $y \in S$,

$$\text{if } y \notin \text{support}(\tau(t)) \text{ then } \pi_{s,t}(x,y) = 0. \tag{2.3}$$

Let $y \in S$ and assume that $y \notin \text{support}(\tau(t))$, that is, $\tau(t)(y) = 0$. Since $\pi_{s,t} \in \Omega(\tau(s), \tau(t))$, we have that $\sum_{x \in S} \pi_{s,t}(x,y) = \tau(t)(y) = 0$. Hence, $\pi_{s,t}(x,y) = 0$ for all $x \in S$. Similarly, one can prove that for all y, $z \in S$,

$$\text{if } y \notin \text{support}(\tau(t)) \text{ then } \pi_{t,u}(y,z) = 0. \tag{2.4}$$

For all x, $z \in S$,

$$\begin{aligned}
\sum_{z \in S} \pi_{s,u}(x,z) &= \sum_{z \in S} \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{s,t}(x,y)\pi_{t,u}(y,z)}{\tau(t)(y)} \\
&= \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{s,t}(x,y)}{\tau(t)(y)} \sum_{z \in S} \pi_{t,u}(y,z) \\
&= \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{s,t}(x,y)}{\tau(t)(y)} \tau(t)(y) \qquad [\pi_{t,u} \in \Omega(\tau(t), \tau(u))] \\
&= \sum_{y \in \text{support}(\tau(t))} \pi_{s,t}(x,y) \\
&= \sum_{y \in S} \pi_{s,t}(x,y) \qquad [(2.3)] \\
&= \tau(s)(x) \qquad [\pi_{s,t} \in \Omega(\tau(s), \tau(t))]
\end{aligned}$$

and

$$\begin{aligned}
\sum_{x \in S} \pi_{s,u}(x,z) &= \sum_{x \in S} \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{s,t}(x,y)\pi_{t,u}(y,z)}{\tau(t)(y)} \\
&= \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{t,u}(y,z)}{\tau(t)(y)} \sum_{x \in S} \pi_{s,t}(x,y)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{t,u}(y,z)}{\tau(t)(y)} \tau(t)(y) \qquad [\pi_{s,t} \in \Omega(\tau(s), \tau(t))] \\
&= \sum_{y \in \text{support}(\tau(t))} \pi_{t,u}(y,z) \\
&= \sum_{y \in S} \pi_{t,u}(y,z) \qquad [(2.4)] \\
&= \tau(u)(z) \qquad [\pi_{t,u} \in \Omega(\tau(t), \tau(u))]
\end{aligned}
$$

Hence, $\pi_{s,u} \in \Omega(\tau(s), \tau(u))$.

Thus, we have

$$
\begin{aligned}
\Delta(d)(s,u) &= \min_{\omega \in \Omega(\tau(s),\tau(u))} \sum_{x,z \in S} \omega(x,z)\, d(x,z) \\
&\leq \sum_{x,z \in S} \pi_{s,u}(x,z)\, d(x,z) \qquad [\pi_{s,u} \in \Omega(\tau(s), \tau(u))] \\
&= \sum_{x,z \in S} \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{s,t}(x,y)\, \pi_{t,u}(y,z)}{\tau(t)(y)}\, d(x,z) \\
&\leq \sum_{x,z \in S} \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{s,t}(x,y)\, \pi_{t,u}(y,z)}{\tau(t)(y)} \big(d(x,y) + d(y,z)\big) \\
&\qquad [d(x,z) \leq d(x,y) + d(y,z) \text{ for all } x,y,z \in S] \\
&= \sum_{x,z \in S} \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{s,t}(x,y)\, \pi_{t,u}(y,z)}{\tau(t)(y)}\, d(x,y) + \\
&\qquad \sum_{x,z \in S} \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{s,t}(x,y)\, \pi_{t,u}(y,z)}{\tau(t)(y)}\, d(y,z) \\
&= \sum_{x \in S} \sum_{z \in S} \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{s,t}(x,y)\, \pi_{t,u}(y,z)}{\tau(t)(y)}\, d(x,y) + \\
&\qquad \sum_{x \in S} \sum_{z \in S} \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{s,t}(x,y)\, \pi_{t,u}(y,z)}{\tau(t)(y)}\, d(y,z) \\
&= \sum_{x \in S} \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{s,t}(x,y)}{\tau(t)(y)}\, d(x,y) \sum_{z \in S} \pi_{t,u}(y,z) +
\end{aligned}
$$

$$
\begin{aligned}
& \sum_{z\in S}\sum_{y\in \mathrm{support}(\tau(t))} \frac{\pi_{t,u}(y,z)}{\tau(t)(y)}\, d(y,z) \sum_{x\in S} \pi_{s,t}(x,y) \\
= \; & \sum_{x\in S}\sum_{y\in \mathrm{support}(\tau(t))} \frac{\pi_{s,t}(x,y)}{\tau(t)(y)}\, d(x,y)\, \tau(t)(y) + \\
& \sum_{z\in S}\sum_{y\in \mathrm{support}(\tau(t))} \frac{\pi_{t,u}(y,z)}{\tau(t)(y)}\, d(y,z)\, \tau(t)(y) \\
& \qquad [\sum_{z\in S} \pi_{t,u}(y,z) = \tau(t)(y) \text{ and } \sum_{x\in S} \pi_{s,t}(x,y) = \tau(t)(y)] \\
= \; & \sum_{x\in S}\sum_{y\in \mathrm{support}(\tau(t))} \pi_{s,t}(x,y)\, d(x,y) + \\
& \sum_{z\in S}\sum_{y\in \mathrm{support}(\tau(t))} \pi_{t,u}(y,z)\, d(y,z) \\
= \; & \sum_{x\in S}\sum_{y\in S} \pi_{s,t}(x,y)\, d(x,y) + \sum_{z\in S}\sum_{y\in S} \pi_{t,u}(y,z)\, d(y,z) \\
& \qquad [(2.3) \text{ and } (2.4)] \\
= \; & \sum_{x,y\in S} \pi_{s,t}(x,y)\, d(x,y) + \sum_{y,z\in S} \pi_{t,u}(y,z)\, d(y,z) \\
= \; & \min_{\omega\in\Omega(\tau(s),\tau(t))} \sum_{x,y\in S} \omega(x,y)\, d(x,y) + \\
& \min_{\omega\in\Omega(\tau(t),\tau(u))} \sum_{y,z\in S} \omega(y,z)\, d(y,z) \\
= \; & \Delta(d)(s,t) + \Delta(d)(t,u).
\end{aligned}
$$

□

Proposition 2.1.29. *For all $d \in [0,1]^{S^2}$, if d is a pseudometric then $\Delta(d)$ is a pseudometric.*

Proof. Let $d \in [0,1]^{S^2}$. Assume that d is a pseudometric. It remains to show that $\Delta(d)$ satisfies the three properties given on page 35.

1. Let the function $\pi : S^2 \to [0,1]$ be defined by

$$\pi(u,v) = \begin{cases} \tau(s)(u) & \text{if } u = v \\ 0 & \text{otherwise} \end{cases}$$

Since

$$\sum_{v \in S} \pi(u,v) = \tau(s)(u)$$

and

$$\sum_{u \in S} \pi(u,v) = \tau(s)(v),$$

we conclude that $\pi \in \Omega(\tau(s), \tau(s))$.

Let $s \in S$. Then

$$\begin{aligned} \Delta(d)(s,s) &= \min_{\omega \in \Omega(\tau(s),\tau(s))} \sum_{u,v \in S} \omega(u,v)\, d(u,v) \\ &\leq \sum_{u,v \in S} \pi(u,v)\, d(u,v) \qquad [\pi \in \Omega(\tau(s),\tau(s))] \\ &= \sum_{u \in S} \pi(u,u)\, d(u,u) \\ &= 0 \qquad [d \text{ is a pseudometric}] \end{aligned}$$

2. Since d is a pseudometric, $d(s,t) = d(t,s)$ for all $s,t \in S$. According to Proposition 2.1.27, $\Delta(d)(s,t) = \Delta(d)(t,s)$ for all $s,t \in S$.

3. Since d is a pseudometric, $d(s,u) \leq d(s,t) + d(t,u)$ for all $s,t,u \in S$. According to Proposition 2.1.28, $\Delta(d)(s,u) \leq \Delta(d)(s,t) + \Delta(d)(t,u)$ for all $s,t,u \in S$.

□

Theorem 2.1.30. $\boldsymbol{\mu}(\Delta)$ *is a pseudometric.*

Proof. First, we show that for all $n \in \mathbb{N}$, $\Delta^n(\mathbf{0})$ is a pseudometric by induction on n. The base case, $n = 0$, is immediate. The induction step follows from Proposition 2.1.29.

It remains to show that $\boldsymbol{\mu}(\Delta)$ satisfies the three properties given on page 35.

- Let $s \in S$. Then

$$\begin{aligned}
\boldsymbol{\mu}(\Delta)(s,s) &= \sup_{n\in\mathbb{N}} \Delta^n(\mathbf{0})(s,s) \\
&\qquad\qquad \text{[Proposition 2.1.13 and 2.1.26 and Theorem 2.1.21]} \\
&= 0 \qquad [\Delta^n(\mathbf{0}) \text{ is a pseudometric}].
\end{aligned}$$

- Let s, $t \in S$. Then

$$\begin{aligned}
\boldsymbol{\mu}(\Delta)(s,t) &= \sup_{n\in\mathbb{N}} \Delta^n(\mathbf{0})(s,t) \\
&\qquad\qquad \text{[Proposition 2.1.13 and 2.1.26 and Theorem 2.1.21]} \\
&= \sup_{n\in\mathbb{N}} \Delta^n(\mathbf{0})(t,s) \qquad [\Delta^n(\mathbf{0}) \text{ is a pseudometric}] \\
&= \boldsymbol{\mu}(\Delta)(t,s) \qquad \text{[Proposition 2.1.13 and 2.1.26 and Theorem 2.1.21]}
\end{aligned}$$

- Let s, t, $u \in S$. Then

$$\boldsymbol{\mu}(\Delta)(s,u) = \sup_{n\in\mathbb{N}} \Delta^n(\mathbf{0})(s,u)$$

[Proposition 2.1.13 and 2.1.26 and Theorem 2.1.21]

$$\begin{aligned}
&\leq \sup_{n\in\mathbb{N}} \left(\Delta^n(\mathbf{0})(s,t) + \Delta^n(\mathbf{0})(t,u)\right) && [\Delta^n(\mathbf{0}) \text{ is a pseudometric}] \\
&= \sup_{n\in\mathbb{N}} \Delta^n(\mathbf{0})(s,t) + \sup_{n\in\mathbb{N}} \Delta^n(\mathbf{0})(t,u) \\
&= \boldsymbol{\mu}(\Delta)(s,t) + \boldsymbol{\mu}(\Delta)(t,u)
\end{aligned}$$

[Proposition 2.1.13 and 2.1.26 and Theorem 2.1.21]

□

We slightly modify the function Δ, defining the probabilistic bisimilarity distances, to the function Δ_1. This new function is a key ingredient of the algorithm based on the ellipsoid method (see Chapter 4), the policy iteration algorithms (see Chapter 6), and the three new algorithms in which deciding distance one is a key step (see Chapter 8).

Definition 2.1.31. The function $\Delta_1 : [0,1]^{S^2} \to [0,1]^{S^2}$ is defined by

$$\Delta_1(d)(s,t) = \begin{cases} 0 & \text{if } s \sim t \\ \Delta(d)(s,t) & \text{otherwise.} \end{cases}$$

Some properties of Δ_1 are collected in the following theorem, which will be used in the correctness proofs of the algorithms presented in Chapter 4, Chapter 6 and Chapter 8.

Theorem 2.1.32.

(a) The function Δ_1 is monotone.

(b) The function Δ_1 is nonexpansive.

(c) $\boldsymbol{\mu}(\Delta_1) = \boldsymbol{\nu}(\Delta_1)$.

(d) $\boldsymbol{\mu}(\Delta_1) = \boldsymbol{\mu}(\Delta)$.

(e) $\boldsymbol{\mu}(\Delta_1) = \sup_{m \in \mathbb{N}} \Delta_1^m(\mathbf{0})$.

Proof.

(a) Since Δ is monotone (Proposition 2.1.13), we can easily deduce that Δ_1 is monotone as well.

(b) Since Δ is nonexpansive (Proposition 2.1.26), we can easily deduce that Δ_1 is nonexpansive as well.

(c) Same as the proof of Theorem 6.2.2(c) where $B = S_1^2$.

(d) Same as the proof of Theorem 6.2.2(d) where $B = S_1^2$.

(e) Since Δ_1 is monotone (part (a)) and nonexpansive (part (b)), we can conclude from Theorem 2.1.21 that $\boldsymbol{\mu}(\Delta_1) = \sup_{n \in \mathbb{N}} \Delta_1^n(\mathbf{0})$.

□

2.2 Probabilistic Automata

In this section, we review the other model of interest, probabilistic automata, its most well known behavioural equivalence, probabilistic bisimilarity due to Segala

and Lynch [79], and the probabilistic bisimilarity pseudometric due to Deng *et al.* [29]. This model was first studied in the context of concurrency by Segala in [78]. It captures both nondeterminism (and, hence, concurrency) and probabilities.

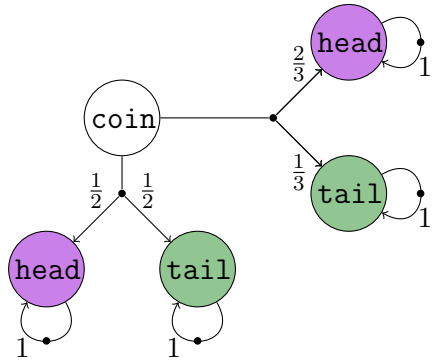
Definition 2.2.1. A *probabilistic automaton* is a tuple $\langle S, L, \to, \ell \rangle$ consisting of

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

For the remainder of this section, we fix a probabilistic automaton $\langle S, L, \to, \ell \rangle$. Instead of $(s, \mu) \in \to$, we will write $s \to \mu$, where $s \in S$ and $\mu \in \mathrm{Distr}(S)$. Note that, we suppose the transition relation $\to$ to be finitely branching, that is, for each $s \in S$ the set $\{\, \mu \in \mathrm{Distr}(S) \mid s \to \mu \,\}$ is *nonempty* and finite.

Example 2.2.2. *In Example 2.1.2, we use the state* `fair` *to model a fair coin flip and the state* `unfair` *to model a biased coin flip. In this example, we model the system which non-deterministically chooses to flip either a fair coin or a biased one using a probabilistic automaton.*

We consider the probabilistic automaton above with five states. Recall that a transition in a labelled Markov chains is probabilistic, that is, it takes a state to



a probability distribution on states. Each state of a probabilistic automaton has a set of such probabilistic transitions emanating from it. The state **`coin`** *has two non-deterministic alternatives. The one takes the automaton to state* **`head`** *and* **`tail`** *with probability* $\frac{1}{2}$*. The other goes to state* **`head`** *with probability* $\frac{2}{3}$ *and state* **`tail`** *with probability* $\frac{1}{3}$*.*

□

Next, we introduce the notion of probabilistic bisimilarity for probabilistic automata due to Segala and Lynch [79].

Definition 2.2.3. An equivalence relation $\mathcal{R} \subseteq S^2$ is a *probabilistic bisimulation* if for all s, $t \in S$, if $(s,t) \in \mathcal{R}$ then

- $\ell(s) = \ell(t)$,
- for all $s \to \mu$ there exists $t \to \nu$ such that $(\mu, \nu) \in \mathcal{R}\uparrow$ and
- for all $t \to \nu$ there exists $s \to \mu$ such that $(\nu, \mu) \in \mathcal{R}\uparrow$.

Definition 2.2.4. *Probabilistic bisimilarity* is defined by

$$\sim = \bigcup \{\, \mathcal{R} \subseteq S^2 \mid \mathcal{R} \text{ is a probabilistic bisimulation} \,\}.$$

Proposition 2.2.5. $\sim$ *is a probabilistic bisimulation.*

Proof. See, for example, [71, Proposition 7.12]. □

As we will see below, the probabilistic bisimilarity pseudometric, due to Deng *et al.* [29] is defined as the least fixed point of the following function.

Definition 2.2.6. The function $\Delta : [0,1]^{S^2} \to [0,1]^{S^2}$ is defined by

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

The following proposition shows that the function Δ is monotone.

Proposition 2.2.7. [29, Lemma 2.10] *For all* d, $e \in [0,1]^{S^2}$, *if* $d \sqsubseteq e$ *then* $\Delta(d) \sqsubseteq \Delta(e)$.

Proof. Let d, $e \in [0,1]^{S^2}$ with $d \sqsubseteq e$. Let s, $t \in S$. We distinguish two cases.

- If $\ell(s) \neq \ell(t)$ then

$$\Delta(d)(s,t) = 1 = \Delta(e)(s,t).$$

- Otherwise,

$$\begin{aligned}
\Delta(d)(s,t) &= \max\Big\{ \max_{s\to\mu}\min_{t\to\nu}\min_{\omega\in\Omega(\mu,\nu)}\sum_{u,v\in S}\omega(u,v)\,d(u,v), \\
&\qquad \max_{t\to\nu}\min_{s\to\mu}\min_{\omega\in\Omega(\mu,\nu)}\sum_{u,v\in S}\omega(u,v)\,d(u,v)\Big\} \\
&\leq \max\Big\{ \max_{s\to\mu}\min_{t\to\nu}\min_{\omega\in\Omega(\mu,\nu)}\sum_{u,v\in S}\omega(u,v)\,e(u,v), \\
&\qquad \max_{t\to\nu}\min_{s\to\mu}\min_{\omega\in\Omega(\mu,\nu)}\sum_{u,v\in S}\omega(u,v)\,e(u,v)\Big\} \\
&\qquad [d \sqsubseteq e] \\
&= \Delta(e)(s,t).
\end{aligned}$$

□

Since $\langle [0,1]^{S^2}, \sqsubseteq\rangle$ is a complete lattice according to Proposition 2.1.11 and Δ is monotone by Proposition 2.2.7, we can conclude from Theorem 2.1.8(a), the Knaster-Tarski fixed point theorem, that Δ has a least fixed point. This least fixed point is denoted by $\boldsymbol{\mu}(\Delta)$ and captures the probabilistic bisimilarity distances for a probabilistic automaton. The fact that these probabilistic bisimilarity distances generalize probabilistic bisimilarity is shown next.

Theorem 2.2.8. [29, Corollary 2.14] *For all s, $t \in S$, $\boldsymbol{\mu}(\Delta)(s,t) = 0$ if and only if $s \sim t$.*

Proof. The proof is very similar to the one of Theorem 2.1.14, so we do not provide the proof here. Instead, we refer the reader to [29, Corollary 2.14]. □

Theorem 2.2.9. $\boldsymbol{\mu}(\Delta)$ *is a pseudometric.*

Proof. The proof is very similar to the one of Theorem 2.1.30, so we do not provide the proof here. □

The Hausdorff metric [47] is defined next.

Definition 2.2.10. The function $H : [0,1]^{X^2} \to [0,1]^{(2^X)^2}$ is defined by

$$H(d)(M,N) = \max\left\{\max_{\mu\in M}\min_{\nu\in N} d(\mu,\nu), \max_{\nu\in N}\min_{\mu\in M} d(\mu,\nu)\right\}.$$

We will show that the function H is nonexpansive and will use this fact to prove that Δ is nonexpansive.

Proposition 2.2.11. *For all* $d, e \in [0,1]^{X^2}, ||H(d) - H(e)|| \leq ||d - e||$.

Proof. Let d, $e \in [0,1]^{X^2}$. It suffices to show that for all M, $N \subseteq X$, $|H(d)(M,N) - H(e)(M,N)| \leq \|d - e\|$. Let M, $N \subseteq X$. Without loss of generality, assume that $H(d)(M,N) \geq H(e)(M,N)$. In this case, it remains to show $H(d)(M,N) \leq H(e)(M,N) + \|d - e\|$. From the definition of H, we can conclude that we need to show

$$\forall \mu \in M : \exists \nu \in N : d(\mu,\nu) \leq H(e)(M,N) + \|d - e\| \wedge \tag{2.5}$$

$$\forall \nu \in N : \exists \mu \in M : d(\mu,\nu) \leq H(e)(M,N) + \|d - e\| \tag{2.6}$$

We only prove (2.5), as (2.6) can be proved similarly. Let $\mu \in M$. From the definition of H, we can conclude that there exists $\nu \in N$ such that $e(\mu,\nu) \leq H(e)(M,N)$.

Since $d(\mu,\nu) - e(\mu,\nu) \leq \|d-e\|$, we can conclude that

$$d(\mu,\nu) \leq e(\mu,\nu) + \|d-e\| \leq H(e)(M,N) + \|d-e\|.$$

□

Proposition 2.2.12. *For all* $d, e \in [0,1]^{S^2}$, $||\Delta(d) - \Delta(e)|| \leq ||d-e||$.

Proof. Let $d, e \in [0,1]^{S^2}$. Let $s,t \in S$. We distinguish two cases.

- Let $\ell(s) \neq \ell(t)$. By the definition of Δ, we have

$$|\Delta(d)(s,t) - \Delta(e)(s,t)| = |1-1| = 0 \leq ||d-e||.$$

- Otherwise, $\ell(s) = \ell(t)$.

$$\begin{aligned}
&|\Delta(d)(s,t) - \Delta(e)(s,t)| \\
&= \ | \ H(K(d))(\{\mu \mid s \to \mu\}, \{\nu \mid t \to \nu\}) - \\
&\qquad H(K(d)) \ H(K(e))(\{\mu \mid s \to \mu\}, \{\nu \mid t \to \nu\})| \\
&\leq \ \|H(K(d)) - H(K(e))\| \\
&\leq \ \|K(d) - K(e)\| \qquad \text{[Proposition 2.2.11]} \\
&\leq \ \|d-e\| \qquad \text{[Proposition 2.1.25]}
\end{aligned}$$

□

Similarly to the discussion of labelled Markov chains, we partition the set S^2 of state pairs of a probabilistic automaton into

- $S_0^2 = \{(s,t) \in S^2 \mid s \sim t\}$
- $S_1^2 = \{(s,t) \in S^2 \mid \ell(s) \neq \ell(t)\}$
- $S_?^2 = S^2 \setminus (S_0^2 \cup S_1^2)$.

We slightly modify the function Δ defining the probabilistic bisimilarity distances to the function Δ_1.

Definition 2.2.13. The function $\Delta_1 : [0,1]^{S^2} \to [0,1]^{S^2}$ is defined as follows.

$$\Delta_1(d)(s,t) = \begin{cases} 0 & \text{if } (s,t) \in S_0^2 \\ \Delta(d)(s,t) & \text{otherwise.} \end{cases}$$

We collect some of the properties of Δ_1 in the following theorem, which will be used later.

Theorem 2.2.14.

(a) The function Δ_1 is monotone.

(b) $\boldsymbol{\mu}(\Delta_1) = \boldsymbol{\mu}(\Delta)$.

Proof.

(a) Since Δ is monotone (Proposition 2.2.7), we can easily deduce that Δ_1 is monotone as well.

(b) Similar to [23, Corollary 18].

□

3 First Order Theory over the Reals

In this chapter, we look at the algorithm to approximate the probabilistic bisimilarity distances for probabilistic transition systems by Van Breugel, Sharma and Worrell [18] (see also [19]). The key idea is to express the distance function using the first order theory over the reals. We adapt this key idea to come up with algorithms to approximate probabilistic bisimilarity distances for labelled Markov chains and probabilistic automata.

3.1 Labelled Markov Chains

In [18], Van Breugel, Sharma and Worrell show that the probabilistic bisimilarity distances for probabilistic transition systems, a variant of Markov chains, can be defined as a greatest fixed point of a function. The probabilistic bisimilarity distances are then a post-fixed point of this function, which can be expressed in the first order theory over real closed fields. Based on that, they come up with an algorithm which exploits Tarski's decision procedure [89] to approximate the probabilistic bisimilarity distances. We describe this algorithm in this section, adapted to the case of labelled

Markov chains.

A probabilistic transition system differs from a labelled Markov chains in 1) there are no labels in a probabilistic transition system, while the states are labelled in a labelled Markov chain; 2) a state in a probabilistic transition system may have no outgoing transitions, while the outgoing transitions of a state of a labelled Markov chain form a probability distribution, that is, for any state s we have $\sum_{t \in S} \tau(s)(t) = 1$.

It is noted that the probabilistic bisimilarity distances in [18] are defined as a greatest fixed point of a function, while in Section 2.1 we define these distances as a least fixed point of Δ. This is mainly due to the fact that their ordering [18, Definition 8] is the opposite to ours in Definition 2.1.10.

The probabilistic bisimilarity distances on the labelled Markov chain are defined as $\boldsymbol{\mu}(\Delta)$, the least fixed point of Δ of Definition 2.1.7. According to Theorem 2.1.8(c), $\boldsymbol{\mu}(\Delta)$ is a pre-fixed point of Δ.

For the rest of this section, we assume that the labelled Markov chain $\langle S, L, \tau, \ell \rangle$ has N states $s_1, s_2, \cdots, s_N$. We represent the probabilistic bisimilarity distances on the set S of states of the labelled Markov chain as a collection of real valued variables d_{ij} where $1 \leq i, j \leq N$.

We use the predicate below to capture that d is a pseudometric. It requires that the distance must be a real value in $[0, 1]$, and the three properties which we mentioned in Section 2.1.

Definition 3.1.1. The predicate pseudo(d) is defined by

$$\begin{aligned}\text{pseudo}(d) \quad \equiv \quad & \bigwedge_{1\leq i,j\leq N} \big(d_{ij} \leq 1 \wedge d_{ij} \geq 0\big) \wedge \bigwedge_{1\leq i\leq N} d_{ii} = 0 \wedge \\ & \bigwedge_{1\leq i,j\leq N} d_{ij} = d_{ji} \wedge \bigwedge_{1\leq i,j,k\leq N} d_{ik} \leq d_{ij} + d_{jk}\end{aligned}$$

Since we have shown in Theorem 2.1.30 that $\boldsymbol{\mu}(\Delta)$ is a pseudometric even though the function Δ is not restricted to pseudometrics, we can replace Definition 3.1.1 with the definition below.

Definition 3.1.2. The predicate unit(d) is defined by

$$\text{unit}(d) \equiv \bigwedge_{1\leq i,j\leq N} d_{ij} \leq 1 \wedge d_{ij} \geq 0 \tag{3.1}$$

The following two predicates help define the predicate pre-fixed(d). pre-fixed$_1(d,i,j)$ captures the case where s_i and s_j have different labels and, hence, their distance is one. pre-fixed$_2(d,i,j)$ captures the case where s_i and s_j have the same label and in which case $\Delta(d)(s_i,s_j) \leq d_{ij}$ captures the pre-fixed point property. The latter is the case if there exists $\omega \in \Omega(\tau(s_i),\tau(s_j))$ such that $\sum_{1\leq x,y\leq N} \omega_{xy}\, d_{xy} \leq d_{ij}$. The predicate pre-fixed(d) below is then used to capture that d is a pre-fixed point of Δ.

Definition 3.1.3. The predicate pre-fixed(d) is defined by

$$\text{pre-fixed}(d) \equiv \bigwedge_{1\leq i,j\leq N} \big(\text{pre-fixed}_1(d,i,j) \vee \text{pre-fixed}_2(d,i,j)\big)$$

where

$$\text{pre-fixed}_1(d,i,j) \;\equiv\; \ell(s_i) \neq \ell(s_j) \wedge d_{ij} = 1$$

and

$$\begin{aligned}
\text{pre-fixed}_2(d,i,j) \;\equiv\;& \ell(s_i) = \ell(s_j) \wedge \\
& \exists (\omega_{xy})_{1\leq x,y\leq N} \Bigg(\Big(\bigwedge_{1\leq x,y\leq N} \omega_{xy} \leq 1 \wedge \omega_{xy} \geq 0 \Big) \wedge \\
& \Big(\bigwedge_{1\leq x\leq N} \sum_{1\leq y\leq N} \omega_{xy} = \tau(s_i)(s_x) \Big) \wedge \\
& \Big(\bigwedge_{1\leq y\leq N} \sum_{1\leq x\leq N} \omega_{xy} = \tau(s_j)(s_y) \Big) \wedge \\
& \Big(\sum_{1\leq x,y\leq N} \omega_{xy}\, d_{xy} \leq d_{ij} \Big) \Bigg)
\end{aligned}$$

With the definitions above, we are ready to present the algorithm. The inputs of the algorithm include a labelled Markov chain $\langle S, L, \tau, \ell \rangle$, a positive threshold $\epsilon \in (0, 1]$, and a pair of states (s_i, s_j) for which the distance will be approximated. The algorithm outputs an interval of at most size ϵ in which the probabilistic bisimilarity distance for (s_i, s_j) lies.

`tarski` is a decision procedure which takes as input a predicate expressed in the first order theory over the reals. This procedure outputs whether there is solution to the input predicate. The existence of such a procedure was first proved by Tarski [89]. The upper-bound of the time complexity of this decision procedure is doubly exponential in the number of quantifier alternations, and exponential in the number of variables [74].

According to Definition 2.1.7, a pair of states with different labels has distance one. This leads to the algorithm directly returning the interval $[1, 1]$, if the states s_i and s_j have different labels. Otherwise, we use binary search to find the approximation interval.

```
1  if  ℓ(s_i) ≠ ℓ(s_j)
2     return  [1, 1];
3  else
4     l = 0
5     u = 1
6     while  u − l > ϵ
7        m = (l+u)/2
8        if  tarski(∃ d ∈ ℝ^{N²} :  unit(d) ∧ pre-fixed(d) ∧ d_ij ≤ m  )
9           u = m;
10       else
11          l = m;
12    return  [l, u];
```

The next proposition indicates that if there is a solution to the input predicate of the procedure `tarski` on line 8 for a distance function d, a state pair (s_i, s_j) and a real number m , then the distance between s_i and s_j must be less than or equal to m. Otherwise, the distance between s_i and s_j is greater than m.

Proposition 3.1.4. $\exists d \in [0,1]^{S^2} : \Delta(d) \sqsubseteq d \wedge d(s_i, s_j) \leq m \iff \boldsymbol{\mu}(\Delta)(s_i, s_j) \leq m$.

Proof. We prove two implications.

- Assume

$$\exists d \in [0,1]^{S^2} : \Delta(d) \sqsubseteq d \wedge d(s_i, s_j) \leq m. \tag{3.2}$$

  According to Theorem 2.1.8(c), $\boldsymbol{\mu}(\Delta)$ is the least pre-fixed point of Δ. Thus, $\boldsymbol{\mu}(\Delta) \sqsubseteq e$ for all $e \in [0,1]^{S^2}$ with $\Delta(e) \sqsubseteq e$. By (3.2), we have $\boldsymbol{\mu}(\Delta) \sqsubseteq d$ and, hence, $\boldsymbol{\mu}(\Delta)(s_i, s_j) \leq d(s_i, s_j) \leq m$.

- Assume $\boldsymbol{\mu}(\Delta)(s_i, s_j) \leq m$. Since $\boldsymbol{\mu}(\Delta)$ is a pre-fixed point of Δ, we have $\Delta(\boldsymbol{\mu}(\Delta)) \sqsubseteq \boldsymbol{\mu}(\Delta) \wedge \boldsymbol{\mu}(\Delta)(s_i, s_j) \leq m$.

□

To prove the partial correctness of the above algorithm, we annotate it with the following assertions.

```
1 if  ℓ(s_i) ≠ ℓ(s_j)
2     return  [1,1];
3 else
4     l = 0
5     u = 1
6     while  u − l > ϵ
```

$\{\boldsymbol{\mu}(\Delta)(s_i, s_j) \in [l, u]\}$

7 $m = \frac{l+u}{2}$

8 if `tarski` $(\exists d \in \mathbb{R}^{N^2} : \text{unit}(d) \wedge \text{pre-fixed}(d) \wedge d_{ij} \leq m \ )$

$\{\boldsymbol{\mu}(\Delta)(s_i, s_j) \in [l, u] \wedge \boldsymbol{\mu}(\Delta)(s_i, s_j) \leq m\}$

9 $u = m$;

$\{\boldsymbol{\mu}(\Delta)(s_i, s_j) \in [l, u]\}$

10 else

$\{\boldsymbol{\mu}(\Delta)(s_i, s_j) \in [l, u] \wedge \boldsymbol{\mu}(\Delta)(s_i, s_j) > m\}$

11 $l = m$;

$\{\boldsymbol{\mu}(\Delta)(s_i, s_j) \in [l, u]\}$

12 return $[l, u]$;

To conclude that the above iterative procedure terminates, we observe that in each iteration the size of the interval $[l, u]$ is cut in half. Thus, the algorithm terminates in $\log_{\frac{1}{2}} \epsilon$ iterations.

3.2 Probabilistic Automata

In this section, we apply the first order theory over the reals to approximate the probabilistic bisimilarity distances for probabilistic automata. Similar to labelled Markov chains, the probabilistic bisimilarity distances for probabilistic automata are defined as $\boldsymbol{\mu}(\Delta)$, which is a pre-fixed point of Δ. As it can also be expressed in

the first order theory over the reals, we can come up with a similar algorithm for probabilistic automata. Some similar work can be found in [24, 25].

For the rest of this section, we assume that the probabilistic automaton $\langle S, L, \to, \ell \rangle$ has N states $s_1, s_2, \cdots, s_N$. We represent the probabilistic bisimilarity distances on the set S of states of the probabilistic automaton as a collection of real-valued variables d_{ij} for $1 \leq i, j \leq N$. We assume each state s_i, where $1 \leq i \leq N$, has M_i non-deterministic transitions to $\mu_1^i, \mu_2^i, \cdots, \mu_{M_i}^i$.

We use the same predicate unit(d) defined in Definition 3.1.2 to capture that each d_{ij} is in the unit interval $[0, 1]$. The following two predicates help define the predicate pre-fixed(d). pre-fixed$_1(d, i, j)$ is the same as in Definition 3.1.3 which captures the case where s_i and s_j have different labels and, hence, the distance is one. pre-fixed$_2(d, i, j)$ captures the case where s_i and s_j have the same label and in which case $\Delta(d)(s_i, s_j) \leq d_{ij}$ captures the pre-fixed point property. The latter is the case if for each $s_i \to \mu_u^i$ there exists $s_j \to \mu_v^j$ and a coupling $\omega \in \Omega(\mu_u^i, \mu_v^j)$ such that $\sum_{1 \leq x, y \leq N} \omega_{xy} d_{xy} \leq d_{ij}$ and the same condition with the roles of s_i and s_j interchanged. The predicate pre-fixed(d) below is then used to capture that d is a pre-fixed point of Δ.

Definition 3.2.1. The predicate pre-fixed(d) is defined by

$$\text{pre-fixed}(d) \equiv \bigwedge_{1 \leq i, j \leq N} \text{pre-fixed}_1(d, i, j) \vee \text{pre-fixed}_2(d, i, j)$$

where

$$\text{pre-fixed}_1(d,i,j) \;\equiv\; \ell(s_i) \neq \ell(s_j) \wedge d_{ij} = 1$$

and

$$\begin{aligned}
\text{pre-fixed}_2(d,i,j) \;\equiv\; & \ell(s_i) = \ell(s_j) \wedge \\
& \bigwedge_{1\leq u\leq M_i} \bigvee_{1\leq v\leq M_j} \exists(\omega_{xy})_{1\leq x,y\leq N} \\
& \Bigg(\Big(\bigwedge_{1\leq x,y\leq N} \omega_{xy} \leq 1 \wedge \omega_{xy} \geq 0 \Big) \wedge \\
& \Big(\bigwedge_{1\leq x\leq N} \sum_{1\leq y\leq N} \omega_{xy} = \mu_u^i(s_x) \Big) \wedge \\
& \Big(\bigwedge_{1\leq y\leq N} \sum_{1\leq x\leq N} \omega_{xy} = \mu_v^j(s_y) \Big) \wedge \\
& \sum_{1\leq x,y\leq N} \omega_{xy}\, d_{xy} \leq d_{ij} \Bigg)
\end{aligned}$$

With the above definitions, we are ready to describe the algorithm. The inputs of the algorithm include a probabilistic automaton $\langle S, L, \to, \ell\rangle$, a positive threshold $\epsilon \in (0, 1]$, and a pair of states (s_i, s_j) for which the distance will be approximated. The algorithm outputs an interval of at most size ϵ in which the probabilistic bisimilarity distance for (s_i, s_j) lies.

According to Definition 2.2.6, a pair of states with different labels has distance one. This leads to the algorithm directly returning the interval $[1, 1]$, if the states s_i and s_j have different labels. Otherwise, we use binary search to find the approximated interval. To avoid redundancy, we do not list the algorithm here as it is the same

as the one for labelled Markov chains. The only difference is the definition of the predicate pre-fixed$(d)_2$ which is used as the input predicate to the decision procedure `tarski`.

4 Ellipsoid Method

In [23], Chen, Van Breugel and Worrell showed that probabilistic bisimilarity distances for a labelled Markov chain can be expressed as the solution of a linear program. This linear program can be solved by Khachiyan's ellipsoid method [58], of which the running time is polynomial in the size of the labelled Markov chain. In this chapter, we give a brief introduction on how to use binary search together with the ellipsoid method to solve linear programs. We then use this algorithm to approximate the probabilistic bisimilarity distances for labelled Markov chains. At the end of this chapter, we discuss the possibility of applying this technique to compute the probabilistic bisimilarity distances for probabilistic automata.

4.1 Linear Programming and the Ellipsoid Method

In this section, we review some definitions related to linear programming and give an introduction of the ellipsoid method.

Definition 4.1.1. Let m, n be positive integers. A *linear program* in inequality

form is of the following form:

$$\text{maximize } \boldsymbol{c^T} \cdot \boldsymbol{x} \text{ such that}$$

$$\boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}$$

where $\boldsymbol{x}, \boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$.

The linear program in $\mathbb{R}^n$ defined above is said to have m constraints which are expressed by $\boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}$. Note that for a linear program, the constraints are not necessarily given explicitly. The linear program in Section 4.2, which corresponds to computing the probabilistic bisimilarity distances, does not give the constraints explicitly. $\boldsymbol{c^T} \cdot \boldsymbol{x}$ is called the objective function. For the remainder of this section, we fix a linear program **LP** as defined above.

For a linear program, there are several possibilities:

- the linear program is infeasible, that is, no vector $\boldsymbol{x}$ satisfying the constraints exists;
- the maximized value of $\boldsymbol{c^T} \cdot \boldsymbol{x}$ is unbounded;
- an optimum exists.

We assume that the linear program **LP** is feasible and bounded and, hence, has an optimal value.

Let c be a positive integer. The ellipsoid method can be used to check whether there exists a vector $\boldsymbol{x}$ that satisfies the following system. Such a point is called a

feasible point.

$$\boldsymbol{c^T} \cdot \boldsymbol{x} \geq c$$

$$\boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}$$

The ellipsoid method determines the feasibility of the above system in polynomial time [58]. We denote the procedure as `ellipsoid`($\mathbf{LP}, c$). This procedure returns `true` if the system has a feasible point. It implies that the optimum of **LP** is at least c. It returns `false` if the system is infeasible which implies that the optimum of **LP** is less than c.

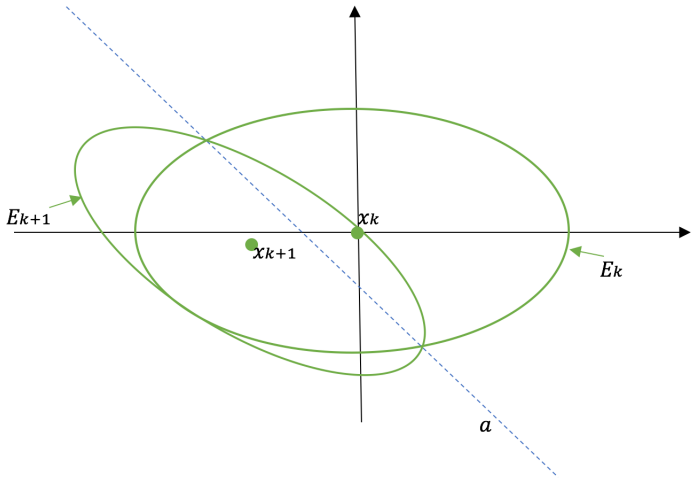
There are several ways to use the ellipsoid method to solve a linear program (see, for example, [75, Section 3.1]). Here we present the one which can solve **LP** by combining binary search and the ellipsoid method. Let l and u be the lower bound and the upper-bound of $\boldsymbol{c^T} \cdot \boldsymbol{x}$, respectively. Let ϵ be a positive number. The following algorithm outputs an interval of at most size ϵ in which the optimum of **LP** lies.

```
1 while  u − l > ε
2    c = (l+u)/2
3    if  ellipsoid(LP, c)
4       l = c;
5    else
6       u = c;
```

7 return $[l, u]$;

In the above algorithm, if line 3 returns `true`, it means the optimum is greater than c, so that the optimum lies in $[c, u]$. Otherwise, line 3 returns `false` and the optimum lies in $[l, c)$. We continue the algorithm until we reach a solution with the desired accuracy.

Now that we know how to use the ellipsoid method to solve a linear problem, we are ready to introduce the ellipsoid method at a high level. For proofs of the correctness of the ellipsoid method, we refer the interested reader to, for example, [77, Chapter 13-14].



The ellipsoid method proceeds in iterations. It starts with an initial ellipsoid of which the volume is big enough to include all the feasible points of **LP** with bound c. The centre of the ellipsoid at each iteration is a candidate for a feasible point of the problem. At each iteration, it verifies whether the centre of the ellipsoid

is a feasible point. If it is, the algorithm terminates and returns `true`. If it is not, the algorithm calls the **Separation** procedure find a vector. This vector is used to construct a new ellipsoid of which the centre might be feasible for the next iteration. If a maximum number of iterations is reached and a feasible point is not found, the algorithm terminates and returns `false`.

The above figure shows the kth iteration of the ellipsoid method where k is smaller than the maximum number of iterations. At the kth iteration, the ellipsoid is E_k. If its centre $\boldsymbol{x_k}$ is not a feasible point, we find a vector $\boldsymbol{a}$. This vector forms a hyperplane $\{\boldsymbol{x} \mid \boldsymbol{a^T} \cdot \boldsymbol{x} < \boldsymbol{a^T} \cdot \boldsymbol{x_k}\}$ which separates $\boldsymbol{x_k}$ and the region defined on line 9 of the algorithm below. The separating hyperplane is then used to construct the new ellipsoid E_{k+1} with centre $\boldsymbol{x_{k+1}}$ which includes the portion of E_k that lies on the opposite side of the hyperplane from $\boldsymbol{x_k}$ for the next iteration.

In the algorithm below, the initial ellipsoid $E_0 = R^2\boldsymbol{I_n}$ is a ball with centre $\mathbf{0}$ and radius R. Note that R should be initialized properly to make sure that the ellipsoid includes all the feasible solutions. We will come back to how to initialise R later. N is the maximum number of iterations the algorithm can have. Its value depends on the size of the linear program. For details of what value should be assigned to N, we refer the readers to [77, page 168] for the linear programs with explicit inequalities and [77, page 173] for the linear programs without explicit inequalities.

The ellipsoid method `ellipsoid`($\mathbf{LP}, c, \mathbf{Separation}$) is presented below.

1 k = 0;

2 $\boldsymbol{x_0} = \mathbf{0}$;

3 $\boldsymbol{B_0} = R^2\boldsymbol{I_n}$ where $\boldsymbol{I_n}$ is the $n \times n$ identity matrix;

4 while $k \leq N$

5 if $\boldsymbol{c^T} \cdot \boldsymbol{x_k} \geq c$ and $\boldsymbol{x_k}$ satisfies the constraints of **LP**

6 return `true`;

7 else

8 **Separation** outputs $\boldsymbol{a} \in \mathbb{R}^n$ such that $\boldsymbol{a^T} \cdot \boldsymbol{x} < \boldsymbol{a^T} \cdot \boldsymbol{x_k}$

9 for all $\boldsymbol{x}$ satisfying the constraints of **LP** and $\boldsymbol{c^T} \cdot \boldsymbol{x} \geq c$;

10 $\boldsymbol{d} = \frac{1}{\sqrt{\boldsymbol{a}^T \boldsymbol{B_k} \boldsymbol{a}}} \boldsymbol{B_k} \boldsymbol{a}$;

11 $\boldsymbol{x_{k+1}} = \boldsymbol{x_k} - \frac{1}{n+1}\boldsymbol{d}$;

12 $\boldsymbol{B_{k+1}} = \frac{n^2}{n^2-1}(\boldsymbol{B_k} - \frac{2}{n+1}\boldsymbol{d}\boldsymbol{d^T})$;

13 k = k+1;

14 return `false`;

If after N iterations, a feasible point is not found, the algorithm returns `false` at line 14. Otherwise, at the kth iteration, the ellipsoid $E_k = \{\boldsymbol{x} \in \mathbb{R}^n \mid (\boldsymbol{x} - \boldsymbol{x_k})^T \boldsymbol{B_k^{-1}} (\boldsymbol{x} - \boldsymbol{x_k})\}$ is defined by the $n \times n$ matrix $\boldsymbol{B_k}$. The ellipsoid has centre $\boldsymbol{x_k}$, which is a potential feasible point and this is checked at line 5. If $\boldsymbol{x_k}$ is a feasible point, the algorithm terminates with `true`. If it is not, we need to find a separating hyperplane to construct a new ellipsoid for the $(k+1)$th iteration. The construction

of the new ellipsoid is shown on line 8-12. The procedure which checks whether $\boldsymbol{x_k}$ is a feasible point or not and in the latter case finds a separating hyperplane is called a separation algorithm. Note that a separation procedure **Separation** running in polynomial time is a necessary condition for the ellipsoid method being polynomial time as well.

4.2 Labelled Markov Chains

In this section, we review the polynomial time algorithm to compute the distances presented by Chen, Van Breugel and Worrell [23]. They showed that the distances are rational [23, page 446] and that those distances can be computed by means of Khachiyan's ellipsoid method [58]. 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 [70].

According to Theorem 2.1.32(c, d), for a labelled Markov chain $\langle S, L, \tau, \ell \rangle$, the probabilistic bisimilarity pseudometric $\boldsymbol{\mu}(\Delta)$ is the unique fixed point of Δ_1. It follows that $\boldsymbol{\mu}(\Delta)$ is the greatest fixed point of Δ_1. By Theorem 2.1.8(d), $\boldsymbol{\mu}(\Delta)$ is the greatest post-fixed point of Δ_1. Thus, the following linear program computes the probabilistic bisimilarity distances. Note that the constraints of this linear program

are not given explicitly, for the case where a state pair has the same label but is not probabilistic bisimilar.

$$\text{maximize} \sum_{(s,t)\in S^2} d(s,t) \text{ such that}$$

$$\begin{aligned}
d(s,t) &\geq 0 \\
d(s,t) &\leq 1 \\
d(s,t) &\leq 0 && s \sim t \\
d(s,t) &\leq 1 && \ell(s) \neq \ell(t) \\
d(s,t) &\leq \sum_{(u,v)\in S^2} \omega(u,v)\, d(u,v) && \omega \in V(\Omega(\tau(s),\tau(t))), s \not\sim t, \ell(s) = \ell(t)
\end{aligned}$$

Recall that $S^2_?$, S^2_1 and S^2_0 form a partition of S^2. S^2_1 contains all the pairs of states that have different labels. Thus, S^2_1 can be decided by comparing the labels of every pair of states, of which the running time is polynomial in the number of states. S^2_0 contains the state pairs which have distance zero. According to Theorem 2.1.14, distance zero coincides with probabilistic bisimilarity. The first decision procedure for probabilistic bisimilarity was provided by Baier [7]. More efficient decision procedures were subsequently proposed by Derisavi, Hermanns and Sanders [30] and also by Valmari and Franceschinis [92]. The latter two both run in $O(|E| \log |S|)$ time, where $|S|$ and $|E|$ are the number of states and transitions of the labelled Markov chain. Hence, it remains to compute the probabilistic bisimilarity distances of the state pairs in $S^2_?$.

The above system is equivalent to the system below. In this section, we fix **LP**

to be the following linear program.

$$\text{maximize} \sum_{(s,t)\in S^2_?} d(s,t) \text{ such that}$$

$$\begin{array}{lll}
\forall (s,t) \in S^2_? & d(s,t) & \geq 0 \\
\forall (s,t) \in S^2_? & d(s,t) & \leq 1 \\
\forall (s,t) \in S^2_? \; \forall \omega \in V(\Omega(\tau(s),\tau(t))) & d(s,t) & \leq \sum_{(u,v)\in S^2_?} \omega(u,v)\, d(u,v) + \\
 & & \sum_{(u,v)\in S^2_1} \omega(u,v)
\end{array}$$

We can restrict the linear program to the set $S^2_?$ since $d(s,t)$ is zero for $(s,t) \in S^2_0$ and is one for $(s,t) \in S^2_1$. Note that for each $(s,t) \in S^2_?$, instead of $d(s,t) \leq \sum_{(u,v)\in S^2} \omega(u,v)\, d(u,v)$, we have $d(s,t) \leq \sum_{(u,v)\in S^2_?} \omega(u,v)\, d(u,v) + \sum_{(u,v)\in S^2_1} \omega(u,v)$, since $d(u,v)$ can be replaced with zero if $(u,v) \in S^2_0$ and one if $(u,v) \in S^2_1$. That is,

$$\begin{aligned}
& \sum_{(u,v)\in S^2} \omega(u,v)\, d(u,v) \\
= & \sum_{(u,v)\in S^2_?} \omega(u,v)\, d(u,v) + \sum_{(u,v)\in S^2_0} \omega(u,v)\, d(u,v) + \sum_{(u,v)\in S^2_1} \omega(u,v)\, d(u,v) \\
= & \sum_{(u,v)\in S^2_?} \omega(u,v)\, d(u,v) + \sum_{(u,v)\in S^2_0} \omega(u,v) \times 0 + \sum_{(u,v)\in S^2_1} \omega(u,v) \times 1 \\
= & \sum_{(u,v)\in S^2_?} \omega(u,v)\, d(u,v) + \sum_{(u,v)\in S^2_1} \omega(u,v)
\end{aligned}$$

Thus, for each $(s,t) \in S^2_?$ and $\omega \in V(\Omega(\tau(s),\tau(t)))$, we have $d(s,t) \leq \sum_{(u,v)\in S^2_?} \omega(u,v)\, d(u,v) + \sum_{(u,v)\in S^2_1} \omega(u,v)$.

The dimension n of this linear program is $|S^2_?|$. As discussed in the previous section, we can use binary search to compute the distances. Since for each $(s,t) \in S^2_?$

we have $0 \leq d(s,t) \leq 1$, we can set the lower bound of the objective function $\sum_{(s,t)\in S_?^2} d(s,t)$ to 0 and the upper-bound to n. In each iteration of the binary search, we run the ellipsoid method `ellipsoid`($\mathbf{LP}, c, \mathbf{Separation}$), where $c \in [0, n]$.

There are several input parameters to be set for the procedure `ellipsoid`($\mathbf{LP}, c,$ $\mathbf{Separation}$). Let η be the size of the $\mathbf{LP}$, the maximum number of iterations N can be set to $125n^3\eta$ according to [77, page 173].

A ball in n dimensions can be defined by the set of points $(x_1, ..., x_n)$ which is represented by the equation

$$R^2 \geq \sum_{i=1}^{n} (x_i - c_i)^2 \tag{4.1}$$

where R is the radius of the ball and $(c_1, \cdots, c_n)$ is the centre of the ball. The initial ellipsoid is a ball with centre $\mathbf{0}$, so $c_i = 0$ for all $1 \leq i \leq n$. Since for each $(s,t) \in S_?^2$ we have $0 \leq d(s,t) \leq 1$, the maximum value of the right hand side of (4.1) can be obtained by replacing x_i with one for all i. Thus, the radius of the ball R can be set to $\sqrt{n}$.

We need one more ingredient for the ellipsoid method, the polynomial time separation algorithm. Firstly, the separation algorithm for `ellipsoid`($\mathbf{LP}, c, \mathbf{Separation}$) should check in polynomial time if a distance function d is a feasible solution, that is, if d satisfies the following constraints:

- C1: $\forall (s,t) \in S_?^2 : \ d(s,t) \geq 0$;
- C2: $\forall (s,t) \in S_?^2 : \ d(s,t) \leq 1$;

- C3: $\forall (s,t) \in S^2_? : \ \forall \omega \in V(\Omega(\tau(s),\tau(t))) : \ d(s,t) \leq \sum_{(u,v)\in S^2_?} \omega(u,v)\, d(u,v) + \sum_{(u,v)\in S^2_1} \omega(u,v)$;
- C4: $\sum_{(s,t)\in S^2_?} d(s,t) \geq c$.

C1, C2 and C3 are the constraints in **LP**. The constraint C4 is added in each iteration of the binary search algorithm.

It is obvious that checking the constraints C1, C2 and C4 only requires polynomial time. For the constraint C3, it suffices to check whether

$$d(s,t) \leq \min_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{(u,v)\in S^2} \omega(u,v)\, d(u,v)$$

is satisfied for all $(s,t) \in S^2_?$. Thus, it suffices to give a polynomial time procedure to compute a $\pi \in S^2 \to [0,1]$ such that

$$\pi(s,t) = \operatorname*{argmin}_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{(u,v)\in S^2} \omega(u,v)\, d(u,v).$$

This problem can be formulated as the minimum-cost flow problem.

minimize $\sum_{(s,t)\in S^2_?} \omega(s,t)\, d(s,t)$ such that

$$\begin{aligned} \forall u \in S : \ \sum_{v\in S} \omega(u,v) &= \tau(s)(u) \\ \forall v \in S : \ \sum_{u\in S} \omega(u,v) &= \tau(t)(v) \\ \forall (s,t) \in S^2 : \ \omega(s,t) &\geq 0 \end{aligned}$$

This problem can be solved in polynomial time using, for example, Orlin's network simplex algorithm [70].

Secondly, if any of the C1-C4 constraints is violated, the following proposition shows that a polynomial time procedure to find a separation hyperplane exists.

Proposition 4.2.1. *If d is not a feasible solution, the separating hyperplane can be defined as follows.*

(a) If there exists $(s,t) \in S^2_?$ such that $d(s,t) < 0$, the hyperplane is defined as

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

(b) If there exists $(s,t) \in S^2_?$ such that $d(s,t) > 1$, the hyperplane is defined as

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

(c) If there exist $(s,t) \in S^2_?$ and $\omega \in V(\Omega(\tau(s), \tau(t)))$ such that

$$d(s,t) > \sum_{(u,v) \in S^2} \omega(u,v)\, d(u,v)$$

the hyperplane is defined as

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

(d) If $\sum_{(s,t) \in S^2_?} d(s,t) < c$, the hyperplane is defined as $\alpha(u,v) = -1$ for all $(u,v) \in S^2_?$.

Proof. We have to show that the hyperplane α is a separating hyperplane such that

$$\sum_{(u,v) \in S^2_?} \alpha(u,v)\, d'(u,v) < \sum_{(u,v) \in S^2_?} \alpha(u,v)\, d(u,v)$$

for all d' satisfying the constraints C1-C4.

We consider four cases.

(a) If d violates C1, there must exist some $(s,t) \in S_?^2$ such that $d(s,t) < 0$. Let the hyperplane α be defined as in case (a) above. Then,

$$\sum_{(u,v)\in S_?^2} \alpha(u,v)\, d'(u,v) = -d'(s,t) \leq 0 < -d(s,t) = \sum_{(u,v)\in S_?^2} \alpha(u,v)\, d(u,v).$$

(b) If d violates C2, there must exist some $(s,t) \in S_?^2$ such that $d(s,t) > 1$. Let the hyperplane α be defined as in case (b) above. Then,

$$\sum_{(u,v)\in S_?^2} \alpha(u,v)\, d'(u,v) = d'(s,t) \leq 1 < d(s,t) = \sum_{(u,v)\in S_?^2} \alpha(u,v)\, d(u,v).$$

(c) If d violates C3, there must exist some $(s,t) \in S_?^2$ and $\omega \in V(\Omega(\tau(s),\tau(t)))$ such that $d(s,t) > \sum_{(u,v)\in S^2} \omega(u,v)\, d(u,v)$. Let the hyperplane α be defined as in case (c) above. Then,

$$\begin{aligned}
\sum_{(u,v)\in S_?^2} \alpha(u,v)\, d'(u,v) &= \alpha(s,t)\, d'(s,t) + \sum_{(u,v)\in S_?^2\setminus(s,t)} \alpha(u,v)\, d'(u,v) \\
&= (1-\omega(s,t))\, d'(s,t) + \sum_{(u,v)\in S_?^2\setminus(s,t)} -\omega(u,v)\, d'(u,v) \\
&= d'(s,t) - \sum_{(u,v)\in S_?^2} \omega(u,v)\, d'(u,v) \\
&\leq 0 \qquad [d' \text{ satisfies C3}] \\
&< d(s,t) - \sum_{(u,v)\in S_?^2} \omega(u,v)\, d(u,v) \\
&= (1-\omega(s,t))\, d(s,t) + \sum_{(u,v)\in S_?^2\setminus(s,t)} -\omega(u,v)\, d(u,v)
\end{aligned}$$

$$
\begin{aligned}
&= \alpha(s,t)\, d(s,t) + \sum_{(u,v)\in S_?^2 \setminus (s,t)} \alpha(u,v)\, d(u,v) \\
&= \sum_{(u,v)\in S_?^2} \alpha(u,v)\, d(u,v).
\end{aligned}
$$

(d) Assume C4 is violated, that is, $\sum_{(s,t)\in S_?^2} d(s,t) < c$. Let the hyperplane α be defined as in case (d) above. Then,

$$
\begin{aligned}
\sum_{(u,v)\in S_?^2} \alpha(u,v)\, d'(u,v) &= -\sum_{(u,v)\in S_?^2} d'(u,v) \\
&\leq -c \qquad [d' \text{ satisfies C4}] \\
&< -\sum_{(u,v)\in S_?^2} d(u,v) \\
&= \sum_{(u,v)\in S_?^2} \alpha(u,v)\, d(u,v).
\end{aligned}
$$

□

4.3 Probabilistic Automata

A linear program has an objective function, which is either maximized or minimized. In the definition of the probabilistic bisimilarity distances for probabilistic automata, when two states have the same label the distance is defined by a mixture of maximizations and minimizations. It is not straightforward whether the transformation to a linear program is possible. In [4], Bacci *et al.* define the probabilistic bisimilarity distances for probabilistic automata by maximizing the Hausdorff distance. However,

the definition of Hausdorff distance itself involves alternations of maximizations and minimizations. It is unclear whether we can formulate the computation of the probabilistic bisimilarity distances for probabilistic automata as a linear program.

5 Labelled Markov Chains and Markov Decision Processes

In this chapter, we introduce a Markov decision process (MDP) as a game played by a single player. As the player moves from one vertex of the game graph to another, a cost may be incurred. The goal of the game is to minimize the cost according to some criterion [ii]. We limit our goal to minimizing the cost according to the so-called total cost criterion, that is, we want to minimize the expected cumulative cost in the long run. There are other criteria, including the discounted cost criterion and the average cost criterion. For more details about the criteria, we refer the readers to, for example, [46, Chapter 2]. The MDPs we are interested in have positive cost and are stopping MDPs, a special kind of MDPs which we will introduce in Section 5.1. We will introduce the optimal policy that can minimize the expected cumulative cost for stopping MDPs. The so-called optimal value of a vertex is the expected cumulative cost of the player starting at that vertex and following an optimal policy.

[ii] In other settings, the goal of the game might be to maximize the rewards. For example, in the reinforcement learning literature, the goal of the player is to maximize the expected cumulative rewards in the long run (see, for example, [84, Section 1.3]).

In Section 5.2, we will present a classic algorithm by Howard [53] to compute such an optimal policy. In Section 5.3.1, we will present a new transformation mapping a labelled Markov chain to a stopping MDP, where the probabilistic bisimilarity distances of the labelled Markov chain correspond to the so-called optimal values of the stopping MDP. This transformation is similar to the one in [21] which maps a probabilistic automaton to a simple stochastic game (see Chapter 10).

5.1 Markov Decision Processes

MDPs were introduced by Bellman [12]. We can view an MDP as a game played by a single player [iii]. An MDP has vertices and actions where each vertex has at least one action. In a game, the player is at a vertex and chooses an action available at that vertex. The player then moves to a vertex which is chosen randomly according to the probability distribution determined by that action. Each transition incurs a cost which is a number in the unit interval $[0, 1]$. The goal of the player is to minimize the expected cumulative cost in the long run.

A *policy* for the player maps each vertex to one of the actions available at that vertex. In game theory, these policies are called pure and stationary (see, for example, [44]). The policies are pure since they map the vertices to one of the actions, not a probability distribution on the actions. The policies are stationary

[iii] The reinforcement learning community uses agent instead of player (see, for example [84]).

(or memoryless), since the policy is the same every time the player visits the same vertex. Policies are also known as *strategies*. Now let us formally define MDPs.

Definition 5.1.1. A *Markov decision process (MDP)* is a tuple $\langle V, A, \alpha, \pi, \mathbf{c} \rangle$ consisting of

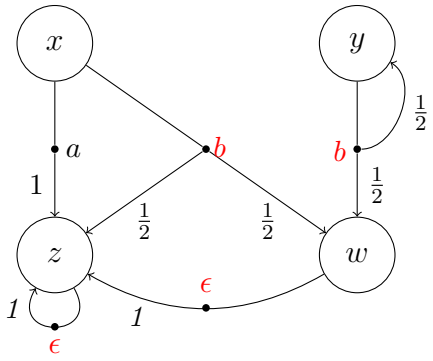
- a set V of *vertices*,
- a set A of *actions*,
- an *available action function* $\alpha : V \to 2^A$,
- a *transition probability function* $\pi : V \times A \to \mathrm{Distr}(V)$, and
- a *cost function* $\mathbf{c} : V \times A \times V \nrightarrow [0, 1]$.

Furthermore, we restrict our attention to MDPs with finitely many vertices and finitely many actions. We restrict the transition probabilities to be rationals. For each vertex $x \in V$, $\alpha(x)$ is the set of actions that are available at vertex x. We denote the cost of the transition from x to y under action a as $\mathbf{c}(x, a, y)$, where the action a should be available at x, that is, $a \in \alpha(x)$, and the transition probability should be positive, that is, $\pi(x, a)(y) > 0$.

The set of (total) policies is defined as follows. We will consider partial policies in Chapter 7.

Definition 5.1.2. The set $\mathcal{T}$ of *total policies* is defined by $\mathcal{T} = \{T \in A^V \mid \forall x \in V : T(x) \in \alpha(x)\}$.

Example 5.1.3. *We consider an MDP with four vertices: x, y, z and w. There are three actions: a, b and ϵ. The vertex x has two available actions a and b. All the other vertices have only one available action. The MDP can be depicted as the graph shown below. The transition probabilities are denoted along the edges. The cost of all transitions is zero except the one from w to z under the action ϵ which is one.*



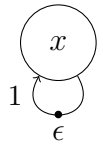
A total policy T maps each vertex to an available action at that vertex. For example, the following defines a policy.

$$\begin{aligned} T(x) &= b \\ T(y) &= b \\ T(z) &= \epsilon \\ T(w) &= \epsilon \end{aligned}$$

□

As there are no negative costs in an MDP, the expected cumulative cost is always non-negative. However, the expected cumulative cost is not always bounded as the

following simple MDP shows. The cost of the transition from x back to itself taking the action ϵ is one. As there is only one action available at vertex x, the player can only take this action. Obviously, the expected cumulative cost is not bounded.



In this dissertation, we consider a special kind of MDP of which the cost is always bounded (see [46, Section 2.4]). An MDP and a policy for the player give rise naturally to a Markov chain, where the vertices of the MDP are the states of the Markov chain (see Definition 5.3.3). A vertex is called a *terminal* vertex if no cost will be accumulated once such a vertex is reached. The value under any policy of a terminal vertex is zero. An MDP is *stopping* if for each policy, in the corresponding Markov chain, from each vertex the player reaches a terminal vertex with probability one.

For the remainder of this section, we fix a stopping MDP $\langle V, A, \alpha, \pi, \mathbf{c}\rangle$. Let us assume that the expected cumulative cost is bounded by one. The value function $v^T : V \to [0, 1]$ under a policy T maps each vertex x to the expected cumulative cost in the long run provided that the player plays according to the policy T and starts at vertex x. The expected cumulative cost is used to evaluate the policy T and can be characterized as the unique fixed point of the following function [46, Definition 2.4.4].

Definition 5.1.4. Let $T \in \mathcal{T}$ be a policy for the player. The function $\Theta^T : [0,1]^V \to [0,1]^V$ is defined by

$$\Theta^T(v)(x) = \begin{cases} \sum_{y \in V} \pi\big(x, T(x)\big)\big(y\big)\big(\mathbf{c}(x, T(x), y) + v(y)\big) & \text{if } x \text{ is not a terminal vertex} \\ 0 & \text{if } x \text{ is a terminal vertex.} \end{cases}$$

Theorem 5.1.5. *Θ^T has a unique fixed point.*

Proof. In [46, Definition 2.4.4], v^T is defined to be the unique solution of the following equations.

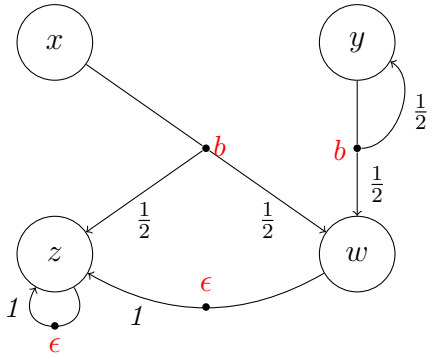
$$v^T(x) = \begin{cases} \sum_{y \in V} \pi\big(x, T(x)\big)\big(y\big)\big(\mathbf{c}(x, T(x), y) + v^T(x)\big) & \text{if } x \text{ is not a terminal vertex} \\ 0 & \text{if } x \text{ is a terminal vertex.} \end{cases}$$

□

Θ^T has a unique fixed point which is equal to v^T, thus the values of the vertices under the policy T are defined as the unique fixed point of Θ^T.

Example 5.1.6. *Consider the MDP in Example 5.1.3. Vertex z is a terminal vertex as it can only go back to itself by taking the action ϵ and the cost of that transition is zero. Vertex w reaches z by taking ϵ and ϵ is the only available action at z. Vertex y can take its only available action b and go to w first, from which it reaches z. Vertex x has two available actions. If the player chooses a, it reaches z. Otherwise, it chooses b and reaches z and w each with probability $\frac{1}{2}$. Hence, it's a stopping MDP.*

The MDP and the policy T of Example 5.1.3 give rise to a Markov chain which is shown below.



We can calculate the values of the vertices under the policy T by solving the following equations according to Definition 5.1.4.

$$
\begin{aligned}
v^T(x) &= \Theta^T(v^T)(x) = \tfrac{1}{2} \times \Big(\mathbf{c}\big(x,b,z\big) + v^T(z)\Big) + \\
&\quad \tfrac{1}{2} \times \Big(\mathbf{c}\big(x,b,w\big) + v^T(w)\Big) && [T(x) = b] \\
v^T(y) &= \Theta^T(v^T)(y) = \tfrac{1}{2} \times \Big(\mathbf{c}\big(y,b,y\big) + v^T(y)\Big) + \\
&\quad \tfrac{1}{2} \times \Big(\mathbf{c}\big(y,b,w\big) + v^T(w)\Big) && [T(y) = b] \\
v^T(z) &= \Theta^T(v^T)(z) = 0 && [z \text{ is a terminal vertex}] \\
v^T(w) &= \Theta^T(v^T)(w) = 1 \times \Big(\mathbf{c}\big(w,\epsilon,z\big) + v^T(z)\Big) && [T(w) = \epsilon]
\end{aligned}
$$

Note that all costs are zero except $\mathbf{c}\big(w,\epsilon,z\big) = 1$. The values of the vertices under the policy T, which are the solutions of the above equations, are shown in the table below.

x	y	z	w
$\frac{1}{2}$	1	0	1

□

The set $[0,1]^V$ is endowed with the partial order $\sqsubseteq$ defined as $v \sqsubseteq v'$ if and only if $v(x) \leq v'(x)$ for all $x \in V$ (cf. Definition 2.1.18). A policy T is better than a policy T' if $v^T(x) \leq v^{T'}(x)$ for every $x \in V$, that is, $v^T \sqsubseteq v^{T'}$. A policy is *optimal* if it minimizes the expected accumulated cost. It has been shown that every MDP has an optimal policy [46, Theorem 2.1.9]. An optimal policy T_{opt} for a stopping MDP satisfies $v^{T_{opt}} \sqsubseteq v^T$ for every policy T.

The optimal values of a stopping MDP, that is, the values of the vertices under the optimal policy, can be characterized as the least fixed point of the following function.

Definition 5.1.7. The function $\Phi : [0,1]^V \to [0,1]^V$ is defined by

$$\Phi(f)(x) = \begin{cases} \min\limits_{a \in \alpha(x)} \sum\limits_{y \in V} \pi(x,a)(y)\big(\mathbf{c}(x,a,y) + f(y)\big) & \text{if } x \text{ is not a terminal vertex} \\ 0 & \text{if } x \text{ is a terminal vertex.} \end{cases}$$

Proposition 5.1.8. *For all $f, g \in [0,1]^V$, if $f \sqsubseteq g$ then $\Phi(f) \sqsubseteq \Phi(g)$.*

Proof. Let $f, g \in [0,1]^V$ with $f \sqsubseteq g$. Let $x \in V$. We consider two cases.

- If x is a terminal vertex, $\Phi(f)(x) = 0 = \Phi(g)(x)$.
- Otherwise x is not a terminal vertex and

$$\begin{aligned} \Phi(f)(x) &= \min_{a \in \alpha(x)} \sum_{y \in V} \pi(x,a)(y)\big(\mathbf{c}(x,a,y) + f(y)\big) \\ &\leq \min_{a \in \alpha(x)} \sum_{y \in V} \pi(x,a)(y)\big(\mathbf{c}(x,a,y) + g(y)\big) \qquad [f \sqsubseteq g] \\ &= \Phi(g)(x). \end{aligned}$$

□

Since $\langle [0,1]^V, \sqsubseteq \rangle$ is a complete lattice (Proposition 2.1.19) and Φ is monotone, we can conclude from Theorem 2.1.8(a), the Knaster-Tarski fixed point theorem, that Φ has a least fixed point. We denote this least fixed point by $\boldsymbol{\mu}(\Phi)$.

Proposition 5.1.9. *For all $T \in \mathcal{T}$, we have $\boldsymbol{\mu}(\Phi) \sqsubseteq v^T$.*

Proof. Let $T \in \mathcal{T}$. According to Theorem 2.1.8(c), it suffices to prove that $\Phi(v^T) \sqsubseteq v^T$. Let x be a vertex. We consider two cases.

- If x is a terminal vertex, then

$$\boldsymbol{\mu}(\Phi)(x) = \Phi(\boldsymbol{\mu}(\Phi))(x) = 0 = \Theta^T(v^T)(x) = v^T(x).$$

- Otherwise, x is not a terminal vertex and

$$\begin{aligned}
\Phi(v^T)(x) &= \min_{a \in \alpha(x)} \sum_{y \in V} \pi(x,a)(y)\big(\mathbf{c}(x,a,y) + v^T(y)\big) \\
&\leq \sum_{y \in V} \pi(x,T(x))(y)\big(\mathbf{c}(x,T(x),y) + v^T(y)\big) \\
&= \Theta^T(v^T)(x) \\
&= v^T(x).
\end{aligned}$$

□

Proposition 5.1.10. *There exists a policy $T \in \mathcal{T}$ such that $v^T = \boldsymbol{\mu}(\Phi)$.*

Proof. For each non-terminal vertex x, we define

$$T(x) = \operatorname*{argmin}_{a \in \alpha(x)} \sum_{y \in V} \pi(x,a)(y)\big(\mathbf{c}(x,a,y) + \boldsymbol{\mu}(\Phi)(y)\big). \tag{5.1}$$

Since the set A is assumed to be finite, the above exists. By Theorem 5.1.5, v^T is the unique fixed point of the function Θ^T. To conclude that $v^T = \boldsymbol{\mu}(\Phi)$, it remains to show that $\boldsymbol{\mu}(\Phi)$ is a fixed point of Θ^T. Let $x \in V$. We distinguish two cases.

- If x is a terminal vertex, then

$$\Theta^T(\boldsymbol{\mu}(\Phi))(x) = 0 = \Phi(\boldsymbol{\mu}(\Phi))(x) = \boldsymbol{\mu}(\Phi)(x).$$

- Otherwise, x is not a terminal vertex. Then

$$\begin{aligned}
\Theta^T(\boldsymbol{\mu}(\Phi))(x) &= \sum_{y \in V} \pi(x, T(x))(y)\big(\mathbf{c}(x, T(x), y) + \boldsymbol{\mu}(\Phi)(y)\big) \\
&= \min_{a \in \alpha(x)} \sum_{y \in V} \pi(x,a)(y)\big(\mathbf{c}(x,a,y) + \boldsymbol{\mu}(\Phi)(y)\big) \qquad [(5.1)] \\
&= \Phi(\boldsymbol{\mu}(\Phi))(x) \\
&= \boldsymbol{\mu}(\Phi)(x).
\end{aligned}$$

□

Theorem 5.1.11. $\boldsymbol{\mu}(\Phi) = \min_{T \in \mathcal{T}} v^T$.

Proof. Immediate consequence of Proposition 5.1.9 and Proposition 5.1.10. □

Example 5.1.12. *We show that the policy T in Example 5.1.3 is not optimal. Let us define a new policy T', which is the same as T except that T' maps x to a. The values of the vertices under T' are shown in the table below.*

x	y	z	w
0	1	0	1

We have

$$v^T(x) = \tfrac{1}{2} > 0 = v^{T'}(x).$$

Thus, the policy T is not optimal. The reader can verify that $v^{T'}$ is the least fixed point of Φ and the policy T' is optimal. □

Next, we show that the function Φ is nonexpansive. This result will be used to show $\boldsymbol{\mu}(\Phi) = \boldsymbol{\mu}(\Delta)$ in Section 5.3.

Theorem 5.1.13. *For all $f, g \in [0,1]^V$, $||\Phi(f) - \Phi(g)|| \leq ||f - g||$.*

Proof. Let $f, g \in [0,1]^V$. Let $x \in V$. We distinguish two cases.

- Assume x is a terminal vertex. According to the definition of Φ, we have

$$|\Phi(f)(x) - \Phi(g)(x)| = |0 - 0| = 0 \leq ||f - g||.$$

- Otherwise, x is not a terminal vertex. Without loss of generality, assume that $\Phi(f)(x) \geq \Phi(g)(x)$. Let

$$b = \operatorname*{argmin}_{a \in \alpha(x)} \sum_{y \in V} \pi(x,a)(y)\big(\mathbf{c}(x,a,y) + g(y)\big).$$

We have

$$\begin{aligned}
|\Phi(f)(x) - \Phi(g)(x)| &= \Phi(f)(x) - \Phi(g)(x) \\
&= \min_{a\in\alpha(x)} \sum_{y\in V} \pi(x,a)(y)\big(\mathbf{c}(x,a,y) + f(y)\big) - \\
&\quad \min_{a\in\alpha(x)} \sum_{y\in V} \pi(x,a)(y)\big(\mathbf{c}(x,a,y) + g(y)\big) \\
&= \min_{a\in\alpha(x)} \sum_{y\in V} \pi(x,a)(y)\big(\mathbf{c}(x,a,y) + f(y)\big) - \\
&\quad \sum_{y\in V} \pi(x,b)(y)\big(\mathbf{c}(x,b,y) + g(y)\big) \\
&\leq \sum_{y\in V} \pi(x,b)(y)\big(\mathbf{c}(x,b,y) + f(y)\big) - \\
&\quad \sum_{y\in V} \pi(x,b)(y)\big(\mathbf{c}(x,b,y) + g(y)\big) \\
&= \sum_{y\in V} \pi(x,b)(y)\big(f(y) - g(y)\big) \\
&\leq \sum_{y\in V} \pi(x,b)(y)||f-g|| \\
&= ||f-g||.
\end{aligned}$$

□

5.2 Policy Iteration for MDPs

Next, we review Howard's policy iteration algorithm which correctly computes an optimal policy of an MDP in a finite number of iterations [53]. Note that we only consider stopping MDPs but Howard's policy iteration algorithm can correctly compute an optimal policy for MDPs in general.

Consider a stopping MDP $\langle V, A, \alpha, \pi, \mathbf{c}\rangle$. The action

$$b = \operatorname*{argmin}_{a\in\alpha(x)} \sum_{y\in V} \pi(x,a)(y)\big(\mathbf{c}(x,a,y) + v^T(y)\big)$$

for vertex x is called optimal with respect to a fixed policy T. Note that b may not be unique. We call a vertex x of an MDP switchable with respect to a fixed policy T if $T(x)$ is not an optimal action, that is,

$$\sum_{y\in V} \pi(x,T(x))(y)\big(\mathbf{c}(x,T(x),y) + v^T(y)\big) > \min_{a\in\alpha(x)} \sum_{y\in V} \pi(x,a)(y)\big(\mathbf{c}(x,a,y) + v^T(y)\big).$$

The policy iteration algorithm starts with a random policy. It repeatedly selects switchable vertices, and switches them. That is, it changes the policy so that the actions corresponding to the selected vertices are replaced by the optimal actions of those vertices. The algorithm halts when there is no switchable vertex, in which case an optimal policy has been found.

Different policy iteration algorithms have different select procedures which we denote by `select` in the code below. Such a select procedure determines the set of vertices to be switched. The select procedures for three different policy iteration algorithms are as follows:

- Simple policy iteration algorithm: it selects only one of the switchable vertices. Specifically, it assigns each vertex a unique id and it selects the switchable vertex with the largest id.

- General policy iteration algorithm: it selects all the switchable vertices.

- Random policy iteration algorithm: it selects a switchable vertex uniformly at random from the set of switchable vertices.

```
1 T = a random policy
2 S = {x ∈ V | x is switchable with respect to T}
3 while S ≠ ∅
4     for each x ∈ select(S)
5         T(x) = argmin_{a∈α(x)} Σ_{y∈V} π(x,a)(y)(c(x,a,y) + v^T(y))
6     S = {x ∈ V | x is switchable with respect to T}
```

5.3 Labelled Markov Chains

Now, we are ready to introduce the transformation that maps each labelled Markov chain to a stopping MDP such that distances correspond to optimal values.

Definition 5.3.1. Let $\langle S, L, \tau, \ell \rangle$ be a labelled Markov chain. The corresponding Markov decision process $\langle V, A, \alpha, \pi, \mathbf{c} \rangle$ consists of

- the set of *vertices* $V = S^2 \cup \{\boldsymbol{\infty}\}$,
- the set of *actions* $A = \bigcup\{\, V(\Omega(\tau(s), \tau(t))) \mid x \in S^2_? \,\} \cup \{\epsilon\}$,
- the *available actions function* $\alpha : V \to 2^A$ defined by

$$\alpha(x) = \begin{cases} V(\Omega(\tau(s), \tau(t))) & \text{if } x = (s,t) \wedge (s,t) \in S^2_? \\ \{\epsilon\} & \text{if } x \in S^2_0 \cup S^2_1 \cup \{\boldsymbol{\infty}\} \end{cases}$$

- the *probability transition function* $\pi : V \times A \to \mathrm{Distr}(V)$ defined by

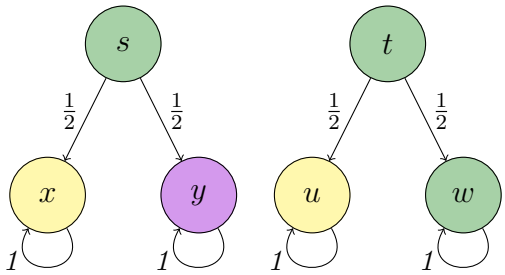
$$\pi(x,a)(y) = \begin{cases} a(u,v) & \text{if } x \in S^2_? \wedge a \in \alpha(x) \wedge y \in S^2 \wedge y = (u,v) \\ 1 & \text{if } x \in S^2_0 \cup S^2_1 \cup \{\boldsymbol{\infty}\} \wedge a = \epsilon \wedge y = \boldsymbol{\infty} \\ 0 & \text{otherwise} \end{cases}$$

- the *cost function* $\mathbf{c} : V \times A \times V \to [0,1]$ defined by

$$\mathbf{c}(x,a,y) = \begin{cases} 1 & \text{if } x \in S^2_1 \wedge a = \epsilon \wedge y = \boldsymbol{\infty} \\ 0 & \text{otherwise} \end{cases}$$

Note that $\boldsymbol{\infty}$ is a *terminal vertex* as it has only one transition which takes it back to itself and there is no cost associated with this transition. Let $(s,t) \in S^2_0$. This vertex has only one transition which always takes it to the terminal vertex $\boldsymbol{\infty}$. The cost associated with that transition is zero. Let $(s,t) \in S^2_1$. Similarly, there is only one transition which always takes (s,t) to the terminal vertex $\boldsymbol{\infty}$. However, the cost of that transition is one. Let $(s,t) \in S^2_?$. The available actions at vertex (s,t) are the couplings in $V(\Omega(\tau(s),\tau(t)))$ and the cost is zero no matter which action is taken. The probability of transitioning from (s,t) to (u,v) with action ω is $\omega(u,v)$. Recall that a coupling is a probability distribution on S^2.

Example 5.3.2. *We consider the labelled Markov chain with six states shown below. The labels are represented by the colours of the states.*



Let $V(\Omega(\tau(s), \tau(t))) = \{\rho, \omega\}$ *with*

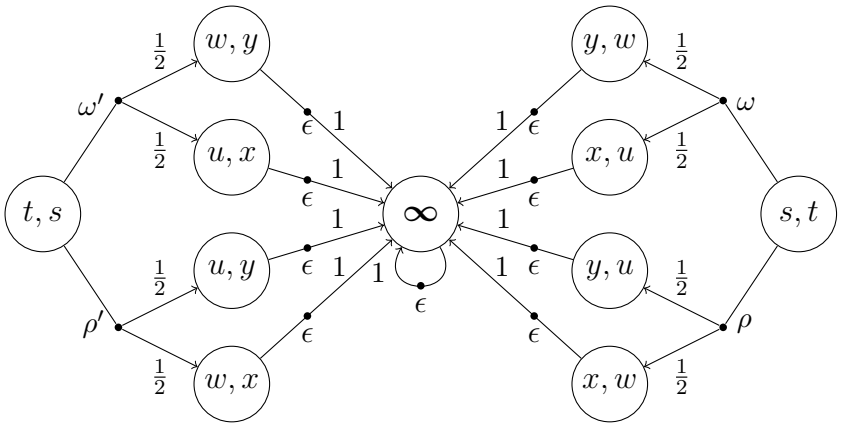
$$\rho(x, w) = \rho(y, u) = \omega(x, u) = \omega(y, w) = \tfrac{1}{2}.$$

Let $V(\Omega(\tau(t), \tau(s))) = \{\rho', \omega'\}$ *with*

$$\rho'(w, x) = \rho'(u, y) = \omega'(u, x) = \omega'(w, y) = \tfrac{1}{2}.$$

According to Definition 5.3.1, the labelled Markov chain is transformed into the MDP shown below. The cost of all transitions is zero, except the ones starting from vertices in S_1^2*, that is, the cost is one for the transitions starting from* (x, w)*,* (y, w)*,* (y, u) *and their symmetric counterparts* (w, x)*,* (w, y)*,* (u, y)*.*

We will show later that the distance of a state pair in a labelled Markov chain is equivalent to the expected cumulative cost of the vertex in the corresponding MDP under the optimal policy. As can be seen from the following figure, the MDP is symmetric since the distance function is symmetric.



The MDP mapped to the labelled Markov chain above.

□

Next, we will show that the MDP transformed from a labelled Markov chain according to Definition 5.3.1 is stopping.

By using only the actions of the policy, an MDP and a policy naturally give rise to a Markov chain, where the vertices of the MDP are the states of the Markov chain. We will call this the coupled Markov chain.

Definition 5.3.3. Let $\langle V, A, \alpha, \pi, \mathbf{c} \rangle$ be an MDP. Let $T \in \mathcal{T}$ be a policy. The *coupled Markov chain* $\langle S, \tau \rangle$ consists of

- the set of states $S = V$, and
- the transition probability function $\tau : S \to \text{Distr}(S)$ is defined by

$$\tau(s) = \pi(s, T(s)).$$

We define the relation I by $I = \{ (s, s) \mid s \in S \}$. Given the relations P, $R \subseteq S^2$, we define the relations R^{-1} and $P \circ R$ by

$$\begin{aligned} R^{-1} &= \{ (t, s) \mid (s, t) \in R \} \\ P \circ R &= \{ (s, u) \mid \exists t \in S : (s, t) \in P \wedge (t, u) \in R \} \end{aligned}$$

A strongly connected component of a Markov chain is a maximal set of states such that there is a path between every pair of states in the set. A bottom strongly connected component of a Markov chain, also known as a closed communication class, is a strongly connected component such that every state in the component can only reach states in the component. For details on the strongly connected components and bottom strongly connected components, we refer interested readers to, for example, [8, Notation 10.26].

Lemma 5.3.4. *For a relation $R \subseteq S^2$, let the predicate $B(R)$* [iv] *be defined by*

$B(R)$ *iff* $\forall (s, t) \in R : \ell(s) = \ell(t) \wedge \exists \omega \in \Omega(\tau(s), \tau(t)) : \text{support}(\omega) \subseteq R$. *Then,*

(a) $B(I)$.

(b) If $B(R)$ *then* $B(R^{-1})$.

(c) If $B(P)$ *and* $B(R)$ *then* $B(P \circ R)$.

(d) If $B(P)$ *and* $B(R)$ *then* $B(P \cup R)$.

[iv] The predicate $B(R)$ captures that R is a probabilistic bisimulation if R is an equivalence relation.

(e) If C is a closed communication class not containing any element in $S_0^2 \cup S_1^2$, then $B(C)$.

(f) If C is a closed communication class not containing any element in $S_0^2 \cup S_1^2$ and $\bar{C}$ is the reflexive, symmetric and transitive closure of C, then $B(\bar{C})$.

(g) If C is a closed communication class not containing any element in $S_0^2 \cup S_1^2$ and $(s,t) \in C$, then $s \sim t$.

Proof. (a) Let $s \in S$. We define $\omega : S^2 \to [0,1]$ by

$$\omega(u,v) = \begin{cases} \tau(s)(u) & \text{if } u = v \\ 0 & \text{otherwise} \end{cases}$$

It is proved in Proposition 2.1.29 that $\omega \in \Omega(\tau(s), \tau(s))$. We also observe that $\text{support}(\omega) \subseteq I$. Since also for all $s \in S$, $\ell(s) = \ell(s)$, we can conclude that $B(I)$.

(b) Assume that $B(R)$. Let $(s,t) \in R$. Since $B(R)$, we have that $\ell(s) = \ell(t)$ and there exists $\pi \in \Omega(\tau(t), \tau(s))$ such that $\text{support}(\pi) \subseteq R$. We define $\rho : S^2 \to [0,1]$ by

$$\rho(u,v) = \pi(v,u).$$

According to the proof of Proposition 2.1.27, $\rho \in \Omega(\tau(s), \tau(t))$. Furthermore,

$$\text{support}(\rho) \;=\; \{\, (u,v) \mid \rho(u,v) > 0 \,\}$$

$$
\begin{aligned}
&= \{\, (u,v) \mid \pi(v,u) > 0 \,\} \\
&\subseteq \{\, (u,v) \mid (v,u) \in R \,\} \qquad [\text{support}(\pi) \subseteq R] \\
&= R^{-1}.
\end{aligned}
$$

Hence, we can conclude that $B(R^{-1})$.

(c) Assume that $B(P)$ and $B(R)$. Suppose that $(s,u) \in P \circ R$, that is, $(s,t) \in P$ and $(t,u) \in R$ for some $t \in S$. Since $B(P)$ and $B(R)$, we have that $\ell(s) = \ell(t)$ and $\ell(t) = \ell(u)$ and, hence, $\ell(s) = \ell(u)$. From $B(P)$ and $B(R)$, we can conclude that there exist $\pi_{st} \in \Omega(\tau(s), \tau(t))$ and $\pi_{tu} \in \Omega(\tau(t), \tau(u))$ such that $\text{support}(\pi_{st}) \subseteq P$ and $\text{support}(\pi_{tu}) \subseteq R$. We define the function $\pi_{su} : S^2 \to [0,1]$ by

$$
\pi_{su}(x,z) = \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{st}(x,y)\pi_{tu}(y,z)}{\tau(t)(y)}.
$$

According to the proof of Proposition 2.1.28, we have $\pi_{su} \in \Omega(\tau(s), \tau(u))$. Thus, for all x, $z \in S$,

$$
\begin{aligned}
&(x,z) \in \text{support}(\pi_{su}) \\
&\quad \text{iff} \quad \sum_{y \in \text{support}(\tau(t))} \frac{\pi_{st}(x,y)\pi_{tu}(y,z)}{\tau(t)(y)} > 0 \\
&\quad \text{iff} \quad \exists y \in \text{support}(\tau(t)) : \pi_{st}(x,y) > 0 \wedge \pi_{tu}(y,z) > 0 \\
&\quad \text{iff} \quad \exists y \in \text{support}(\tau(t)) : (x,y) \in \text{support}(\pi_{st}) \wedge (y,z) \in \text{support}(\pi_{tu}) \\
&\quad \text{implies} \quad \exists y \in \text{support}(\tau(t)) : (x,y) \in P \wedge (y,z) \in R \\
&\qquad\qquad [\text{support}(\pi_{st}) \subseteq P \text{ and } \text{support}(\pi_{tu}) \subseteq R]
\end{aligned}
$$

implies $(x, z) \in P \circ R$.

Therefore, $B(P \circ R)$.

(d) Assume that $B(P)$ and $B(R)$. Suppose that $(s, t) \in P \cup R$. Let us assume that $(s, t) \in P$. The other case, when $(s, t) \in R$, can be proved similarly. Since $B(P)$, we can conclude that $\ell(s) = \ell(t)$ and there exists $\omega \in \Omega(\tau(s), \tau(t))$ such that support$(\omega) \subseteq P \subseteq P \cup R$. Hence, $B(P \cup R)$.

(e) Assume that C is a closed communication class that does not contain any element in $S_0^2 \cup S_1^2$. Let $(s, t) \in C$. Since $(s, t) \notin S_1^2$, $\ell(s) = \ell(t)$.

Let u, $v \in S$. Since C is a closed communication class, if $\pi((s, t), (u, v)) > 0$ then $(u, v) \in C$. From the construction of the Markov chain and the fact that (s, t) is in neither S_0^2 nor S_1^2, we can conclude that $\pi((s, t), (u, v)) = T(s, t)(u, v)$. We have that $T(s, t) \in \Omega(\tau(s), \tau(t))$ and support$(T(s, t)) = \{ (u, v) \mid \pi(s, t)(u, v) > 0 \} \subseteq C$. Hence, $B(C)$.

(f) Let C be a closed communication class not containing any element in $S_0^2 \cup S_1^2$. From part (a) and (d) we can conclude that the reflexive closure of C satisfies B. From part (b) and (d) we can deduce that the reflexive and symmetric closure satisfies B as well. Finally, from part (c) and (d) we can conclude that the transitive closure of the reflexive and symmetric closure also satisfies B.

(g) Let C be a closed communication class not containing any element in $S_0^2 \cup S_1^2$

and let $\bar{C}$ be the reflexive, symmetric and transitive closure of C. Since $\bar{C}$ is an equivalence relation and $B(\bar{C})$ by part (f), we can conclude that $\bar{C}$ is a probabilistic bisimulation. If $(s,t) \in C$ then $(s,t) \in \bar{C}$ and, hence, $s \sim t$.

□

Theorem 5.3.5. *The MDP defined in Definition 5.3.1 is a stopping MDP.*

Proof. The proof is very similar to the one of [85, Theorem 14]. To show that the MDP is stopping, we need to show that every vertex reaches $\boldsymbol{\infty}$ with probability one. Obviously, the vertices in S_0^2 and S_1^2 reach $\boldsymbol{\infty}$ with probability one. It remains to consider the vertices in $S_?^2$. Due to the construction of the MDP in Definition 5.3.1, if a vertex in $S_?^2$ cannot reach a vertex in $S_0^2 \cup S_1^2$, it cannot reach $\boldsymbol{\infty}$. Hence, it suffices to show that in a coupled Markov chain induced by any policy T, a vertex $(s,t) \in S_?^2$ reaches elements in $S_0^2 \cup S_1^2$ with probability one. Towards a contradiction, assume there exists a policy such that a vertex $(s,t) \in S_?^2$ does not reach elements in $S_0^2 \cup S_1^2$ with probability one in the coupled Markov chain. Each vertex in a Markov chain reaches with probability one a closed communication class (see, for example, [8, Theorem 10.27]). Since (s,t) does not reach elements in $S_0^2 \cup S_1^2$ with probability one, (s,t) reaches a closed communication class C not containing any element in $S_0^2 \cup S_1^2$. According to part (g) of Lemma 5.3.4, $u \sim v$ for every $(u,v) \in C$. Hence, $(u,v) \in S_0^2$ which contradicts the fact that C does not contain any elements of $S_0^2 \cup S_1^2$. □

As we will show next, there is direct correspondence between the function Φ from Definition 5.1.7 and the function Δ from Definition 2.1.7. From this correspondence it is straightforward that the optimal values of the vertices in the MDP and the probabilistic bisimilarity distances of the labelled Markov chain agree.

Theorem 5.3.6. *For all* $(s,t) \in S^2, \boldsymbol{\mu}(\Phi)(s,t) = \boldsymbol{\mu}(\Delta)(s,t)$. [v]

Proof. First, let us show that for all $n \in \mathbb{N}$,

$$\Phi^n(\mathbf{0})(\boldsymbol{\infty}) = 0 \tag{5.2}$$

by induction on n. Obviously, the above holds if $n = 0$. Let $n > 0$. We have $\Phi^n(\mathbf{0})(\boldsymbol{\infty}) = \Phi\big(\Phi^{n-1}(\mathbf{0})\big)(\boldsymbol{\infty}) = 0$ since $\boldsymbol{\infty}$ is a terminal state.

Next, we show that for all $n \in \mathbb{N}$ and for all $(s,t) \in S_0^2$,

$$\Delta^n(\mathbf{0})(s,t) = 0 \tag{5.3}$$

by induction on n. Obviously, the above holds if $n = 0$.

Let $n > 0$. Let $(s,t) \in S_0^2$. By Definition 2.1.5, $\ell(s) = \ell(t)$, and there exists an $\rho \in \Omega(\tau(s), \tau(t))$ such that $\text{support}(\rho) \subseteq \sim$.

$$\begin{aligned}
\Delta^n(\mathbf{0})(s,t) &= \min_{\omega \in \Omega(\tau(s),\tau(t))} \sum_{u,v \in S} \omega(u,v)\, \Delta^{n-1}(\mathbf{0})(u,v) \\
&\leq \sum_{u,v \in S} \rho(u,v)\, \Delta^{n-1}(\mathbf{0})(u,v) \\
&= \sum_{(u,v) \in \sim} \rho(u,v)\, \Delta^{n-1}(\mathbf{0})(u,v) \qquad [\text{support}(\rho) \subseteq \sim]
\end{aligned}$$

[v] We do not write $\boldsymbol{\mu}(\Phi) = \boldsymbol{\mu}(\Delta)$ since $\boldsymbol{\infty}$ is a vertex in the MDP.

$$= \quad 0 \qquad \text{[induction hypothesis: } \Delta^{n-1}(\mathbf{0})(u,v) = 0 \text{ for all } u \sim v\text{]}$$

Next, we show that for all $s, t \in S$ and $n \in \mathbb{N}$,

$$\Phi^n(\mathbf{0})(s,t) = \Delta^n(\mathbf{0})(s,t) \tag{5.4}$$

by induction on n. Obviously, the above holds if $n = 0$. Let $n > 0$. We distinguish the following cases.

- If $(s,t) \in S_1^2$ then

$$\begin{aligned}
\Phi^n(\mathbf{0})(s,t) &= \min_{a \in \alpha(s,t)} \sum_{y \in V} \pi((s,t),a)(y)\big(\mathbf{c}((s,t),a,y) + \Phi^{n-1}(\mathbf{0})(y)\big) \\
&= \pi((s,t),\epsilon)(\boldsymbol{\infty})\big(\mathbf{c}((s,t),\epsilon,\boldsymbol{\infty}) + \Phi^{n-1}(\mathbf{0})(\boldsymbol{\infty})\big) \\
&\qquad [\alpha(s,t) = \{\epsilon\} \text{ since } (s,t) \in S_1^2] \\
&= 1 \times (1+0) \\
&\qquad [\pi\big((s,t),\epsilon\big)\big(\boldsymbol{\infty}\big) = 1,\ \mathbf{c}\big((s,t),\epsilon,\boldsymbol{\infty}\big) = 1 \text{ and } (5.2)] \\
&= 1 \\
&= \Delta(\Delta^{n-1}(\mathbf{0}))(s,t) \\
&= \Delta^n(\mathbf{0})(s,t).
\end{aligned}$$

- If $(s,t) \in S_0^2$ then

$$\begin{aligned}
\Phi^n(\mathbf{0})(s,t) &= \min_{a \in \alpha(s,t)} \sum_{y \in V} \pi((s,t),a)(y)\big(\mathbf{c}((s,t),a,y) + \Phi^{n-1}(\mathbf{0})(y)\big) \\
&= \pi((s,t),\epsilon)(\boldsymbol{\infty})\big(\mathbf{c}((s,t),\epsilon,\boldsymbol{\infty}) + \Phi^{n-1}(\mathbf{0})(\boldsymbol{\infty})\big) \\
&\qquad [\alpha(s,t) = \{\epsilon\} \text{ since } (s,t) \in S_0^2] \\
&= 1 \times (0+0) \\
&\qquad [\pi\big((s,t),\epsilon\big)\big(\boldsymbol{\infty}\big) = 1,\ \mathbf{c}\big((s,t),\epsilon,\boldsymbol{\infty}\big) = 0 \text{ and } (5.2)] \\
&= 0 \\
&= \Delta^n(\mathbf{0})(s,t) \qquad [(5.3)].
\end{aligned}$$

- Otherwise, $(s,t) \in S_?^2$. Then

$$\forall \omega \in \alpha(s,t) : \forall (u,v) \in S^2 : \pi\big((s,t),\omega\big)(u,v) = \omega(u,v) \tag{5.5}$$

$$\forall \omega \in \alpha(s,t) : \forall (u,v) \in S^2 : \mathbf{c}\big((s,t),\omega,(u,v)\big) = 0 \tag{5.6}$$

Hence,

$$\begin{aligned}
\Phi^n(\mathbf{0})(s,t) &= \min_{a\in\alpha(s,t)} \sum_{y\in V} \pi((s,t),a)(y)\big(\mathbf{c}((s,t),a,y) + \Phi^{n-1}(\mathbf{0})(y)\big) \\
&= \min_{\omega\in V(\Omega(\tau(s),\tau(t)))} \sum_{(u,v)\in S^2} \pi((s,t),\omega)(y)\Big(\mathbf{c}\big(s,t),\omega,y\big) + \\
&\qquad\qquad \Phi^{n-1}(\mathbf{0})(y)\Big) \\
&\qquad [\alpha(s,t) = V(\Omega(\tau(s),\tau(t))) \text{ since } (s,t)\in S^2_?] \\
&= \min_{\omega\in V(\Omega(\tau(s),\tau(t)))} \sum_{(u,v)\in S^2} \omega(u,v)\Big(0 + \Phi^{n-1}(\mathbf{0})(u,v)\Big) \\
&\qquad [(5.5) \text{ and } (5.6)] \\
&= \min_{\omega\in V(\Omega(\tau(s),\tau(t)))} \sum_{(u,v)\in S^2} \omega(u,v)\,\Phi^{n-1}(\mathbf{0})(u,v) \\
&= \min_{\omega\in V(\Omega(\tau(s),\tau(t)))} \sum_{(u,v)\in S^2} \omega(u,v)\,\Delta^{n-1}(\mathbf{0})(u,v) \\
&\qquad [\text{induction hypothesis}] \\
&= \Delta^n(\mathbf{0})(s,t).
\end{aligned}$$

Now, we are ready to prove the result. Let $(s,t)\in S^2$. Then

$$\begin{aligned}
\boldsymbol{\mu}(\Phi)(s,t) &= \sup_{n\in\mathbb{N}} \Phi^n(\mathbf{0})(s,t) \\
&\qquad [\text{Proposition 5.1.8, Proposition 5.1.13 and Theorem 2.1.21}] \\
&= \sup_{n\in\mathbb{N}} \Delta^n(\mathbf{0})(s,t) \qquad [(5.4)] \\
&= \boldsymbol{\mu}(\Delta)(s,t) \\
&\qquad [\text{Proposition 2.1.13, Proposition 2.1.26 and Theorem 2.1.21}].
\end{aligned}$$

□

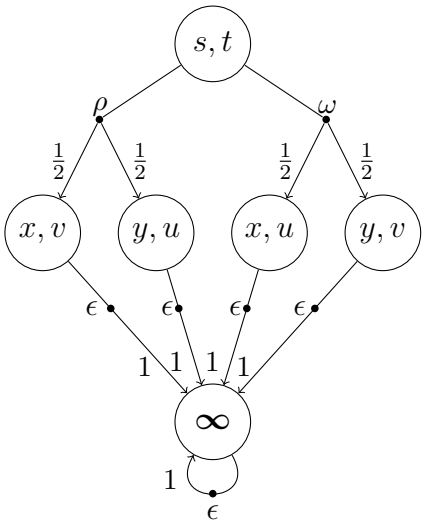
Example 5.3.7. *We are interested in the distance of the states s and t of the labelled Markov chain in Example 5.3.2. We have*

$$\begin{aligned}
\boldsymbol{\mu}(\Delta)(s,t) &= \Delta(\boldsymbol{\mu}(\Delta))(s,t) \qquad [\boldsymbol{\mu}(\Delta) \textit{ is a fixed point of } \Delta] \\
&= \min_{\omega \in V(\Omega(\tau(s),\tau(t))} \sum_{(u,v)\in S^2} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) \\
&= \min_{\omega \in \{\rho,\omega\}} \sum_{(u,v)\in S^2} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) \qquad [V(\Omega(\tau(s),\tau(t)) = \{\rho,\omega\}] \\
&= \min \left\{ \sum_{(u,v)\in S^2} \rho(u,v)\, \boldsymbol{\mu}(\Delta)(u,v), \sum_{(u,v)\in S^2} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) \right\} \\
&= \min \left\{ \tfrac{1}{2}\, \boldsymbol{\mu}(\Delta)(x,w) + \tfrac{1}{2}\, \boldsymbol{\mu}(\Delta)(y,u), \tfrac{1}{2}\, \boldsymbol{\mu}(\Delta)(x,u) + \tfrac{1}{2}\, \boldsymbol{\mu}(\Delta)(y,w) \right\} \\
&\qquad [\rho(x,w) = \rho(y,u) = \omega(x,u) = \omega(y,w) = \tfrac{1}{2}] \\
&= \min \left\{ \tfrac{1}{2} \times 1 + \tfrac{1}{2} \times 1 \,, \tfrac{1}{2} \times 0 + \tfrac{1}{2} \times 1 \right\} \\
&\qquad [\boldsymbol{\mu}(\Delta)(x,w) = \boldsymbol{\mu}(\Delta)(y,u) = \boldsymbol{\mu}(\Delta)(y,w) = 1 \wedge \boldsymbol{\mu}(\Delta)(x,u) = 0] \\
&= \min\{1 \,, \tfrac{1}{2}\} \\
&= \tfrac{1}{2}.
\end{aligned}$$

The MDP below is transformed from the labelled Markov chain. Note that we only present half of the MDP which is related to (s,t).

There are two policies in this MDP denoted by W and U, where

$$W(x) = \begin{cases} \rho & \textit{if } x = (s,t) \\ \epsilon & \textit{otherwise} \end{cases} \qquad U(x) = \begin{cases} \omega & \textit{if } x = (s,t) \\ \epsilon & \textit{otherwise} \end{cases}$$



We can calculate the values of the vertices under the policy W by solving the following equations according to Definition 5.1.4.

$$
\begin{aligned}
v^W(s,t) &= \tfrac{1}{2} \times \Big(\mathbf{c}\big((s,t),\rho,(x,v)\big) + v^W(x,v)\Big) + \\
&\quad \tfrac{1}{2} \times \Big(\mathbf{c}\big((s,t),\rho,(y,u)\big) + v^W(y,u)\Big) && [W(s,t)=\rho] \\
v^W(x,v) &= 1 \times \Big(\mathbf{c}\big((x,v),\epsilon,\boldsymbol{\infty}\big) + v^W(\boldsymbol{\infty})\Big) && [W(x,v)=\epsilon] \\
v^W(y,u) &= 1 \times \Big(\mathbf{c}\big((y,u),\epsilon,\boldsymbol{\infty}\big) + v^W(\boldsymbol{\infty})\Big) && [W(y,u)=\epsilon] \\
v^W(\boldsymbol{\infty}) &= 0 \qquad [\boldsymbol{\infty} \textit{ is a terminal vertex}]
\end{aligned}
$$

Note that

$$\mathbf{c}\big((s,t),\rho,(x,v)\big) = \mathbf{c}\big((s,t),\rho,(y,u)\big) = 0$$

and

$$\mathbf{c}\big((x,v),\epsilon,\boldsymbol{\infty}\big) = \mathbf{c}\big((y,u),\epsilon,\boldsymbol{\infty}\big) = 1.$$

The values of the vertices under the policy W are the solution of the above equations shown in the table below.

(s,t)	(x,v)	(y,u)	$\boldsymbol{\infty}$
1	1	1	0

We can calculate the values of the vertices under the policy U by solving the following equations according to Definition 5.1.4.

$$\begin{aligned}
v^U(s,t) &= \tfrac{1}{2} \times \Big(\mathbf{c}\big((s,t),\omega,(x,u)\big) + v^U(x,u)\Big) + \\
&\quad \tfrac{1}{2} \times \Big(\mathbf{c}\big((s,t),\omega,(y,v)\big) + v^U(y,v)\Big) && [U(s,t)=\omega] \\
v^U(x,u) &= 1 \times \Big(\mathbf{c}\big((x,u),\epsilon,\boldsymbol{\infty}\big) + v^U(\boldsymbol{\infty})\Big) && [U(x,u)=\epsilon] \\
v^U(y,v) &= 1 \times \Big(\mathbf{c}\big((y,v),\epsilon,\boldsymbol{\infty}\big) + v^U(\boldsymbol{\infty})\Big) && [U(y,v)=\epsilon] \\
v^U(\boldsymbol{\infty}) &= 0 \qquad [\boldsymbol{\infty} \textit{ is a terminal vertex}]
\end{aligned}$$

Note that

$$\mathbf{c}\big((s,t),\omega,(x,u)\big) = \mathbf{c}\big((s,t),\rho,(y,v)\big) = \mathbf{c}\big((x,u),\epsilon,\boldsymbol{\infty}\big) = 0$$

and

$$\mathbf{c}\big((y,v),\epsilon,\boldsymbol{\infty}\big) = 1.$$

The values of the vertices under the policy U are the solution of the above equations shown in the table below.

(s,t)	(x,u)	(y,v)	$\boldsymbol{\infty}$
$\frac{1}{2}$	0	1	0

According to Theorem 5.1.11, we have

$$\boldsymbol{\mu}(\Phi) = \min_{T \in \mathcal{T}} v^T = \min\{v^W, v^U\} = v^U.$$

Thus, $\boldsymbol{\mu}(\Phi)(s,t) = v^U(s,t) = \frac{1}{2} = \boldsymbol{\mu}(\Delta)(s,t)$.

□

6 Policy Iteration for Labelled Markov Chains

We have shown in Section 5.1 that Howard's policy iteration algorithm can compute the optimal values for stopping MDPs. Also, we have shown in Section 5.3 that each labelled Markov chain can be mapped to a stopping MDP and the probabilistic bisimilarity distances of the labelled Markov chain correspond to the optimal values of the stopping MDP. We can use policy iteration algorithms to compute the optimal values of the corresponding stopping MDP, thus the probabilistic bisimilarity distances for the labelled Markov chain.

We define the set D_1 of state pairs with probabilistic bisimilarity distance one as

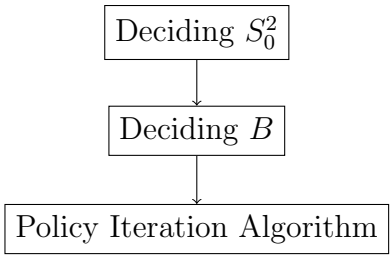
$$D_1 = \{\, (s,t) \in S^2 \mid \boldsymbol{\mu}(\Delta)(s,t) = 1 \,\}.$$

Let a set B be such that $S_1^2 \subseteq B \subseteq D_1$. We will prove that the policy iteration algorithm can compute the distances for the state pairs in $S_?^2 \setminus B$.

To compute all distances of a labelled Markov chain, the algorithm is as follows. It starts with computing S_0^2 (step 1). These state pairs have distance zero. The second step is to compute the set B. If we choose $B = S_1^2$, as we have discussed in

Section 4.2, it can be decided by comparing the labels of each state pair (step 2). The state pairs in B have distance one. Then it runs the policy iteration algorithm to compute the distances for the remaining state pairs in $S^2_? \setminus B$ (step 3).

The following diagram shows the decision procedure for distance zero, the set B and the policy iteration algorithm.



Note that the policy iteration algorithm can be either the simple policy iteration (Section 6.2) or the general policy iteration (Section 6.4).

The first two steps require polynomial time as discussed in Section 4.2 [vi]. We will see in Section 6.3 that the running time of the simple policy iteration can be exponential. Thus, the algorithm of computing all the distances, using simple policy iteration, has an exponential lower bound. As the complexity of the general policy iteration is unknown, the complexity of the algorithm of deciding all the distances, using general policy iteration, remains unknown as well.

The algorithms presented in this section are modifications of the basic algorithm of Bacci et al. [3]. Their algorithm can be viewed as a basic algorithm, enhanced

[vi] It is polynomial time to decide S^2_1 as shown in Section 4.2. It is also polynomial time to decide D_1, which will be discussed in detail in Chapter 8.

with an optimization. The basic algorithm corresponds to the policy iteration algorithm (step 3) and the key idea behind this optimization is to compute the distances "on the fly." Roughly speaking, to compute the distance of s and t we only need to compute the distance of u and v where s and t can reach u and v in n transitions for some $n > 0$. We will show in Theorem 6.1.9 that the basic algorithm of Bacci et al. [3], since it does not determine probabilistic bisimilarity first, does not always compute the distances correctly [vii].

6.1 An Alternative Characterization of $\boldsymbol{\mu}(\Delta)$

Before we proceed to the policy iteration algorithms, we provide an alternative characterization of the probabilistic bisimilarity pseudometric $\boldsymbol{\mu}(\Delta)$ in terms of the policy of the corresponding MDP. This alternative characterization generalizes the one presented by Chen et al. [23] which only considers the case $B = S_1^2$.

Since all the state pairs in S_1^2 have distance one, $S_1^2 \subseteq D_1$. Our characterization is parametric in a set B satisfying $S_1^2 \subseteq B \subseteq D_1$. The alternative characterization relies on couplings, a notion from the theory of Markov chains.

Definition 6.1.1. The set $\mathcal{C}_B$ of *couplings* of the labelled Markov chain $\langle S, L, \tau, \ell \rangle$ is defined by

$$\mathcal{C}_B = \{\, T \in \mathrm{Distr}(S^2)^{S_?^2 \setminus B} \mid \forall (s,t) \in S_?^2 \setminus B : T(s,t) \in \Omega(\tau(s), \tau(t)) \,\}.$$

[vii] In [5], as they only consider the pseudometric which is parametrized by a discount factor less than one, the basic algorithm is correct.

Just as we denote the vertices of $\Omega(\mu, \nu)$ for $\mu, \nu \in \mathrm{Distr}(S)$ as $V(\Omega(\mu, \nu))$, we define

$$V(\mathcal{C}_B) = \{\, T \in \mathrm{Distr}(S^2)^{S^2_? \setminus B} \mid \forall (s,t) \in S^2_? \setminus B : T(s,t) \in V(\Omega(\tau(s), \tau(t))) \,\}.$$

Recall that for each $(s,t) \in S^2_? \setminus B$, 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 $S^2_? \setminus B$ is finite as well. Therefore, the set $V(\mathcal{C}_B)$ contains only finite number of couplings. This fact is crucial in proving the termination of the policy iteration algorithms.

For the remainder of this section, we fix a labelled Markov chain $\langle S, L, \tau, \ell \rangle$. We also fix a coupling $T \in V(\mathcal{C}_B)$ of the labelled Markov chain. We have shown in Section 5.3 that the labelled Markov chain can be transformed into an MDP. The coupling T can then be viewed as a policy of the MDP, where a vertex in $S^2_? \setminus B$ is mapped to the action $T(s,t)$, a vertex in S^2_0 is mapped to the action ϵ and a vertex in B is mapped to any available action.

Next, we show that the values under the policy T of the MDP transformed from the labelled Markov chain can be characterized as the least fixed point of the following function.

Definition 6.1.2. The function $\Theta_B^T : [0,1]^{S^2} \to [0,1]^{S^2}$ is defined by

$$\Theta_B^T(d)(s,t) = \begin{cases} 0 & \text{if } (s,t) \in S_0^2 \\ 1 & \text{if } (s,t) \in B \\ \displaystyle\sum_{u,v \in S} T(s,t)(u,v)\, d(u,v) & \text{otherwise.} \end{cases}$$

Proposition 6.1.3. *For all $d, e \in [0,1]^{S^2}$, if $d \sqsubseteq e$ then $\Theta_B^T(d) \sqsubseteq \Theta_B^T(e)$.*

Proof. Let $d, e \in [0,1]^{S^2}$ with $d \sqsubseteq e$. Let $s, t \in S$. We distinguish three cases.

- If $(s,t) \in S_0^2$ then

$$\Theta_B^T(d)(s,t) = 0 = \Theta_B^T(e)(s,t).$$

- If $(s,t) \in B$ then

$$\Theta_B^T(d)(s,t) = 1 = \Theta_B^T(e)(s,t).$$

- Otherwise,

$$\begin{aligned} \Theta_B^T(d)(s,t) &= \sum_{u,v \in S} T(s,t)(u,v)\, d(u,v) \\ &\leq \sum_{u,v \in S} T(s,t)(u,v)\, e(u,v) \qquad [d \sqsubseteq e] \\ &= \Theta_B^T(e)(s,t). \end{aligned}$$

□

By Proposition 2.1.11, $[0,1]^{S^2}$ is a complete lattice. By Proposition 6.1.3, Θ_B^T is a monotone function. We can conclude from the Knaster-Tarski fixed point theorem

(Theorem 2.1.8(a)) that Θ_B^T has a least fixed point. We denote this fixed point by $\boldsymbol{\mu}(\Theta_B^T)$.

Θ_B^T is very similar to Θ^T of Definition 5.1.4. In fact, for all $(s,t) \in S^2$, we have $\boldsymbol{\mu}(\Theta_B^T)(s,t) = \boldsymbol{\mu}(\Theta^T)(s,t)$. This can be easily proved by showing that for any $v \in [0,1]^{S^2 \cup \{\infty\}}$, $\Theta^T(v)(s,t) = 0$ for any $(s,t) \in S_0^2$ and $\Theta^T(v)(s,t) = 1$ for any $(s,t) \in B$.

Next, we show that the distances of the labelled Markov chain can be characterized as the values of the coupled MDP under a policy. In particular, this policy is optimal for the coupled MDP since it leads to minimum values of all vertices. That is, the other policies will not lead to smaller values of vertices. The proof is very similar to that of Theorem 5.1.11.

Proposition 6.1.4. *For all $T \in V(\mathcal{C}_B)$, $\Delta(\boldsymbol{\mu}(\Theta_B^T)) \sqsubseteq \boldsymbol{\mu}(\Theta_B^T)$.*

Proof. Let s, $t \in S$. We distinguish three cases.

- If $(s,t) \in B$ then

$$\Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t) \leq 1 = \boldsymbol{\mu}(\Theta_B^T)(s,t).$$

- If $(s,t) \in S_0^2$, then $s \sim t$. According to Definition 2.1.5, we can conclude that $\ell(s) = \ell(t)$, and there exists an $\pi \in \Omega(\tau(s), \tau(t))$ such that

$$\text{support}(\pi) \subseteq \sim \tag{6.1}$$

$$\begin{aligned}
\Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t) &= \min_{\omega\in\Omega(\tau(s),\tau(t))} \sum_{(u,v)\in S^2} \omega(u,v)\,\boldsymbol{\mu}(\Theta_B^T)(u,v) \\
&\leq \sum_{(u,v)\in S^2} \pi(u,v)\,\boldsymbol{\mu}(\Theta_B^T)(u,v) \\
&= \sum_{(u,v)\in \mathrm{support}(\pi)} \pi(u,v)\,\boldsymbol{\mu}(\Theta_B^T)(u,v) \\
&= 0 \\
&\qquad [(6.1),\ \forall u\sim v : \boldsymbol{\mu}(\Theta_B^T)(u,v) = \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(u,v) = 0] \\
&= \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(s,t) \qquad [\text{since } s\sim t,\ \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(s,t) = 0] \\
&= \boldsymbol{\mu}(\Theta_B^T)(s,t).
\end{aligned}$$

- Otherwise,

$$\begin{aligned}
\Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t) &= \min_{\omega\in\Omega(\tau(s),\tau(t))} \sum_{u,v\in S} \omega(u,v)\,\boldsymbol{\mu}(\Theta_B^T)(u,v) \\
&\leq \sum_{u,v\in S} T(s,t)(u,v)\,\boldsymbol{\mu}(\Theta_B^T)(u,v) \qquad [T(s,t)\in\Omega(\tau(s),\tau(t))] \\
&= \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(s,t) \\
&= \boldsymbol{\mu}(\Theta_B^T)(s,t).
\end{aligned}$$

□

Corollary 6.1.5. *For all $T\in V(\mathcal{C}_B)$, $\boldsymbol{\mu}(\Delta)\sqsubseteq\boldsymbol{\mu}(\Theta_B^T)$.*

Proof. The corollary follows from Proposition 6.1.4 and Theorem 2.1.8(c). □

Proposition 6.1.6. *There exists a $T\in V(\mathcal{C}_B)$ such that $\boldsymbol{\mu}(\Theta_B^T)\sqsubseteq\boldsymbol{\mu}(\Delta)$.*

Proof. For each $(s,t) \in S^2_? \setminus B$, define

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

Obviously, $T \in V(\mathcal{C}_B)$.

Since $\boldsymbol{\mu}(\Theta^T_B)$ is the least fixed point of Θ^T_B, it suffices to show that $\boldsymbol{\mu}(\Delta)$ is a fixed point of Θ^T_B. Let $s,t \in S$. We distinguish three cases.

- If $(s,t) \in B$, then

$$\boldsymbol{\mu}(\Theta^T_B)(s,t) = \Theta^T_B(\boldsymbol{\mu}(\Theta^T_B))(s,t) = 1 = \boldsymbol{\mu}(\Delta)(s,t).$$

- If $(s,t) \in S^2_0$ then $s \sim t$. We have

$$\begin{aligned} \boldsymbol{\mu}(\Theta^T_B)(s,t) &= 0 \\ &= \boldsymbol{\mu}(\Delta)(s,t) \qquad \text{[Theorem 2.1.14]}. \end{aligned}$$

- Otherwise,

$$\begin{aligned} \Theta^T_B(\boldsymbol{\mu}(\Delta))(s,t) &= \sum_{u,v \in S} T(s,t)(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) \\ &= \min_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) \\ &\qquad \text{[by construction of } T\text{]} \\ &= \Delta(\boldsymbol{\mu}(\Delta))(s,t) \\ &= \boldsymbol{\mu}(\Delta)(s,t). \end{aligned}$$

□

The probabilistic bisimilarity pseudometric $\boldsymbol{\mu}(\Delta)$ can be characterized as follows.

Theorem 6.1.7. $\boldsymbol{\mu}(\Delta) = \min_{T \in V(\mathcal{C}_B)} \boldsymbol{\mu}(\Theta_B^T)$.

Proof. Immediate consequence of Proposition 6.1.5 and 6.1.6. □

Bacci, Bacci, Larsen and Mardare [3] put forward an algorithm to compute the probabilistic bisimilarity distances. Their algorithm can be viewed as a basic algorithm, enhanced with an optimization. Here, we focus on the basic algorithm. The optimization will be discussed in Chapter 7. Next, we will present the basic algorithm by Bacci et al.

Definition 6.1.8. Let $T \in \mathrm{Distr}(S^2)^{S^2 \setminus S_1^2}$ such that

$$\forall (s,t) \in S^2 \setminus S_1^2 : T(s,t) \in V(\Omega(\tau(s), \tau(t))). \tag{6.2}$$

The function $\Psi^T : [0,1]^{S^2} \to [0,1]^{S^2}$ is defined by

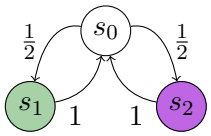
$$\Psi^T(d)(s,t) = \begin{cases} 1 & \text{if } (s,t) \in S_1^2 \\ \displaystyle\sum_{u,v \in S} T(s,t)(u,v)\, d(u,v) & \text{otherwise.} \end{cases}$$

The basic algorithm of Bacci et al. relies on the fact that if a coupling $T \in \mathrm{Distr}(S^2)^{S^2 \setminus S_1^2}$ satisfying (6.2) is locally optimal, that is, $\Delta(\boldsymbol{\mu}(\Psi^T)) = \boldsymbol{\mu}(\Psi^T)$, then it is globally optimal as well, that is, $\boldsymbol{\mu}(\Psi^T) = \boldsymbol{\mu}(\Delta)$ (see [3, Lemma 18]). However, as we will show next, this is not the case in general.

We denote the Dirac distribution concentrated at the pair of states (s,t) by $\mathrm{Dir}_{(s,t)}$, that is, $\mathrm{Dir}_{(s,t)}(u,v) = 1$ if $s = u$ and $t = v$ and $\mathrm{Dir}_{(s,t)}(u,v) = 0$ otherwise.

Theorem 6.1.9. *There exists a labelled Markov chain and* $T \in \mathrm{Distr}(S^2)^{S^2 \setminus S_1^2}$ *satisfying (6.2) such that* $\Delta(\boldsymbol{\mu}(\Psi^T)) = \boldsymbol{\mu}(\Psi^T)$ *and* $\boldsymbol{\mu}(\Psi^T) \neq \boldsymbol{\mu}(\Delta)$.

Proof. Consider the following labelled Markov chain.



Note that

$$V(\Omega(\tau(s_0), \tau(s_0)) = \{\tfrac{1}{2}\mathrm{Dir}_{(s_1,s_1)} + \tfrac{1}{2}\mathrm{Dir}_{(s_2,s_2)}, \tfrac{1}{2}\mathrm{Dir}_{(s_1,s_2)} + \tfrac{1}{2}\mathrm{Dir}_{(s_2,s_1)}\} \quad (6.3)$$

$$V(\Omega(\tau(s_1), \tau(s_1)) = \{\mathrm{Dir}_{(s_0,s_0)}\} \quad (6.4)$$

$$V(\Omega(\tau(s_2), \tau(s_2)) = \{\mathrm{Dir}_{(s_0,s_0)}\} \quad (6.5)$$

Now take $T \in \mathrm{Distr}(S^2)^{S^2 \setminus S_1^2}$ satisfying (6.2) such that

$$T(s_0, s_0) = \tfrac{1}{2}\mathrm{Dir}_{(s_1,s_2)} + \tfrac{1}{2}\mathrm{Dir}_{(s_2,s_1)} \quad (6.6)$$

$$(6.7)$$

First, let us consider the state pairs which have different labels, namely (s_0, s_1), (s_1, s_2), (s_0, s_2) and their counterparts. Let $(s, t) \in S_1^2$. Then

$$\boldsymbol{\mu}(\Psi^T)(s,t) = \Psi^T(\boldsymbol{\mu}(\Psi^T))(s,t)$$

$$= \quad 1 \qquad \text{[Definition 6.1.8]}. \tag{6.8}$$

$$\tag{6.9}$$

We consider the remaining state pairs which have the same labels, namely (s_0, s_0), (s_1, s_1), (s_2, s_2). We will determine their values under the coupling T as follows.

-

$$\begin{aligned}
\boldsymbol{\mu}(\Psi^T)(s_0, s_0) &= \Psi^T(\boldsymbol{\mu}(\Psi^T))(s_0, s_0) \\
&= \sum_{u,v \in S} T(s_0, t_0)(u, v)\, \boldsymbol{\mu}(\Psi^T)(u, v) \qquad \text{[Definition 6.1.8]} \\
&= \tfrac{1}{2} \times \boldsymbol{\mu}(\Psi^T)(s_1, s_2) + \tfrac{1}{2} \times \boldsymbol{\mu}(\Psi^T)(s_2, s_1) \qquad \text{[(6.6)]} \\
&= \tfrac{1}{2} \times 1 + \tfrac{1}{2} \times 1 \qquad \text{[}(s_1, s_2) \in S_1^2 \text{ and (6.8)]} \\
&= 1
\end{aligned} \tag{6.10}$$

-

$$\begin{aligned}
\boldsymbol{\mu}(\Psi^T)(s_1, s_1) &= \Psi^T(\boldsymbol{\mu}(\Psi^T))(s_1, s_1) \\
&= \sum_{u,v \in S} T(s_1, t_1)(u, v)\, \boldsymbol{\mu}(\Psi^T)(u, v) \qquad \text{[Definition 6.1.8]} \\
&= \boldsymbol{\mu}(\Psi^T)(s_0, s_0) \qquad \text{[(6.4)]} \\
&= 1 \qquad \text{[(6.10)]}
\end{aligned}$$

-

$$\begin{aligned}
\boldsymbol{\mu}(\Psi^T)(s_2, s_2) &= \Psi^T(\boldsymbol{\mu}(\Psi^T))(s_2, s_2) \\
&= \sum_{u,v \in S} T(s_2, t_2)(u, v)\, \boldsymbol{\mu}(\Psi^T)(u, v) \qquad \text{[Definition 6.1.8]} \\
&= \boldsymbol{\mu}(\Psi^T)(s_0, s_0) \qquad \text{[(6.5)]} \\
&= 1 \qquad \text{[(6.10)]}
\end{aligned}$$

Thus, $\boldsymbol{\mu}(\Psi^T) = \mathbf{1}$.

Next, we will show $\Delta(\boldsymbol{\mu}(\Psi^T)) = \boldsymbol{\mu}(\Psi^T)$. Thus, if the algorithm by Bacci et al. [3] starts with the coupling T, it terminates with $\boldsymbol{\mu}(\Psi^T) = \mathbf{1}$.

We show that $\Delta(\boldsymbol{\mu}(\Psi^T)) = \boldsymbol{\mu}(\Psi^T)$ next. Let $(s, t) \in S_1^2$. Then $\Delta(\boldsymbol{\mu}(\Psi^T))(s, t) = 1$ by Definition 2.1.7. The remaining state pairs (s_0, s_0), (s_1, s_1), (s_2, s_2) are considered as follows.

$$\begin{aligned}
\Delta(\boldsymbol{\mu}(\Psi^T))(s_0, t_0) &= \min\big\{\tfrac{1}{2}\boldsymbol{\mu}(\Psi^T)(s_1, t_2) + \tfrac{1}{2}\boldsymbol{\mu}(\Psi^T)(s_2, t_1), \\
&\qquad\qquad \tfrac{1}{2}\boldsymbol{\mu}(\Psi^T)(s_1, t_1) + \tfrac{1}{2}\boldsymbol{\mu}(\Psi^T)(s_2, t_2)\big\} \\
&= \min\big\{1, 1\big\} = 1 \\
\Delta(\boldsymbol{\mu}(\Psi^T))(s_1, t_1) &= \boldsymbol{\mu}(\Psi^T)(s_0, t_0) = 1 \\
\Delta(\boldsymbol{\mu}(\Psi^T))(s_2, t_2) &= \boldsymbol{\mu}(\Psi^T)(s_0, t_0) = 1
\end{aligned}$$

Finally, we prove that the probabilistic bisimilarity distance of (s_0, s_0), (s_1, s_1), (s_2, s_2) is zero. We define $d \in [0, 1]^{S^2}$ by

$$d(u, v) = \begin{cases} 1 & \text{if } (u, v) \in S_1^2 \\ 0 & \text{otherwise} \end{cases}$$

We will show that $\boldsymbol{\mu}(\Delta) = d$. We first show that $\boldsymbol{\mu}(\Delta)(s,t) = 1$ for all $(s,t) \in S_1^2$. Let $(s,t) \in S_1^2$. By Definition 2.1.7,

$$\boldsymbol{\mu}(\Delta)(s,t) = \Delta(\boldsymbol{\mu}(\Delta))(s,t) = 1.$$

Since d is the smallest distance function such that $d(s,t) = 1$ for all $(s,t) \in S_1^2$, we are left to show that d is a fixed point of Δ.

We consider the remaining state pairs (s_0, s_0), (s_1, s_1), (s_2, s_2) as follows.

-

$$\begin{aligned}
\Delta(d)(s_0,t_0) &= \min_{\omega \in \Omega(\tau(s_0),\tau(t_0))} \sum_{u,v \in S} \omega(u,v)\, d(u,v) \qquad \text{[Definition 2.1.7]} \\
&= \min\{\tfrac{1}{2} \times d(s_1,s_2) + \tfrac{1}{2} \times d(s_2,s_1), \tfrac{1}{2} \times d(s_1,s_1) + \tfrac{1}{2} \times d(s_2,s_2)\} \\
&\qquad [(6.3)] \\
&= \min\{\tfrac{1}{2} \times 1 + \tfrac{1}{2} \times 1, \tfrac{1}{2} \times 0 + \tfrac{1}{2} \times 0\} \\
&= \min\{1, 0\} \\
&= 0 \qquad (6.11)
\end{aligned}$$

-

$$\begin{aligned}
\Delta(d)(s_1,s_1) &= \min_{\omega \in \Omega(\tau(s_1),\tau(t_1))} \sum_{u,v \in S} \omega(u,v) d(u,v) \qquad \text{[Definition 2.1.7]} \\
&= d(s_0,s_0) \qquad [(6.4)] \\
&= 0 \qquad [(6.11)]
\end{aligned}$$

-

$$\begin{aligned}\Delta(d)(s_2, s_2) &= \min_{\omega \in \Omega(\tau(s_2), \tau(t_2))} \sum_{u,v \in S} \omega(u,v) d(u,v) \qquad \text{[Definition 2.1.7]} \\ &= d(s_0, s_0) \qquad [(6.5)] \\ &= 0 \qquad [(6.11)]\end{aligned}$$

Thus, $\Delta(d) = d$ and $\boldsymbol{\mu}(\Delta) = d \neq \boldsymbol{\mu}(\Psi^T)$.

□

6.2 Simple Policy Iteration

In this section we present the simple policy iteration algorithm to compute the distances for labelled Markov chains. Recall that a policy of the player maps each vertex to one of its available actions. That is, such a policy maps (s,t) with $(s,t) \in S_?^2 \setminus B$ to a coupling in $V(\Omega(\tau(s), \tau(t)))$.

The algorithm starts with an arbitrary policy, that is, an arbitrary $T \in V(\mathcal{C}_B)$ (see line 1). As long as there is a vertex which is not locally optimal with respect to the current policy, the policy at that vertex is improved to a locally optimal choice. Note that a vertex (s,t) is not locally optimal if there exists a different choice for that vertex, that is, $\omega \in V(\Omega(\tau(s), \tau(t))$, so that the value of the vertex decreases. This is captured on line 2. On line 3, we compute a locally optimal choice and update the policy.

1 $T \leftarrow$ an element of $V(\mathcal{C}_B)$

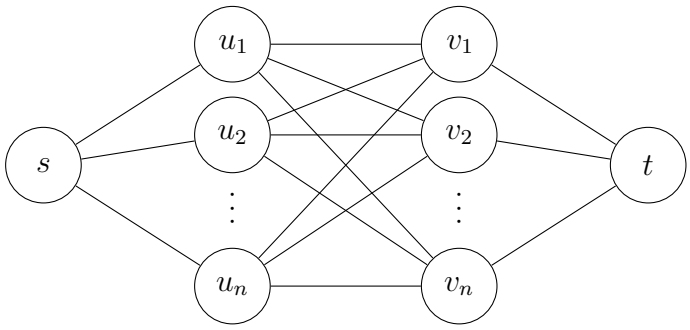
2 while $\exists (s,t) \in S_?^2 \setminus B : \boldsymbol{\mu}(\Theta_B^T)(s,t) > \Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t)$

3 $T(s,t) \leftarrow \underset{\omega \in V(\Omega(\tau(s),\tau(t)))}{\operatorname{argmin}} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta_B^T)(u,v)$

This is our modification of the basic algorithm of Bacci et al. The main difference is that we compute the states which are probabilistic bisimilar before running the simple policy iteration algorithm. We use Θ_B^T of Definition 6.1.2 which assigns the probabilistic bisimilar state pairs to zero, while they use Ψ^T of Definition 6.1.8.

On line 1, an initial policy $T \in V(\mathcal{C}_B)$ can be computed by the North-West corner method in polynomial time (see, for example, [83, page 180]). On line 2, rather than choosing an arbitrary vertex that is not locally optimal, in the simple policy iteration a select procedure can be defined as follows: we number all the vertices in $S_?^2 \setminus B$ and select the one with the highest number (see, for example, [67, Section 2]). Note that $\boldsymbol{\mu}(\Theta_B^T)$ can be computed in polynomial time (see, for example, [8, Section 10.1.1]). On line 3, the computation can be viewed as a minimum-cost flow problem, where s is the source vertex and t is the sink vertex. Below we present the flow network. The sets $\{u_1, \cdots, u_n\}$ and $\{v_1, \cdots, v_n\}$ are copies of S. For the edge (s, u_i) and (v_j, t), the capacity is $\tau(s)(u_i)$ and $\tau(t)(v_j)$, respectively. There is no cost transporting along these edges. Each edge $(u_i, v_j) \in S^2$ has a capacity of $\min(\tau(s)(u_i), \tau(t)(v_j))$ and $\boldsymbol{\mu}(\Theta_B^T)(u_i, v_j)$ is the cost of edge (u_i, v_j). The minimum cost of transporting one unit from s to t is captured by $\Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t)$, which can be solved using the network simplex algorithm in strongly polynomial time [2, Section

11.8]. Thus, each line of the algorithm is in polynomial time and the running time of the algorithm is determined by the number of iterations of the while loop.



Next, 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 $\boldsymbol{\mu}(\Theta_B^T)$ captures $\boldsymbol{\mu}(\Delta)$. We first introduce a new function $\Delta_B : [0,1]^{S^2} \to [0,1]^{S^2}$ as follows.

Definition 6.2.1. The function $\Delta_B : [0,1]^{S^2} \to [0,1]^{S^2}$ is defined by

$$\Delta_B(d)(s,t) = \begin{cases} 0 & \text{if } (s,t) \in S_0^2 \\ 1 & \text{if } (s,t) \in B \\ \Delta(d)(s,t) & \text{otherwise} \end{cases}$$

The following theorem collects some properties of Δ_B that we will use later.

Theorem 6.2.2. *(a) The function Δ_B is monotone.*

(b) The function Δ_B is nonexpansive.

(c) $\boldsymbol{\mu}(\Delta_B) = \boldsymbol{\nu}(\Delta_B)$.

(d) $\boldsymbol{\mu}(\Delta_B) = \boldsymbol{\mu}(\Delta)$.

(e) $\boldsymbol{\mu}(\Delta_B) = \sup_{m \in \mathbb{N}} \Delta_B^m(\mathbf{0})$.

Proof. (a) Since Δ is monotone (Proposition 2.1.13), we can easily deduce that Δ_B is monotone as well.

(b) Since Δ is nonexpansive (Proposition 2.1.26), we can easily deduce that Δ_B is nonexpansive as well.

(c) The proof of this part is very similar to [23, Proposition 17]. Since Δ_B is monotone according to part (a), we can conclude from the Knaster-Tarski fixed point theorem (Theorem 2.1.8(a, b)) that Δ_B has a least fixed point $\boldsymbol{\mu}(\Delta_B)$ and a greatest fixed point $\boldsymbol{\nu}(\Delta_B)$.

To conclude that $\boldsymbol{\mu}(\Delta_B) = \boldsymbol{\nu}(\Delta_B)$, let

$$m = \max\{\, \boldsymbol{\nu}(\Delta_B)(s,t) - \boldsymbol{\mu}(\Delta_B)(s,t) \mid s,t \in S \,\}$$

and

$$M = \{\, (s,t) \in S^2 \mid \boldsymbol{\nu}(\Delta_B)(s,t) - \boldsymbol{\mu}(\Delta_B)(s,t) = m \,\}.$$

We will show that $m = 0$, which implies that $\boldsymbol{\mu}(\Delta_B) = \boldsymbol{\nu}(\Delta_B)$. We distinguish three cases.

- Assume that $M \cap S_0^2 \neq \emptyset$. Let $(s,t) \in M \cap S_0^2$. Then

$$\boldsymbol{\nu}(\Delta_B)(s,t) - \boldsymbol{\mu}(\Delta_B)(s,t)$$

$$
\begin{aligned}
&= \Delta_B(\boldsymbol{\nu}(\Delta_B))(s,t) - \Delta_B(\boldsymbol{\mu}(\Delta_B))(s,t) \\
&= 0 - 0 \\
&= 0
\end{aligned}
$$

and, hence, $m = 0$.

– Assume that $M \cap B \neq \emptyset$. Let $(s,t) \in M \cap B$. Then

$$
\begin{aligned}
&\boldsymbol{\nu}(\Delta_B)(s,t) - \boldsymbol{\mu}(\Delta_B)(s,t) \\
&= \Delta_B(\boldsymbol{\nu}(\Delta_B))(s,t) - \Delta_B(\boldsymbol{\mu}(\Delta_B))(s,t) \\
&= 1 - 1 \\
&= 0
\end{aligned}
$$

and, hence, $m = 0$.

– Otherwise, assume that $M \cap (S_0^2 \cup B) = \emptyset$. We will show that this leads to a contradiction, that is, this case is vacuous. Next, we will prove that M is a probabilistic bisimulation and, hence, $M \subseteq S_0^2$, which contradicts $M \cap (S_0^2 \cup B) = \emptyset$.

Let $(s,t) \in M$. Since $M \cap (S_0^2 \cup B) = \emptyset$, for all $(s,t) \in M$ we have that $(s,t) \notin B$ and, hence, $(s,t) \notin S_1^2$. Therefore, $\ell(s) = \ell(t)$.

According to Definition 2.1.4 and Definition 2.1.5, it remains to show that there exists $\pi \in \Omega(\tau(s), \tau(t))$ such that support$(\pi) \subseteq M$, that is, $\pi(u,v) > 0$ implies $(u,v) \in M$ and, hence, $\boldsymbol{\nu}(\Delta_B)(u,v) - \boldsymbol{\mu}(\Delta_B)(u,v) = m$.

Suppose

$$\Delta_B(\boldsymbol{\mu}(\Delta_B))(s,t) = \sum_{u,v\in S} \pi(u,v)\, \boldsymbol{\mu}(\Delta_B)(u,v),$$

where $\pi \in \Omega(\tau(s),\tau(t))$. Then

$$\begin{aligned} m &= \boldsymbol{\nu}(\Delta_B)(s,t) - \boldsymbol{\mu}(\Delta_B)(s,t) \\ &= \Delta_B(\boldsymbol{\nu}(\Delta_B))(s,t) - \Delta_B(\boldsymbol{\mu}(\Delta_B))(s,t) \\ &= \left(\min_{\omega\in\Omega(\tau(s),\tau(t))} \sum_{u,v\in S} \omega(u,v)\, \boldsymbol{\nu}(\Delta_B)(u,v) \right) - \sum_{u,v\in S} \pi(u,v)\, \boldsymbol{\mu}(\Delta_B)(u,v) \\ &\leq \sum_{u,v\in S} \pi(u,v)\, \boldsymbol{\nu}(\Delta_B)(u,v) - \sum_{u,v\in S} \pi(u,v)\, \boldsymbol{\mu}(\Delta_B)(u,v) \\ &= \sum_{u,v\in S} \pi(u,v)(\boldsymbol{\nu}(\Delta_B)(u,v) - \boldsymbol{\mu}(\Delta_B)(u,v)). \end{aligned}$$

Since $\boldsymbol{\nu}(\Delta_B)(u,v) - \boldsymbol{\mu}(\Delta_B)(u,v) \leq m$ and $\sum_{u,v\in S} \pi(u,v) = 1$, we can conclude from $\sum_{u,v\in S} \pi(u,v)(\boldsymbol{\nu}(\Delta_B)(u,v) - \boldsymbol{\mu}(\Delta_B)(u,v)) \geq m$ that $\pi(u,v) > 0$ implies $\boldsymbol{\nu}(\Delta_B)(u,v) - \boldsymbol{\mu}(\Delta_B)(u,v) = m$.

(d) The proof of this part is very similar to [23, Corollary 18]. From part (c), we can conclude that it suffices to prove that $\boldsymbol{\mu}(\Delta)$ is a fixed point of Δ_B. We distinguish the following three cases.

– If $(s,t) \in S_0^2$ then

$$\Delta_B(\boldsymbol{\mu}(\Delta)) = 0 = \boldsymbol{\mu}(\Delta)(s,t)$$

by Theorem 2.1.14.

– If $(s,t) \in B$ then

$$\Delta_B(\boldsymbol{\mu}(\Delta)) = 1 = \boldsymbol{\mu}(\Delta)(s,t)$$

since $B \subseteq D_1$.

– Otherwise,

$$\Delta_B(\boldsymbol{\mu}(\Delta)) = \Delta(\boldsymbol{\mu}(\Delta)) = \boldsymbol{\mu}(\Delta)(s,t).$$

(e) Since Δ_B is monotone (part (a)) and nonexpansive (part (b)), we can conclude from Theorem 2.1.21 that $\boldsymbol{\mu}(\Delta_B) = \sup_{n\in\mathbb{N}} \Delta_B^n(\mathbf{0})$.

□

Theorem 6.2.3. *For all $T \in V(\mathcal{C}_B)$, if $\boldsymbol{\mu}(\Theta_B^T)(s,t) \leq \Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t)$ for all $(s,t) \in S_?^2 \setminus B$, then $\boldsymbol{\mu}(\Theta_B^T) = \boldsymbol{\mu}(\Delta)$.*

Proof. Let $T \in V(\mathcal{C}_B)$. Assume that

$$\boldsymbol{\mu}(\Theta_B^T)(s,t) \leq \Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t) \tag{6.12}$$

for all $(s,t) \in S_?^2 \setminus B$. Let s, $t \in S$.

We have Δ_B has unique fixed point according to Theorem 6.2.2(c) and $\boldsymbol{\mu}(\Delta_B) = \boldsymbol{\mu}(\Delta)$ according to Theorem 6.2.2(d). It suffices to show that $\boldsymbol{\mu}(\Theta_B^T)$ is a fixed point of Δ_B.

By Proposition 6.1.4, $\Delta(\boldsymbol{\mu}(\Theta_B^T)) \sqsubseteq \boldsymbol{\mu}(\Theta_B^T)$. Thus,

$$\Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t) \leq \boldsymbol{\mu}(\Theta_B^T)(s,t) \tag{6.13}$$

It remains to show that $\Delta_B(\boldsymbol{\mu}(\Theta_B^T))(s,t) = \boldsymbol{\mu}(\Theta_B^T)(s,t)$. We distinguish the following three cases.

- If $(s,t) \in B$ then

$$\boldsymbol{\mu}(\Theta_B^T)(s,t) = \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(s,t) = 1 = \Delta_B(\boldsymbol{\mu}(\Theta_B^T))(s,t).$$

- If $(s,t) \in S_0^2$ then

$$\boldsymbol{\mu}(\Theta_B^T)(s,t) = \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(s,t) = 0 = \Delta_B(\boldsymbol{\mu}(\Theta_B^T))(s,t).$$

- Otherwise, $(s,t) \in S_?^2 \setminus B$. We have

$$\begin{aligned} \boldsymbol{\mu}(\Theta_B^T)(s,t) &= \Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t) && \text{[(6.12) and (6.13)]} \\ &= \Delta_B(\boldsymbol{\mu}(\Theta_B^T))(s,t) && \text{[Definition 6.2.1]} \end{aligned}$$

□

As we already mentioned earlier, the set $V(\mathcal{C}_B)$ is finite. To prove that the loop terminates we show that T becomes smaller in every iteration.

Definition 6.2.4. The order $\prec$ on $V(\mathcal{C}_B)$ is defined by $T \prec U$ if $\boldsymbol{\mu}(\Theta_B^T) \sqsubset \boldsymbol{\mu}(\Theta_B^U)$.

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 6.2.5. Let $T \in V(\mathcal{C}_B)$, $(s,t) \in S_?^2 \setminus B$ and $\omega \in V(\Omega(\tau(s), \tau(t)))$. The function $T[(s,t)/\omega] : (S_?^2 \setminus B) \to \mathrm{Distr}(S^2)$ 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, $T[(s,t)/\omega] \in V(\mathcal{C}_B)$. Next, we show that T indeed becomes smaller in every iteration of the loop and, as a consequence, the loop terminates.

Lemma 6.2.6. *For all T, $U \in V(\mathcal{C}_B)$, if $\Theta_B^U(\boldsymbol{\mu}(\Theta_B^T)) \sqsubset \boldsymbol{\mu}(\Theta_B^T)$ then $\boldsymbol{\mu}(\Theta_B^U) \sqsubset \boldsymbol{\mu}(\Theta_B^T)$.*

Proof. Let T, $U \in V(\mathcal{C}_B)$. Assume that $\Theta_B^U(\boldsymbol{\mu}(\Theta_B^T)) \sqsubset \boldsymbol{\mu}(\Theta_B^T)$. Then $\Theta_B^U(\boldsymbol{\mu}(\Theta_B^T)) \sqsubseteq \boldsymbol{\mu}(\Theta_B^T)$, that is, $\boldsymbol{\mu}(\Theta_B^T)$ is a pre-fixed point of Θ_B^U. From the Knaster-Tarski fixed point theorem (Theorem 2.1.8(c)) we can conclude that $\boldsymbol{\mu}(\Theta_B^U) \sqsubseteq \boldsymbol{\mu}(\Theta_B^T)$. It remains to show that $\boldsymbol{\mu}(\Theta_B^U) \neq \boldsymbol{\mu}(\Theta_B^T)$. Towards a contradiction, assume that $\boldsymbol{\mu}(\Theta_B^U) = \boldsymbol{\mu}(\Theta_B^T)$. Then

$$\Theta_B^U(\boldsymbol{\mu}(\Theta_B^T)) = \Theta_B^U(\boldsymbol{\mu}(\Theta_B^U)) = \boldsymbol{\mu}(\Theta_B^U) = \boldsymbol{\mu}(\Theta_B^T),$$

which contradicts the assumption $\Theta_B^U(\boldsymbol{\mu}(\Theta_B^T)) \sqsubset \boldsymbol{\mu}(\Theta_B^T)$. □

Theorem 6.2.7. *For all $T \in V(\mathcal{C}_B)$ and $(s,t) \in S_?^2 \setminus B$, if $\Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t) < \boldsymbol{\mu}(\Theta_B^T)(s,t)$, then $T[(s,t)/\pi] \prec T$, where $\pi = \operatorname{argmin}_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta_B^T)(u,v)$.*

Proof. Let $T \in V(\mathcal{C}_B)$ and $(s,t) \in S_?^2 \setminus B$. Assume that $\Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t) < \boldsymbol{\mu}(\Theta_B^T)(s,t)$. By Lemma 6.2.6, it remains to prove that $\Theta_B^{T[(s,t)/\pi]}(\boldsymbol{\mu}(\Theta_B^T)) \sqsubset \boldsymbol{\mu}(\Theta_B^T)$, where

$$\pi = \operatorname*{argmin}_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta_B^T)(u,v).$$

Let x, $y \in S$. We distinguish the following cases.

- Assume $(x, y) = (s, t)$. Then since $(s, t) \in S_?^2 \setminus B$,

$$\begin{aligned}
\Theta_B^{T[(s,t)/\pi]}(\boldsymbol{\mu}(\Theta_B^T))(x, y) &= \sum_{u,v \in S} T[(s, t)/\pi](x, y)(u, v)\ \boldsymbol{\mu}(\Theta_B^T)(u, v) \\
&= \sum_{u,v \in S} \pi(u, v)\ \boldsymbol{\mu}(\Theta_B^T)(u, v) \qquad [(x, y) = (s, t)] \\
&= \min_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{u,v \in S} \omega(u, v)\ \boldsymbol{\mu}(\Theta_B^T)(u, v) \\
&= \Delta(\boldsymbol{\mu}(\Theta_B^T))(s, t) \\
&< \boldsymbol{\mu}(\Theta_B^T)(s, t) \\
&= \boldsymbol{\mu}(\Theta_B^T)(x, y).
\end{aligned}$$

- Assume that $(x, y) \in S_0^2$. Then

$$\Theta_B^{T[(s,t)/\pi]}(\boldsymbol{\mu}(\Theta_B^T))(x, y) = 0 = \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(x, y) = \boldsymbol{\mu}(\Theta_B^T)(x, y).$$

- Assume that $(x, y) \in B$. Then

$$\Theta_B^{T[(s,t)/\pi]}(\boldsymbol{\mu}(\Theta_B^T))(x, y) = 1 = \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(x, y) = \boldsymbol{\mu}(\Theta_B^T)(x, y).$$

- Otherwise,

$$\begin{aligned}
&\Theta_B^{T[(s,t)/\pi]}(\boldsymbol{\mu}(\Theta_B^T))(x, y) \\
&= \sum_{u,v \in S} T[(s, t)/\pi](x, y)(u, v)\ \boldsymbol{\mu}(\Theta_B^T)(u, v) \\
&= \sum_{u,v \in S} T(x, y)(u, v)\ \boldsymbol{\mu}(\Theta_B^T)(u, v) \qquad [(x, y) \neq (s, t)]
\end{aligned}$$

$$
\begin{aligned}
&= \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(x,y) \\
&= \boldsymbol{\mu}(\Theta_B^T)(x,y).
\end{aligned}
$$

□

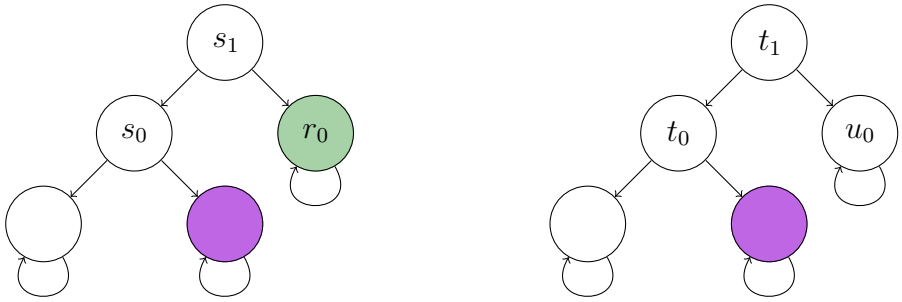
6.3 An Exponential Lower Bound of Simple Policy Iteration

Below, we will show that in the worst case, our algorithm takes exponential time. Many similar lower bounds have been proved for closely related algorithms by showing that the algorithms can be viewed as binary counters. We refer the reader to, for example, the thesis of Friedmann [40] for several such proofs. In Section 5.3, we have presented a transformation mapping a labelled Markov chain to an MDP, which is equivalent to a simple stochastic game with only one player. For simple stochastic games, Melekopoglou and Condon [67] showed that simple policy iteration takes exponential time in the worst case. We cannot directly use their result since no labelled Markov chain maps to the simple stochastic games they use in their proof.

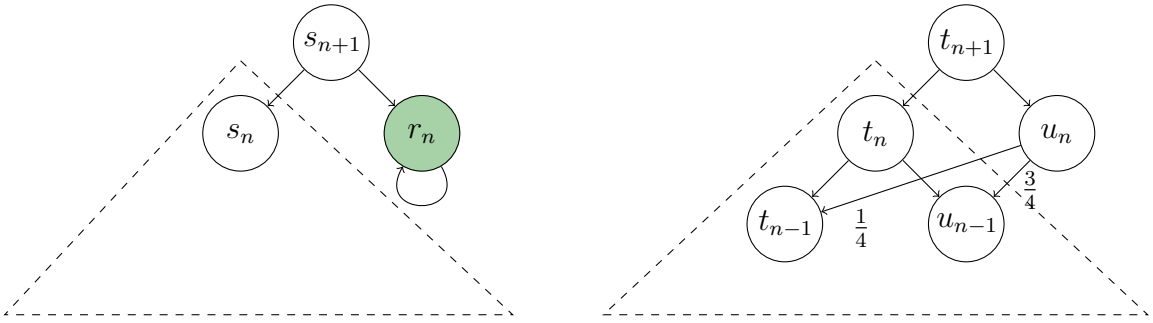
For each $n \in \mathbb{N}$ we will construct a labelled Markov chain of size $O(n)$. Furthermore, we will show that our simple policy iteration algorithm takes $\Omega(2^n)$ iterations for the resulting MDP. From now on, in diagrams, like the one below, only if the probabilities of the outgoing transitions of a state are not all the same, as is the

case for state u_n, we denote the actual probabilities.

Definition 6.3.1. 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



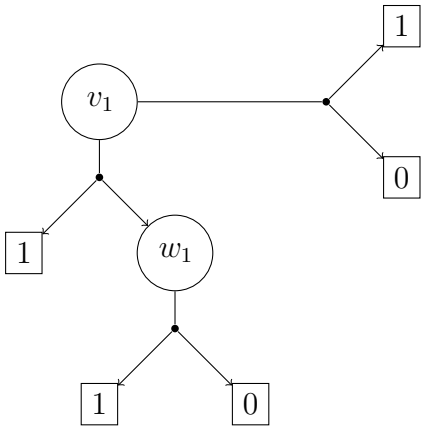
where the two dashed triangles together represent the labelled Markov chain $\mathcal{M}_{n-1}$.

Note that $\mathcal{M}_n$ has $4n + 10$ states and $7n + 14$ transitions and, hence, is of size $O(n)$. Next, we give the MDP corresponding to the above defined labelled Markov chains according to the transformation presented in Definition 5.3.1.

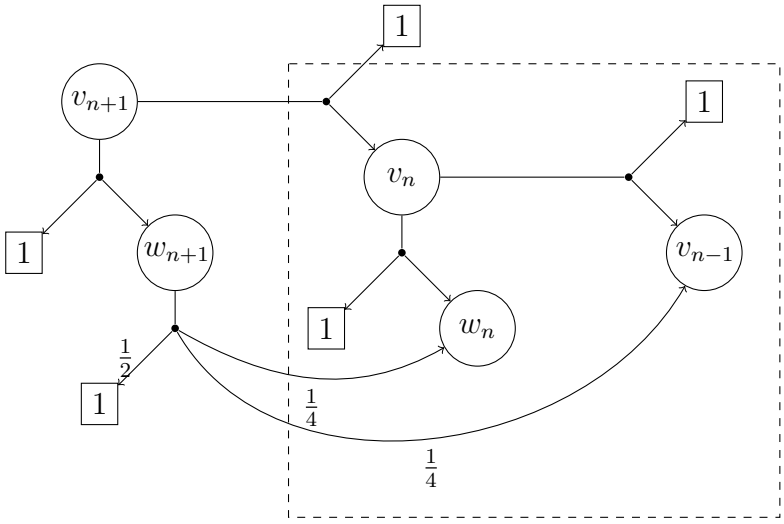
For the labelled Markov chains in Definition 6.3.1, the state pairs which have distance one are the ones with different labels, that is, $D_1 = S_1^2$. The optimal values

of vertices in S_0^2 and B are zero and one, respectively. From now on, to simplify the MDP we denote a vertex in S_0^2 as a square labelled with zero and a vertex in B as a square labelled with one.

Definition 6.3.2. For $n \in \mathbb{N}$, the MDP $\mathcal{G}_n$ is defined as follows by induction on n. The MDP $\mathcal{G}_0$ is defined as

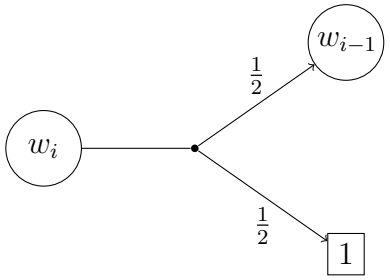


If $n > 0$ then the MDP $\mathcal{G}_n$ is defined as



where the dashed rectangle represents the MDP $\mathcal{G}_{n-1}$.

In the above definition, we use v_i to denote the vertex (s_i, t_i) and w_i to denote the vertex (s_{i-1}, u_{i-1}). Since $s_0 \sim t_0$, $\boldsymbol{\mu}(\Delta)(s_0, t_0) = 0$ and we use a square labelled with zero to denote v_0. The vertex w_i for $2 \leq i \leq n+1$ has lower priority than v_1, that is, w_i can be switched only if none of the vertex v_j for $1 \leq j \leq n+1$ is switchable. The transformation given in Definition 5.3.1 applied to labelled Markov chain $\mathcal{M}_n$ gives rise to an MDP of which $\mathcal{G}_n$ is only a part. In particular, for $2 \leq i \leq n+1$ an action and the following edges have not been included in $\mathcal{G}_n$, they are never selected in any of the policies we construct in our proofs.



Next, we consider the policies for the MDP $\mathcal{G}_n$. Recall that the Dirac distribution Dir_v is defined by $\mathrm{Dir}_v(w) = 1$ if $w = v$ and $\mathrm{Dir}_v(w) = 0$ otherwise. In order to avoid clutter, for $1 \leq i \leq n+1$, instead of $T(v_i) = \frac{1}{2}\mathrm{Dir}_1 + \frac{1}{2}\mathrm{Dir}_{v_{i-1}}$ we write $T(i) = 0$ and instead of $T(v_i) = \frac{1}{2}\mathrm{Dir}_1 + \frac{1}{2}\mathrm{Dir}_{w_i}$ we write $T(i) = 1$.

A vertex is switchable if it is not locally optimal.

Definition 6.3.3. The vertex v_i is switchable with respect to T if $\boldsymbol{\mu}(\Theta_B^T)(v_i) >$

$\boldsymbol{\mu}(\Theta_B^{\overline{T}_i})(v_i)$, where

$$\overline{T}_i(j) = \begin{cases} 1 - T(j) & \text{if } j = i \\ T(j) & \text{otherwise.} \end{cases}$$

Rather than starting from an arbitrary policy, we pick a specific initial policy (line 1–3). Furthermore, rather than choosing an arbitrary vertex that is not locally optimal, we pick the v_i which is not locally optimal with the largest index (line 5).

1 $T(1) \leftarrow 1$

2 for $i = 2, \ldots, n+1$

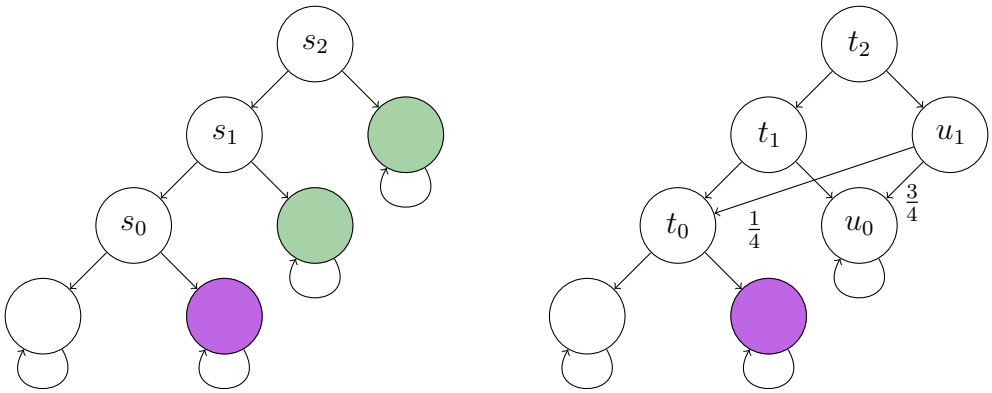
3 $\quad T(i) \leftarrow 0$

4 while $\exists 1 \leq i \leq n+1 : v_i$ is switchable with respect to T

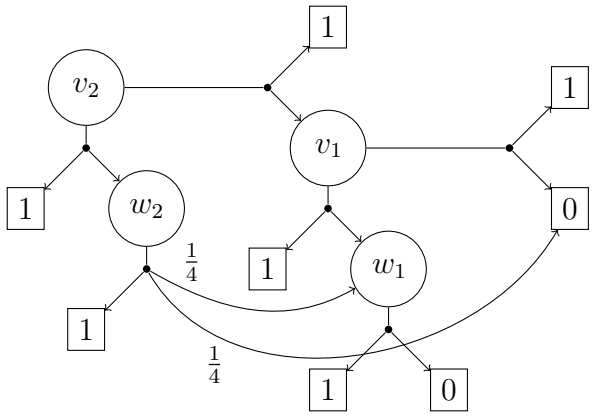
5 $\quad m \leftarrow \max\{\, i \mid v_i$ is switchable with respect to $T \,\}$

6 $\quad T(m) \leftarrow 1 - T(m)$

Example 6.3.4. *In this example, we consider the case where $n = 1$. The labelled Markov chain $\mathcal{M}_1$ is shown below.*



Below we present (part of) the MDP $\mathcal{G}_1$ corresponding to the labelled Markov chain $\mathcal{M}_1$.



For the above MDP, a policy of the player consists of either going to the right (represented by 0) or down (represented by 1) in the vertices v_2 and v_1. The simple policy iteration algorithm starts with policy 01*. That is, for the vertices v_2, v_1, the initial policy chooses the actions represented by the right edge and down edge, respectively. In the table below, we present for each policy the values of the vertices v_2, v_1. Going from one column to the next, the policy for either v_2, v_1 is switched. As a result, none of the values increase and one of the values decreases. The table contains all four 2-bit combinations and, hence, the MDP can be viewed as a 2-bit counter.*

	v_2	*0*	*1*	*1*	*0*
policy	v_1	*1*	*1*	*0*	*0*
	v_2	$\frac{7}{8}$	$\frac{13}{16}$	$\frac{13}{16}$	$\frac{3}{4}$
value	v_1	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{1}{2}$

□

Next, we show that the above simple policy iteration algorithm applied to the MDP $\mathcal{G}_n$ gives rise to exponentially many iterations.

In Proposition 6.3.18, we show that if the initial policy satisfies $T(n+1) = 0, \cdots, T(2) = 0, T(1) = 1$, then all the $n+1$ vertices are switchable. Moreover, by Proposition 6.3.17, the next $2^{n+1} - 1$ switches of the policy iteration algorithm are made on these $n+1$ vertices to reach the policy $T(n+1) = 0, \cdots, T(0) = 0$.

To prove these main propositions, we need to express the value of every vertex as a formula that depends on the current policy and the values of the other vertices. We start with a simple proposition.

It is immediate to observe that the value of w_1 is $\frac{1}{2}$. Also, the value of v_k can be expressed in terms of the current policy at v_k and the values of the neighbour vertices w_k and v_{k-1}.

Proposition 6.3.5.

(a) $\boldsymbol{\mu}(\Theta_B^T)(w_1) = \frac{1}{2}$.

(b) For all $1 \leq k \leq n+1$, $\boldsymbol{\mu}(\Theta_B^T)(v_k) = \frac{1}{2}(T(k)(1 + \boldsymbol{\mu}(\Theta_B^T)(w_k)) + (1 - T(k))(1 + \boldsymbol{\mu}(\Theta_B^T)(v_{k-1})))$.

Proof.

(a) Immediate.

(b) We distinguish two cases.

- If $T(k) = 0$ then $\boldsymbol{\mu}(\Theta_B^T)(v_k) = \frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_{k-1}) = \frac{1}{2}(T(k)(1+\boldsymbol{\mu}(\Theta_B^T)(w_k)) + (1 - T(k))(1 + \boldsymbol{\mu}(\Theta_B^T)(v_{k-1})))$.
- If $T(k) = 1$ then $\boldsymbol{\mu}(\Theta_B^T)(v_k) = \frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(w_k) = \frac{1}{2}(T(k)(1+\boldsymbol{\mu}(\Theta_B^T)(w_k)) + (1 - T(k))(1 + \boldsymbol{\mu}(\Theta_B^T)(v_{k-1})))$.

□

We define a helper function as follows.

Definition 6.3.6. For all $1 \leq k \leq n+1$, $a^T(k)$ is defined by

$$a^T(k) = \begin{cases} \frac{1}{4} & \text{if } k = 1 \\ \frac{1}{2}a^T(k-1)(\frac{1}{2} - T(k-1)) & \text{otherwise.} \end{cases}$$

The next proposition is technical and is only used in the proof of Proposition 6.3.8.

Proposition 6.3.7. *For all* $1 \leq k \leq n+1$,

$$\boldsymbol{\mu}(\Theta_B^T)(w_k) - \boldsymbol{\mu}(\Theta_B^T)(v_{k-1}) = \frac{a^T(k)}{a^T(1)}\,\boldsymbol{\mu}(\Theta_B^T)(w_1).$$

Proof. We prove this property by induction on k. The base case, $k = 1$, is trivially true. Let $k > 1$. Then

$$\begin{aligned} &\boldsymbol{\mu}(\Theta_B^T)(w_k) - \boldsymbol{\mu}(\Theta_B^T)(v_{k-1}) \\ &= \left(\tfrac{1}{2} + \tfrac{1}{4}\boldsymbol{\mu}(\Theta_B^T)(w_{k-1}) + \tfrac{1}{4}\boldsymbol{\mu}(\Theta_B^T)(v_{k-2})\right) - \end{aligned}$$

$$\tfrac{1}{2}\left(T(k-1)(1+\boldsymbol{\mu}(\Theta_B^T)(w_{k-1}))+(1-T(k-1))(1+\boldsymbol{\mu}(\Theta_B^T)(v_{k-2}))\right)$$

[Proposition 6.3.5(b)]

$$\begin{aligned}
&= \tfrac{1}{4}\boldsymbol{\mu}(\Theta_B^T)(w_{k-1})-\tfrac{1}{4}\boldsymbol{\mu}(\Theta_B^T)(v_{k-2})+T(k-1)(\tfrac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_{k-2})-\tfrac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(w_{k-1}))\\
&= \tfrac{1}{2}(\tfrac{1}{2}-T(k-1))(\boldsymbol{\mu}(\Theta_B^T)(w_{k-1})-\boldsymbol{\mu}(\Theta_B^T)(v_{k-2}))\\
&= \frac{a^T(k)}{a^T(k-1)}(\boldsymbol{\mu}(\Theta_B^T)(w_{k-1})-\boldsymbol{\mu}(\Theta_B^T)(v_{k-2}))\\
&= \frac{a^T(k)}{a^T(k-1)}\frac{a^T(k-1)}{a^T(1)}\,\boldsymbol{\mu}(\Theta_B^T)(w_1) \qquad \text{[induction hypothesis]}\\
&= \frac{a^T(k)}{a^T(1)}\,\boldsymbol{\mu}(\Theta_B^T)(w_1).
\end{aligned}$$

□

The proposition below shows the relationship of the value of v_k with the current policy at v_k and the value of v_{k-1}.

Proposition 6.3.8. *For all* $1 \le k \le n+1$,

$$\boldsymbol{\mu}(\Theta_B^T)(v_k) = \tfrac{1}{2} + \tfrac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_{k-1}) + T(k)a^T(k).$$

Proof. First of all, we derive from Proposition 6.3.5(a) that

$$\boldsymbol{\mu}(\Theta_B^T)(w_1) - \boldsymbol{\mu}(\Theta_B^T)(v_0) = \tfrac{1}{2} - 0 = \tfrac{1}{2}. \tag{6.14}$$

As a consequence,

$$\begin{aligned}
&\boldsymbol{\mu}(\Theta_B^T)(v_k)\\
&= \tfrac{1}{2}(T(k)(1+\boldsymbol{\mu}(\Theta_B^T)(w_k))+(1-T(k))(1+\boldsymbol{\mu}(\Theta_B^T)(v_{k-1})))
\end{aligned}$$

[Proposition 6.3.5(b)]

$$
\begin{aligned}
&= \tfrac{1}{2} + \tfrac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_{k-1}) + \tfrac{1}{2}T(k)(\boldsymbol{\mu}(\Theta_B^T)(w_k) - \boldsymbol{\mu}(\Theta_B^T)(v_{k-1})) \\
&= \tfrac{1}{2} + \tfrac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_{k-1}) + \tfrac{1}{2}T(k)\frac{a^T(k)}{a^T(1)}\boldsymbol{\mu}(\Theta_B^T)(w_1) \\
&\qquad \text{[Proposition 6.3.7]} \\
&= \tfrac{1}{2} + \tfrac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_{k-1}) + T(k)a^T(k) \qquad \text{[(6.14)]}
\end{aligned}
$$

□

The next proposition shows that switching the policy at a vertex with higher index will not impact the values of the vertices with lower indices.

Proposition 6.3.9.

(a) For all $1 \leq \ell \leq k \leq n+1$, $a^T(\ell) = a^{\bar{T}_k}(\ell)$.

(b) For all $1 \leq \ell < k \leq n+1$, $\boldsymbol{\mu}(\Theta_B^T)(v_\ell) = \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_\ell)$.

Proof. Both parts are proved by induction on ℓ.

(a) For the base case, $\ell = 1$, we have that $a^T(1) = \frac{1}{4} = a^{\bar{T}_k}(1)$. In the inductive case, $1 < \ell \leq k$ and

$$
\begin{aligned}
a^T(\ell) &= \tfrac{1}{2}a^T(\ell-1)(\tfrac{1}{2} - T(\ell-1)) \\
&= \tfrac{1}{2}a^{\bar{T}_k}(\ell-1)(\tfrac{1}{2} - \bar{T}_k(\ell-1)) \qquad \text{[induction hypothesis]} \\
&= a^{\bar{T}_k}(\ell).
\end{aligned}
$$

(b) For the base case, $\ell = 1$, we have that

$$\begin{aligned}
\boldsymbol{\mu}(\Theta_B^T)(v_1) &= \tfrac{1}{2} + \tfrac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_0) + T(1)a^T(1) \qquad \text{[Proposition 6.3.8]} \\
&= \tfrac{1}{2} + \tfrac{1}{2}\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_0) + \bar{T}_k(1)a^{\bar{T}_k}(1) \qquad \text{[Definition 6.3.6]} \\
&= \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_1) \qquad \text{[Proposition 6.3.8].}
\end{aligned}$$

In the inductive case, $1 < \ell < k$ and

$$\begin{aligned}
\boldsymbol{\mu}(\Theta_B^T)(v_\ell) &= \tfrac{1}{2} + \tfrac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_{\ell-1}) + T(\ell)a^T(\ell) \qquad \text{[Proposition 6.3.8]} \\
&= \tfrac{1}{2} + \tfrac{1}{2}\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_{\ell-1}) + \bar{T}_k(\ell)a^{\bar{T}_k}(\ell) \\
&\qquad \text{[induction hypothesis and part 1]} \\
&= \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_\ell) \qquad \text{[Proposition 6.3.8].}
\end{aligned}$$

□

We can tell whether a vertex is switchable by the current policy T at the vertex and the sign of a^T at the vertex.

Proposition 6.3.10. *For all $1 \leq k \leq n+1$, v_k is switchable with respect to T if and only if*

$$(T(k) = 0 \wedge a^T(k) < 0) \vee (T(k) = 1 \wedge a^T(k) > 0)$$

Proof. Let $1 \leq k \leq n+1$. Then

v_k is switchable with respect to T

iff $\boldsymbol{\mu}(\Theta_B^T)(v_k) - \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_k) > 0$

iff $(\frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_{k-1}) + T(k)a^T(k)) - (\frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_{k-1}) + \bar{T}_k(k)a^{\bar{T}_k}(k)) > 0$

[Proposition 6.3.8]

iff $(\frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_{k-1}) + T(k)a^T(k)) - (\frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_{k-1}) + (1 - T(k))a^T(k)) > 0$

[Proposition 6.3.9]

iff $(2T(k) - 1)a^T(k) > 0$

iff $(T(k) = 0 \wedge a^T(k) < 0) \vee (T(k) = 1 \wedge a^T(k) > 0)$.

□

Corollary 6.3.11. *v_1 is switchable with respect to T if and only if $T(1) = 1$.*

Proof. By Proposition 6.3.10, $v(1)$ is switchable with respect to T is and only if

$$(T(1) = 0 \wedge a^T(1) < 0) \vee (T(1) = 1 \wedge a^T(1) > 0).$$

By Definition 6.3.6, $a^T(1) = \frac{1}{4} > 0$. Thus, v_1 is switchable with respect to T if and only if $T(1) = 1$. □

The next proposition captures that switching a vertex has no impact on the switchability of the vertices with lower indices.

Proposition 6.3.12. *For all $1 \leq \ell < k \leq n + 1$, v_ℓ is switchable with respect to T if and only if v_ℓ is switchable with respect to $\bar{T}_k$.*

Proof. Let $1 \leq \ell < k \leq n+1$. Then

v_ℓ is switchable with respect to T

iff $(T(\ell) = 0 \wedge a^T(\ell) < 0) \vee (T(\ell) = 1 \wedge a^T(\ell) > 0)$ [Proposition 6.3.10]

iff $(\bar{T}_k(\ell) = 0 \wedge a^{\bar{T}_k}(\ell) < 0) \vee (\bar{T}_k(\ell) = 1 \wedge a^{\bar{T}_k}(\ell) > 0)$ [Proposition 6.3.9]

iff v_ℓ is switchable with respect to $\bar{T}_k$ [Proposition 6.3.10]

□

The next proposition is technical and is only used in the proof of Proposition 6.3.14.

Proposition 6.3.13. *For all $1 \leq k < n+1$, if v_k is switchable with respect to T then $a^T(k+1) < 0$.*

Proof. We distinguish two cases. Assume that $T(k) = 0$. Since v_k is switchable with respect to T, we can conclude from Proposition 6.3.10 that $a^T(k) < 0$. Hence,

$$a^T(k+1) = \tfrac{1}{2}a^T(k)(\tfrac{1}{2} - T(k)) = \tfrac{1}{4}a^T(k) < 0.$$

Assume that $T(k) = 1$. Since v_k is switchable with respect to T, we can conclude from Proposition 6.3.10 that $a^T(k) > 0$. Hence,

$$a^T(k+1) = \tfrac{1}{2}a^T(k)(\tfrac{1}{2} - T(k)) = -\tfrac{1}{4}a^T(k) < 0.$$

□

Let T be a policy with $T(n+1) = 0, \ldots, T(k+2) = 0$ and $T(k+1) = 1$. If the vertex v_k is switchable, then after switching the policy at v_k, the leftmost $n+1-k$ vertices will become switchable.

Proposition 6.3.14. *For all $1 \leq k < n+1$, if v_k is switchable with respect to T and $T(n+1) = 0, \ldots, T(k+2) = 0$ and $T(k+1) = 1$ then $v_{n+1}, \ldots, v_{k+1}$ are switchable with respect to $\bar{T}_k$.*

Proof. According to Proposition 6.3.10, it suffices to show that $a^{\bar{T}_k}(n+1) < 0, \ldots, a^{\bar{T}_k}(k+2) < 0$ and $a^{\bar{T}_k}(k+1) > 0$. First of all,

$$\begin{aligned} a^{\bar{T}_k}(k+1) &= \tfrac{1}{2} a^{\bar{T}_k}(k)(\tfrac{1}{2} - \bar{T}_k(k)) \\ &= \tfrac{1}{2} a^{T}(k)(\tfrac{1}{2} - (1 - T(k))) \qquad \text{[Proposition 6.3.9]} \\ &= -\tfrac{1}{2} a^{T}(k)(\tfrac{1}{2} - T(k)) \\ &= -a^{T}(k+1). \end{aligned}$$

From Proposition 6.3.13 we can conclude that $a^{\bar{T}_k}(k+1) > 0$.

Next, we show that for all ℓ satisfying $k+2 \leq \ell \leq n+1$, $a^{\bar{T}_k}(\ell) < 0$ by induction on ℓ. In the base case, $\ell = k+2$, we have

$$\begin{aligned} a^{\bar{T}_k}(k+2) &= \tfrac{1}{2} a^{\bar{T}_k}(k+1)(\tfrac{1}{2} - \bar{T}_k(k+1)) \\ &= \tfrac{1}{2} a^{\bar{T}_k}(k+1)(\tfrac{1}{2} - T(k+1))) \\ &= -\tfrac{1}{4} a^{\bar{T}_k}(k+1). \end{aligned}$$

As we have seen above, $a^{\bar{T}_k}(k+1) > 0$ and, hence, $a^{\bar{T}_k}(k+2) < 0$. In the inductive case, $\ell > k+2$ and

$$\begin{aligned} a^{\bar{T}_k}(\ell) &= \tfrac{1}{2} a^{\bar{T}_k}(\ell-1)(\tfrac{1}{2} - \bar{T}_k(\ell-1)) \\ &= \tfrac{1}{2} a^{\bar{T}_k}(\ell-1)(\tfrac{1}{2} - T(\ell-1))) \\ &= \tfrac{1}{4} a^{\bar{T}_k}(\ell-1). \end{aligned}$$

By induction, $a^{\bar{T}_k}(\ell-1) < 0$. Therefore, $a^{\bar{T}_k}(\ell) < 0$. □

The next proposition reveals that the values of v_k and w_k cannot increase as the algorithm proceeds. We will use this fact in the proof of Proposition 6.3.16.

Proposition 6.3.15. *For all $1 \leq k \leq n+1$, if v_k is switchable with respect to T then*

(a) for all $1 \leq \ell \leq n+1$, $\boldsymbol{\mu}(\Theta_B^T)(v_\ell) \geq \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_\ell)$, and

(b) for all $1 \leq \ell \leq n+1$, $\boldsymbol{\mu}(\Theta_B^T)(w_\ell) \geq \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(w_\ell)$.

Proof. Let $1 \leq k \leq n+1$. Assume that v_k is switchable with respect to T.

The following equation will be used in the proof. Since $v_0 = (s_0, t_0)$ and $s_0 \sim t_0$,

$$\boldsymbol{\mu}(\Theta_B^T)(v_0) = \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(v_0) = 0 = \Theta_B^{\bar{T}_k}(\boldsymbol{\mu}(\Theta_B^{\bar{T}_k}))(v_0) = \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_0). \tag{6.15}$$

We prove this proposition by induction on ℓ. We distinguish the following cases.

- If $k = 1$ and $\ell = 1$, then $T(1) = 1$ by Corollary 6.3.11. Thus, $\bar{T}_k(1) = 0$.

$$\begin{aligned}
\boldsymbol{\mu}(\Theta_B^T)(v_1) &= \tfrac{1}{2} + \tfrac{1}{2}\,\boldsymbol{\mu}(\Theta_B^T)(w_1) \qquad [T(1) = 1 \text{ and Proposition 6.3.5(b)}] \\
&= \tfrac{1}{2} + \tfrac{1}{2} \times \tfrac{1}{2} \qquad [\text{Proposition 6.3.5(a)}] \\
&> \tfrac{1}{2} + \tfrac{1}{2} \times 0 \\
&= \tfrac{1}{2} + \tfrac{1}{2}\,\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_0) \qquad [(6.15)] \\
&= \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_1) \qquad [\bar{T}_k(1) = 0 \text{ and Proposition 6.3.5(b)}]
\end{aligned}$$

- If $k > 1$, $\ell = 1$ and $T(1) = 0$ then

$$\begin{aligned}
\boldsymbol{\mu}(\Theta_B^T)(v_1) &= \tfrac{1}{2} + \tfrac{1}{2}\,\boldsymbol{\mu}(\Theta_B^T)(v_0) \qquad [\text{Proposition 6.3.5(b)}] \\
&= \tfrac{1}{2} + \tfrac{1}{2}\,\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_0) \qquad [(6.15)] \\
&= \tfrac{1}{2} + \tfrac{1}{2}\,\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_0) + \bar{T}_k(1)a^{\bar{T}_k}(1) \\
&\qquad\qquad [k > 1 \text{ and } \bar{T}_k(1) = T(1) = 0] \\
&= \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_1) \qquad [\text{Proposition 6.3.8}]
\end{aligned}$$

- If $k > 1$, $\ell = 1$ and $T(1) = 1$ then

$$\begin{aligned}
\boldsymbol{\mu}(\Theta_B^T)(v_1) &= \tfrac{1}{2} + \tfrac{1}{2}\,\boldsymbol{\mu}(\Theta_B^T)(w_1) \qquad [\text{Proposition 6.3.5(b)}] \\
&= \tfrac{1}{2} + \tfrac{1}{2}\,\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(w_1) \qquad [\text{Proposition 6.3.5(a)}] \\
&= \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_1) \qquad [\text{Proposition 6.3.5(b)}]
\end{aligned}$$

- If $1 < \ell < k$ and $T(k) = 0$ then

$$\boldsymbol{\mu}(\Theta_B^T)(v_\ell) = \tfrac{1}{2} + \tfrac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_{\ell-1}) \qquad [\text{Proposition 6.3.8}]$$

$$\geq \frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_{\ell-1}) \qquad \text{[induction hypothesis]}$$

$$= \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_\ell) \qquad \text{[Proposition 6.3.8]}$$

- If $1 < \ell < k$ and $T(k) = 1$ then

$$\boldsymbol{\mu}(\Theta_B^T)(v_\ell) = \frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(w_\ell) \qquad \text{[Proposition 6.3.5(b)]}$$

$$\geq \frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(w_\ell) \qquad \text{[induction hypothesis]}$$

$$= \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_\ell) \qquad \text{[Proposition 6.3.5(b)]}$$

- If $\ell = k$ then $\boldsymbol{\mu}(\Theta_B^T)(v_\ell) > \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_\ell)$ since v_k is switchable with respect to T.

- If $k < \ell \leq n+1$ and $T(k) = 0$ then

$$\boldsymbol{\mu}(\Theta_B^T)(v_\ell) = \frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(v_{\ell-1}) \qquad \text{[Proposition 6.3.8]}$$

$$\geq \frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_{\ell-1}) \qquad \text{[induction hypothesis]}$$

$$= \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_\ell) \qquad \text{[Proposition 6.3.8]}$$

- If $k < \ell \leq n+1$ and $T(k) = 1$ then

$$\boldsymbol{\mu}(\Theta_B^T)(v_\ell) = \frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^T)(w_\ell) \qquad \text{[Proposition 6.3.5(b)]}$$

$$\geq \frac{1}{2} + \frac{1}{2}\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(w_\ell) \qquad \text{[induction hypothesis]}$$

$$= \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_\ell) \qquad \text{[Proposition 6.3.5(b)]}$$

This completes the proof of part 1. Next, we prove part 2.

- If $\ell = 1$ then, by Proposition 6.3.5(a),

$$\boldsymbol{\mu}(\Theta_B^T)(w_1) = \tfrac{1}{2} = \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(w_1).$$

- If $1 < \ell \leq n+1$ then

$$\begin{aligned}
\boldsymbol{\mu}(\Theta_B^T)(w_\ell) &= \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(w_\ell) \\
&= \tfrac{1}{4} + \tfrac{1}{4}\boldsymbol{\mu}(\Theta_B^T)(w_{\ell-1}) + \tfrac{1}{4}\boldsymbol{\mu}(\Theta_B^T)(v_{\ell-2}) \\
&\geq \tfrac{1}{4} + \tfrac{1}{4}\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(w_{\ell-1}) + \tfrac{1}{4}\boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(v_{\ell-2}) \\
&\qquad \text{[induction hypothesis]} \\
&= \Theta_B^{\bar{T}_k}(\boldsymbol{\mu}(\Theta_B^{\bar{T}_k}))(w_\ell) \\
&= \boldsymbol{\mu}(\Theta_B^{\bar{T}_k})(w_\ell).
\end{aligned}$$

□

By using the pigeonhole principle, we show that the policy iteration algorithm can have at most $2^{n-k+2} - 1$ iterations if the leftmost $n-k+2$ vertices are allowed to be switched. Moreover, after $2^{n-k+2} - 1$ iterations of switching the leftmost $n-k+2$ vertices, these vertices will no longer be switchable.

Proposition 6.3.16. *For all $1 \leq k \leq n+1$, if only $v_{n+1}, \ldots, v_k$ are allowed to be switched, then the loop 4–6 can be executed at most $2^{n-k+2} - 1$ times.*

Proof. Let $1 \leq k \leq n+1$. Assume that only $v_{n+1}, \ldots, v_k$ are allowed to be switched. Towards a contradiction, assume that the loop 4–6 is executed 2^{n-k+2} times. We

denote the value of T at the beginning of the $(i+1)$th iteration of the loop 4–6 by T_i for $0 \leq i \leq 2^{n-k+2}$. Since only $v_{n+1}, \ldots, v_k$ are allowed to be switched, T is only changed for the indices $k, \ldots, n+1$. Therefore, T can only take on 2^{n-k+2} different values. As a consequence, $T_i = T_j$ for some $0 \leq i < j \leq 2^{n-k+2}$. Assume that v_ℓ is switched in the $(i+1)$th iteration of the loop 4–6. Then

$$\begin{aligned}
\boldsymbol{\mu}(\Theta_B^{T_i})(v_\ell) &> \boldsymbol{\mu}(\Theta_B^{T_{i+1}})(v_\ell) \qquad [v_\ell \text{ is switchable in } T_i] \\
&\geq \boldsymbol{\mu}(\Theta_B^{T_j})(v_\ell) \\
&\qquad [\text{the value of } v_\ell \text{ cannot increase by Proposition 6.3.15 }] \\
&= \boldsymbol{\mu}(\Theta_B^{T_i})(v_\ell) \qquad [T_i = T_j]
\end{aligned}$$

which is a contradiction.

□

Finally, we come to the two main propositions.

Proposition 6.3.17. *For all $1 \leq k < n+1$, if $T(n+1) = 0, \ldots, T(k+1) = 0$ and $v_{n+1}, \ldots, v_k$ are switchable with respect to T then during the next $2^{n-k+2} - 1$ iterations of the loop 4–6 only $v_{n+1}, \ldots, v_k$ are switched resulting in $\bar{T}_k$.*

Proof. Let $1 \leq k < n+1$. Assume $T(n+1) = 0, \ldots, T(k+1) = 0$ and $v_{n+1}, \ldots, v_k$ are switchable with respect to T .

We prove the proposition by induction on k. The base case is $k = n$. Since v_{n+1} is switchable, in the first iteration of the loop 4–6, v_{n+1} is switched resulting in $\bar{T}_{n+1}$.

According to Proposition 6.3.12, v_n remains switchable with respect to $\bar{T}_{n+1}$ and v_n is switched resulting $\overline{\bar{T}_{n+1}}_n$ in the second iteration.

Next, we will show that v_{n+1} is switchable with respect to $\overline{\bar{T}_{n+1}}_n$. We have $\overline{\bar{T}_{n+1}}_n(n+1) = 1$. Since v_{n+1} is switchable with respect to T and $T(n+1) = 0$, by Proposition 6.3.10, we have

$$a^T(n+1) = \tfrac{1}{2}a^T(n)(\tfrac{1}{2} - T(n)) < 0.$$

By Proposition 6.3.9(a), we have

$$a^T(n) = a^{\bar{T}_{n+1}}(n) = a^{\overline{\bar{T}_{n+1}}_n}(n) \tag{6.16}$$

We distinguish two cases.

- If $T(n) = 0$, then by Proposition 6.3.10, $a^T(n) < 0$, since v_n is switchable with respect to T. By (6.16), $a^{\overline{\bar{T}_{n+1}}_n}(n) = a^T(n) < 0$ and by Definition 6.3.6

$$a^{\overline{\bar{T}_{n+1}}_n}(n+1) = \tfrac{1}{2}a^{\overline{\bar{T}_{n+1}}_n}(n)(\tfrac{1}{2} - \overline{\bar{T}_{n+1}}_n(n)) = \tfrac{1}{2}a^{\overline{\bar{T}_{n+1}}_n}(n)(\tfrac{1}{2} - 1) > 0$$

- If $T(n) = 1$, then by Proposition 6.3.10, $a^T(n) > 0$, since v_n is switchable with respect to T. By (6.16), $a^{\overline{\bar{T}_{n+1}}_n}(n) = a^T(n) > 0$ and by Definition 6.3.6

$$a^{\overline{\bar{T}_{n+1}}_n}(n+1) = \tfrac{1}{2}a^{\overline{\bar{T}_{n+1}}_n}(n)(\tfrac{1}{2} - \overline{\bar{T}_{n+1}}_n(n)) = \tfrac{1}{2}a^{\overline{\bar{T}_{n+1}}_n}(n)(\tfrac{1}{2} - 0) > 0$$

In both cases, according to Proposition 6.3.10, v^{n+1} is switchable with respect to $\overline{\bar{T}_{n+1}}_n$. Hence, in the third iteration of the loop 4–6, v^{n+1} is switched resulting in $\overline{\overline{\bar{T}_{n+1}}_{n}}_{n+1} = \bar{T}_n$.

In the inductive case, $1 \leq k < n$. Assume that $T(n+1) = 0, \ldots, T(k+1) = 0$ and $v_{n+1}, \ldots, v_k$ are switchable with respect to T. By induction, during the next $2^{n-k+1} - 1$ iterations of the loop 4–6 only $v_{n+1}, \ldots, v_{k+1}$ are switched resulting in $\bar{T}_{k+1}$. According to Proposition 6.3.16, $v_{n+1}, \ldots, v_{k+1}$ are not switchable after $2^{n-k+1} - 1$ iterations of the loop 4–6. By Proposition 6.3.12, v_k is switchable with respect to $\bar{T}_{k+1}$. Hence, in the next iteration of the loop 4–6, v_k remains switched resulting in $\overline{\bar{T}_{k+1k}}$. Since $T(n+1) = 0, \ldots, T(k+1) = 0$, we have that $\overline{\bar{T}_{k+1k}}(n+1) = 0, \ldots, \overline{\bar{T}_{k+1k}}(k+2) = 0, \overline{\bar{T}_{k+1k}}(k+1) = 1$. Hence, by Proposition 6.3.14 we have that $v_{n+1}, \ldots, v_{k+1}$ are switchable with respect to $\overline{\bar{T}_{k+1k}}$. Again, by induction, during the next $2^{n-k+1} - 1$ iterations of the loop 4–6 only $v_{n+1}, \ldots, v_{k+1}$ are switched resulting in $\overline{\overline{\bar{T}_{k+1k}}_{k+1}} = \bar{T}_k$. Hence, in total the loop is iterated $2^{n-k+1} - 1 + 1 + 2^{n-k+1} - 1 = 2^{n-k+2} - 1$ times.

□

Proposition 6.3.18. *The loop 4–6 is iterated $2^{n+1} - 1$ times, resulting in $\bar{T}_1$.*

Proof. Initially, $T(n+1) = 0, \ldots, T(2) = 0, T(1) = 1$.

First, by Definition 6.3.6, we have $a^T(1) = \frac{1}{4} > 0$. Since $T(1) = 1$, from Proposition 6.3.10 we can conclude that v_1 is switchable with respect to T.

Next, we show that $a^T(k) < 0$ for all $2 \leq k \leq n+1$ by induction on k. The base case is $k = 2$. According to Definition 6.3.6, $a^T(1) = \frac{1}{4} > 0$. Then

$$a^T(2) = \tfrac{1}{2} a^T(1)(\tfrac{1}{2} - T(1)) = \tfrac{1}{2} \times \tfrac{1}{4} \times (\tfrac{1}{2} - 1) < 0.$$

In the inductive case, $2 < k \leq n+1$. Then

$$\begin{aligned} a^T(k) &= \tfrac{1}{2}a^T(k-1)(\tfrac{1}{2} - T(k-1)) \\ &= \tfrac{1}{4}a^T(k-1) \\ &< 0 \qquad \text{[by induction]} \end{aligned}$$

From Proposition 6.3.10 we can conclude that $v_{n+1}, \ldots, v_2$ are switchable with respect to T. Hence, from Proposition 6.3.17 we can derive that the loop 4–6 is iterated $2^{n+1} - 1$ times, resulting in $\bar{T}_1$. □

It follows from Proposition 6.3.18 that the simple policy iteration algorithm requires exponential time in the worst case.

Theorem 6.3.19. *For each $n \in \mathbb{N}$, there exists a labelled Markov chain of size $O(n)$ such that simple policy iteration takes $\Omega(2^n)$ iterations.*

Proof. The labelled Markov chain $\mathcal{M}_n$ has size $O(n)$ and simple policy iteration takes $\Omega(2^n)$ iterations according to Proposition 6.3.18. □

6.4 General Policy Iteration

In Section 6.2, we presented the simple policy iteration algorithm where in each iteration of the loop the policy is adjusted for a single state pair (s,t) which is not locally optimal, that is, $\Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t) < \boldsymbol{\mu}(\Theta_B^T)(s,t)$. In this section, we present

the general policy iteration algorithm, for which the policy is updated for all state pairs which are not locally optimal.

1 for each $(s,t) \in S_?^2 \setminus B$

2 $\quad T(s,t) \leftarrow$ an element of $V(\Omega(\tau(s),\tau(t))$

3 while $\exists (s,t) \in S_?^2 \setminus B : \Delta(\boldsymbol{\mu}(\Theta_B^T))(s,t) < \boldsymbol{\mu}(\Theta_B^T)(s,t)$

4 $\quad U \leftarrow T$

5 $\quad$ for each $(s,t) \in S_?^2 \setminus B$ such that $\Delta(\boldsymbol{\mu}(\Theta_B^U))(s,t) < \boldsymbol{\mu}(\Theta_B^U)(s,t)$

6 $\quad\quad T(s,t) \leftarrow \underset{\omega \in V(\Omega(\tau(s),\tau(t)))}{\operatorname{argmin}} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta_B^U)(u,v)$

The proof of partial correctness is the same as the one for the simple policy iteration algorithm. To prove termination, we slightly generalize Theorem 6.2.7.

Theorem 6.4.1. *For all $T \in V(\mathcal{C}_B)$ and distinct (s_1,t_1), ..., $(s_n,t_n) \in S_?^2 \setminus B$, if $\Delta(\boldsymbol{\mu}(\Theta_B^T))(s_i,t_i) < \boldsymbol{\mu}(\Theta_B^T)(s_i,t_i)$ for all $1 \leq i \leq n$, then $\boldsymbol{\mu}(\Theta_B^{T[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}) \sqsubset \boldsymbol{\mu}(\Theta_B^T)$, where*

$$\pi_i = \underset{\omega \in V(\Omega(\tau(s_i),\tau(t_i)))}{\operatorname{argmin}} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta_B^T)(u,v)$$

for all $1 \leq i \leq n$.

Proof. Let $T \in V(\mathcal{C}_B)$ and (s_1,t_1), ..., $(s_n,t_n) \in S_?^2 \setminus B$. Assume that (s_1,t_1), ..., (s_n,t_n) are distinct and $\Delta(\boldsymbol{\mu}(\Theta_B^T))(s_i,t_i) < \boldsymbol{\mu}(\Theta_B^T)(s_i,t_i)$ for all $1 \leq i \leq n$. By Lemma 6.2.6, it remains to prove that $\Theta_B^{T[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}(\boldsymbol{\mu}(\Theta_B^T)) \sqsubset \boldsymbol{\mu}(\Theta_B^T)$, where $\pi_i = \underset{\omega \in V(\Omega(\tau(s_i),\tau(t_i)))}{\operatorname{argmin}} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta_B^T)(u,v)$ for all $1 \leq i \leq n$. Let x, $y \in S$. We distinguish the following cases.

- Assume $(x, y) = (s_i, t_i)$ for some $1 \leq i \leq n$. Then

$$\begin{aligned}
&\Theta_B^{T[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}(\boldsymbol{\mu}(\Theta_B^T))(x,y) \\
&= \sum_{u,v \in S} T[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n](x,y)(u,v)\ \boldsymbol{\mu}(\Theta_B^T)(u,v) \\
&= \sum_{u,v \in S} \pi_i(u,v)\ \boldsymbol{\mu}(\Theta_B^T)(u,v) \\
&= \min_{\omega \in V(\Omega(\tau(s_i),\tau(t_i)))} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta_B^T)(u,v) \\
&= \Delta(\boldsymbol{\mu}(\Theta_B^T))(s_i,t_i) \\
&< \boldsymbol{\mu}(\Theta_B^T)(s_i,t_i) \\
&= \boldsymbol{\mu}(\Theta_B^T)(x,y).
\end{aligned}$$

- Assume that $(x, y) \in S_0^2$. Then

$$\Theta_B^{T[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}(\boldsymbol{\mu}(\Theta_B^T))(x,y) = 0 = \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(x,y) = \boldsymbol{\mu}(\Theta_B^T)(x,y).$$

- Assume that $(x, y) \in B$. Then

$$\Theta_B^{T[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}(\boldsymbol{\mu}(\Theta_B^T))(x,y) = 1 = \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(x,y) = \boldsymbol{\mu}(\Theta_B^T)(x,y).$$

- Otherwise,

$$\begin{aligned}
&\Theta_B^{T[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}(\boldsymbol{\mu}(\Theta_B^T))(x,y) \\
&= \sum_{u,v \in S} T[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n](x,y)(u,v)\ \boldsymbol{\mu}(\Theta_B^T)(u,v) \\
&= \sum_{u,v \in S} T(x,y)(u,v)\ \boldsymbol{\mu}(\Theta_B^T)(u,v) \qquad [(x,y) \neq (s_i,t_i) \text{ for all } 1 \leq i \leq n]
\end{aligned}$$

$$= \Theta_B^T(\boldsymbol{\mu}(\Theta_B^T))(x, y)$$

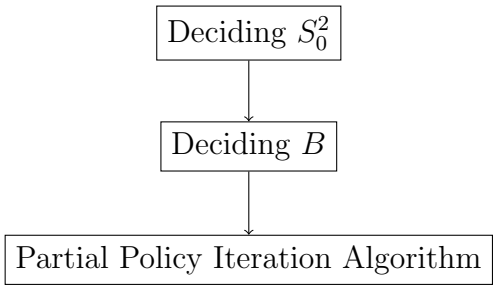
$$= \boldsymbol{\mu}(\Theta_B^T)(x, y).$$

□

Although we have proved an exponential lower bound for the simple policy iteration algorithm, it is still unknown whether there exists an exponential lower bound for the general policy iteration algorithm.

7 Partial Policy Iteration

In the previous chapter, we have shown that the basic algorithm by Bacci *et al.* [3] is the simple policy iteration algorithm. We have proved that their algorithm is wrong and corrected it. We have also presented the general policy iteration algorithm. These algorithms always compute the probabilistic bisimilarity distances for all state pairs. An interesting question is whether the policy iteration algorithms can be modified so that we only need to compute the probabilistic bisimilarity distances for the state pairs we are interested in. In this chapter, we introduce the partial policy iteration algorithms, inspired by the on-the-fly algorithm by Bacci *et al.* [3]. We will show that their on-the-fly algorithm is wrong and will correct it. Furthermore, as we will see in Example 7.1.10, though in the beginning a partial policy algorithm only considers the state pairs in a query set, it may end up computing the probabilistic bisimilarity distances for more state pairs. The following diagram shows the general steps of applying partial policy algorithms, in which the partial policy algorithms replace the policy iteration algorithms as the third step.



Note that the partial policy iteration algorithm can be either the simple partial policy iteration of Section 7.1 or the general partial policy iteration of Section 7.3.

We will study the time complexity of the partial policy iteration algorithms. We will show that the running time of the simple partial policy iteration can be exponential, similar to the simple policy iteration algorithm. Thus, the algorithm of deciding the distances, which uses simple partial policy iteration, has an exponential lower bound. As the time complexity of the general policy iteration is unknown, the time complexity of the general partial policy algorithm, which uses partial policies, remains unknown as well.

7.1 Simple Partial Policy Iteration

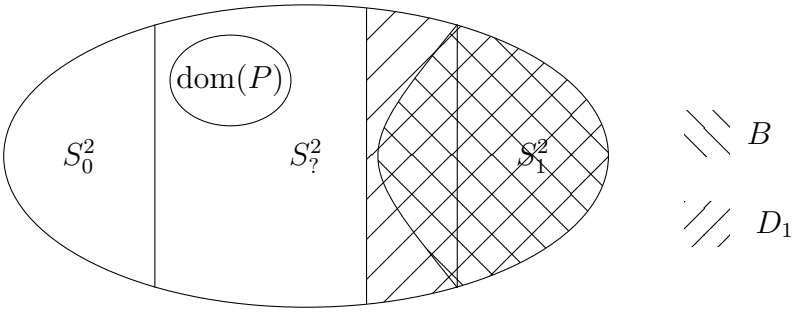
Instead of total policies, the partial policy iteration algorithms use partial ones. Hence, we generalize the set of total policies $V(\mathcal{C}_B)$ as follows. We denote the set of partial functions from $S_?^2 \setminus B$ to $\mathrm{Distr}(S^2)$ by $S_?^2 \setminus B \nrightarrow \mathrm{Distr}(S^2)$ and the domain of such a function P by $\mathrm{dom}(P)$.

Definition 7.1.1. For a labelled Markov chain $\langle S, L, \tau, \ell \rangle$, the set $\mathcal{P}$ of partial

policies is defined by

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

Recall that S^2_0, S^2_1 and $S^2_?$ form a partition of S^2. For a given $P \in \mathcal{P}$, S^2_0, B, $\mathrm{dom}(P)$ and $S^2_? \setminus B \setminus \mathrm{dom}(P)$ form a partition of S^2 as well. This partition is used to generalize the function Θ^T_B of Definition 6.1.2 to the partial setting.



Definition 7.1.2. Let $P \in \mathcal{P}$. The function $\Theta^P_B : [0,1]^{S^2} \to [0,1]^{S^2}$ is defined by

$$\Theta^P_B(d)(s,t) = \begin{cases} 1 & \text{if } (s,t) \in B \\ \displaystyle\sum_{u,v \in S} P(s,t)(u,v)\, d(u,v) & \text{if } (s,t) \in \mathrm{dom}(P) \\ 0 & \text{otherwise.} \end{cases}$$

The following proposition shows that the function is Θ^P_B monotone.

Proposition 7.1.3. *For all d, $e \in [0,1]^{S^2}$, if $d \sqsubseteq e$ then $\Theta^P_B(d) \sqsubseteq \Theta^P_B(e)$.*

Proof. Let d, $e \in [0,1]^{S^2}$ with $d \sqsubseteq e$. Let s, $t \in S$. We distinguish three cases.

- If $(s,t) \in B$ then

$$\Theta^P_B(d)(s,t) = 1 = \Theta^P_B(e)(s,t).$$

- If $(s,t) \in \mathrm{dom}(P)$ then

$$\begin{aligned}\Theta_B^P(d)(s,t) &= \sum_{u,v \in S} P(s,t)(u,v)\, d(u,v) \\ &\leq \sum_{u,v \in S} P(s,t)(u,v)\, e(u,v) \qquad [d \sqsubseteq e] \\ &= \Theta_B^P(e)(s,t).\end{aligned}$$

- Otherwise,

$$\Theta_B^P(d)(s,t) = 0 = \Theta_B^P(e)(s,t).$$

□

Since $[0,1]^{S^2}$ is a complete lattice and Θ_B^P is a monotone function, we can conclude from the Knaster-Tarski fixed point theorem (Theorem 2.1.8(a)) that Θ_B^P has a least fixed point. We denote this fixed point by $\boldsymbol{\mu}(\Theta_B^P)$.

The set $Q \subseteq S_?^2 \setminus B$ contains those pairs of states in which we are interested.

```
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) : Δ(μ(Θ_B^P))(s,t) < μ(Θ_B^P)(s,t)
6     P(s,t) ← argmin_{ω ∈ V(Ω(τ(s),τ(t)))} Σ_{u,v∈S} ω(u,v) μ(Θ_B^P)(u,v)
7     expand(P,s,t)
```

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

```
8  while ∃(u,v) ∈ support(P(s,t)) ∩ (S_?^2 \ B) : (u,v) ∉ dom(P)
9      P(u,v) ← an element of V(Ω(τ(u),τ(v)))
10     expand(P,u,v)
```

Intuitively, the function expand expands dom(P), the domain of the partial function P. As can be seen from Example 7.1.10, without the function expand, the algorithm will not consider a sufficient number of state pairs and will terminate with the incorrect values of the distances.

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

Definition 7.1.4. Let $P \in \mathcal{P}$ and $X \subseteq S_?^2 \setminus B$. The predicate $F(P,X)$ is defined by

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

Let us fix $X \subseteq S_?^2 \setminus B$. Roughly, this predicate $F(P,X)$ captures the fact that P is fully defined when we exclude X from its domain. Let $P \in \mathcal{P}$ and $(s,t) \in S_?^2 \setminus B$. 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 $S_?^2 \setminus B \setminus \mathrm{dom}(P)$ becomes smaller. At the end of the loop we have $F(P, X)$ and $(u, v) \in \mathrm{dom}(P)$ for all $(u, v) \in \mathrm{support}(P(s,t)) \cap (S_?^2 \setminus B)$, that is, $\mathrm{support}(P(s,t)) \cap (S_?^2 \setminus B) \subseteq \mathrm{dom}(P)$. Therefore, $F(P, X \setminus \{(s,t)\})$.

Let $P \in \mathcal{P}$, $(s,t) \in S_?^2 \setminus B$ and $X \subseteq S_?^2 \setminus B$. Next, we annotate the code of the simple partial policy iteration algorithm.

1 $P \leftarrow$ the partial function with empty domain

$\{F(P, \emptyset)\}$

2 for each $(s,t) \in Q$

$\{F(P, \emptyset)\}$

3 $P(s,t) \leftarrow$ an element of $V(\Omega(\tau(s), \tau(t)))$

$\{F(P, \{(s,t)\})\}$

4 expand(P, s, t)

$\{F(P, \emptyset)\}$

$\{F(P, \emptyset) \wedge Q \subseteq \mathrm{dom}(P)\}$

5 while $\exists (s,t) \in \mathrm{dom}(P) : \Delta(\boldsymbol{\mu}(\Theta_B^P))(s,t) < \boldsymbol{\mu}(\Theta_B^P)(s,t)$

$\{F(P, \emptyset) \wedge Q \subseteq \mathrm{dom}(P)\}$

6 $P(s,t) \leftarrow \underset{\omega \in V(\Omega(\tau(s), \tau(t)))}{\mathrm{argmin}} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta_B^P)(u,v)$

$\{F(P, \{(s,t)\}) \wedge Q \subseteq \mathrm{dom}(P)\}$

7 expand(P, s, t)

$\{F(P, \emptyset) \wedge Q \subseteq \mathrm{dom}(P)\}$

$\{F(P, \emptyset) \wedge Q \subseteq \mathrm{dom}(P) \wedge \forall (s,t) \in \mathrm{dom}(P) : \boldsymbol{\mu}(\Theta_B^P)(s,t) \leq \Delta(\boldsymbol{\mu}(\Theta_B^P))(s,t)\}$

We also annotate the code of expand(P, s, t). The assertions are given in red.

$\{F(P, X)\}$

8 while $\exists (u,v) \in \mathrm{support}(P(s,t)) \cap S_?^2 : (u,v) \notin \mathrm{dom}(P)$

$\{F(P, X)\}$

9 $P(u,v) \leftarrow$ an element of $V(\Omega(\tau(u), \tau(v)))$

$\{F(P, X \cup \{(u,v)\})\}$

10 expand(P, u, v)

$\{F(P, X)\}$

$\{F(P, X \setminus \{(s,t)\})\}$

The proof of partial correctness of this simple partial policy iteration algorithm is similar to the partial correctness proof provided in Section 6.2. If the above algorithm terminates, then we have that

$$F(P, \emptyset) \wedge Q \subseteq \mathrm{dom}(P) \wedge \forall (s,t) \in \mathrm{dom}(P) : \ \boldsymbol{\mu}(\Theta_B^P)(s,t) \leq \Delta(\boldsymbol{\mu}(\Theta_B^P))(s,t)$$

at termination. As we will show next, from the above we can conclude that $\boldsymbol{\mu}(\Theta_B^P)$ and $\boldsymbol{\mu}(\Delta)$ coincide on Q and, hence, $\boldsymbol{\mu}(\Theta_B^P)$ contains the probabilistic bisimilarity

distances of Q.

We define a new distance function $d_B^P : S^2 \to [0,1]$. We will show in Proposition 7.1.6 that $\boldsymbol{\mu}(\Delta) \sqsubseteq d_B^P$ and use this fact to prove that $\boldsymbol{\mu}(\Theta_B^P)$ and $\boldsymbol{\mu}(\Delta)$ agree on the values in $\mathrm{dom}(P)$.

Definition 7.1.5. Let $P \in \mathcal{P}$. The function $d_B^P : S^2 \to [0,1]$ is defined by

$$d_B^P(s,t) = \begin{cases} 0 & \text{if } (s,t) \in S_0^2 \\ \boldsymbol{\mu}(\Theta_B^P)(s,t) & \text{if } (s,t) \in \mathrm{dom}(P) \\ 1 & \text{otherwise} \end{cases}$$

Proposition 7.1.6. *For all $P \in \mathcal{P}$, if $F(P,\emptyset)$ then $\boldsymbol{\mu}(\Delta) \sqsubseteq d_B^P$.*

Proof. Let $P \in \mathcal{P}$ and assume $F(P,\emptyset)$. $\boldsymbol{\mu}(\Delta)$ is the least fixed point of Δ. By the Knaster-Tarski theorem (Theorem 2.1.8(c)), $\boldsymbol{\mu}(\Delta)$ is the least pre-fixed point of Δ. Hence, to show that $\boldsymbol{\mu}(\Delta) \sqsubseteq d_B^P$, it suffices to show that $\Delta(d_B^P) \sqsubseteq d_B^P$.

Let s, $t \in S$. Assume $F(P,\emptyset)$. We distinguish the following four cases.

- If $(s,t) \in S_0^2$, then $s \sim t$. According to Definition 2.1.5, we can conclude that $\ell(s) = \ell(t)$, and there must exist an $\pi \in \Omega(\tau(s), \tau(t))$ such that

support$(\pi) \subseteq \sim$.

$$\begin{aligned}
\Delta(d_B^P)(s,t) &= \min_{\omega \in \Omega(\tau(s),\tau(t))} \sum_{(u,v)\in S^2} \omega(u,v)\, d_B^P(u,v) \\
&\leq \sum_{(u,v)\in S^2} \pi(u,v)\, d_B^P(u,v) \\
&= \sum_{(u,v)\in \mathrm{support}(\pi)} \pi(u,v)\, d_B^P(u,v) \\
&= 0 \qquad [\forall u \sim v : d_B^P(u,v) = 0] \\
&= d_B^P(s,t).
\end{aligned}$$

Thus, $\Delta(d_B^P)(s,t) = 0 = d_B^P(s,t)$.

- Assume $(s,t) \in \mathrm{dom}(P)$. We have that $(s,t) \notin B$ and $s \not\sim t$. Since $F(P,\emptyset)$ and $(s,t) \in \mathrm{dom}(P)$, we have that $\mathrm{support}(P(s,t)) \cap (S_?^2 \setminus B) \subseteq \mathrm{dom}(P)$. Hence, $\mathrm{support}(P(s,t)) \subseteq S_0^2 \cup B \cup (S_?^2 \cap \mathrm{dom}(P))$.

  Next, we prove that d_B^P and $\boldsymbol{\mu}(\Theta_B^P)$ coincide on $\mathrm{support}(P(s,t))$, that is, for all $(u,v) \in \mathrm{support}(P(s,t))$,

$$d_B^P(u,v) = \boldsymbol{\mu}(\Theta_B^P)(u,v). \tag{7.1}$$

  Let $(u,v) \in \mathrm{support}(P(s,t))$. We distinguish the following cases.

  - If $(u,v) \in S_0^2$ then

$$d_B^P(u,v) = 0 = \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(u,v) = \boldsymbol{\mu}(\Theta_B^P)(u,v).$$

  - If $(u,v) \in B$ then

$$d_B^P(u,v) = 1 = \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(u,v) = \boldsymbol{\mu}(\Theta_B^P)(u,v).$$

– If $(u,v) \notin B$, $(u,v) \notin S_0^2$ and $(u,v) \in \mathrm{dom}(P)$ then

$$d_B^P(u,v) = \boldsymbol{\mu}(\Theta_B^P)(u,v).$$

Hence,

$$\begin{aligned}
\Delta(d_B^P)(s,t) &= \min_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{u,v \in S} \omega(u,v)\ d_B^P(u,v) \\
&\leq \sum_{u,v \in S} P(s,t)(u,v)\ d_B^P(u,v) \qquad [P(s,t) \in V(\Omega(\tau(s),\tau(t)))] \\
&= \sum_{(u,v) \in \mathrm{support}(P(s,t))} P(s,t)(u,v)\ d_B^P(u,v) \\
&= \sum_{(u,v) \in \mathrm{support}(P(s,t))} P(s,t)(u,v)\ \boldsymbol{\mu}(\Theta_B^P)(u,v) \qquad [(7.1)] \\
&= \sum_{u,v \in S} P(s,t)(u,v)\ \boldsymbol{\mu}(\Theta_B^P)(u,v) \qquad [P(s,t) \in V(\Omega(\tau(s),\tau(t)))] \\
&= \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(s,t) \\
&= \boldsymbol{\mu}(\Theta_B^P)(s,t) \\
&= d_B^P(s,t) \qquad [(s,t) \in \mathrm{dom}(P)].
\end{aligned}$$

- Otherwise, $s \not\sim t$ and $(s,t) \notin \mathrm{dom}(P)$,

$$\Delta(d_B^P)(s,t) \leq 1 = d_B^P(s,t).$$

□

Proposition 7.1.7. *For all $P \in \mathcal{P}$, if $\boldsymbol{\mu}(\Theta_B^P)(s,t) \leq \Delta(\boldsymbol{\mu}(\Theta_B^P))(s,t)$ for all $(s,t) \in \mathrm{dom}(P)$, then $\boldsymbol{\mu}(\Theta_B^P) \sqsubseteq \boldsymbol{\mu}(\Delta)$.*

Proof. Let $P \in \mathcal{P}$. Assume $\boldsymbol{\mu}(\Theta_B^P)(s,t) \leq \Delta(\boldsymbol{\mu}(\Theta_B^P))(s,t)$ for all $(s,t) \in \mathrm{dom}(P)$. According to Theorem 6.2.2(c, d), $\boldsymbol{\mu}(\Delta)$ is the unique fixed point of Δ_B. Hence, $\boldsymbol{\mu}(\Delta)$ is also the greatest fixed point of Δ_B. By the Knaster-Tarski theorem (Theorem 2.1.8(d)), $\boldsymbol{\mu}(\Delta)$ is the greatest post-fixed point of Δ_B. Hence, to show $\boldsymbol{\mu}(\Theta_B^P) \sqsubseteq \boldsymbol{\mu}(\Delta)$, it suffices to show $\boldsymbol{\mu}(\Theta_B^P)$ is a post-fixed point of Δ_B.

Let $(s,t) \notin \mathrm{dom}(P)$. We distinguish two cases.

- If $(s,t) \in B$ then by Definition 7.1.2,

$$\boldsymbol{\mu}(\Theta_B^P)(s,t) = \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(s,t) = 1 = \Delta_B(\boldsymbol{\mu}(\Theta_B^P))(s,t).$$

- Otherwise,

$$\boldsymbol{\mu}(\Theta_B^P)(s,t) = \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(s,t) = 0 \leq \Delta_B(\boldsymbol{\mu}(\Theta_B^P))(s,t).$$

Let $(s,t) \in \mathrm{dom}(P)$. As $(s,t) \notin B$ and $(s,t) \notin S_0^2$, by Definition 7.1.2, we have

$$\boldsymbol{\mu}(\Theta_B^P)(s,t) = \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(s,t) \leq \Delta(\boldsymbol{\mu}(\Theta_B^P))(s,t) = \Delta_B(\boldsymbol{\mu}(\Theta_B^P))(s,t).$$

Hence, $\boldsymbol{\mu}(\Theta_B^P) \sqsubseteq \Delta_B(\boldsymbol{\mu}(\Theta_B^P))$. That is, $\boldsymbol{\mu}(\Theta_B^P)$ is a post-fixed point of Δ_B. □

The next theorem proves the partial correctness of the simple partial policy iteration algorithm, that is, if the algorithm terminates then it computes the probabilistic bisimilarity distances for the state pairs in $\mathrm{dom}(P)$.

Theorem 7.1.8. *For all $P \in \mathcal{P}$, if $F(P,\emptyset)$ and $\boldsymbol{\mu}(\Theta_B^P)(s,t) \leq \Delta(\boldsymbol{\mu}(\Theta_B^P))(s,t)$ for all $(s,t) \in \mathrm{dom}(P)$, then $\boldsymbol{\mu}(\Theta_B^P)(s,t) = \boldsymbol{\mu}(\Delta)(s,t)$ for all $(s,t) \in \mathrm{dom}(P)$.*

Proof. Let $P \in \mathcal{P}$. Assume that $F(P, \emptyset)$ and $\boldsymbol{\mu}(\Theta_B^P)(s,t) \leq \Delta(\boldsymbol{\mu}(\Theta_B^P))(s,t)$ for all $(s,t) \in \mathrm{dom}(P)$. Let $(s,t) \in \mathrm{dom}(P)$. Then

$$\begin{aligned} \boldsymbol{\mu}(\Theta_B^P)(s,t) &\leq \boldsymbol{\mu}(\Delta)(s,t) \qquad \text{[Proposition 7.1.7]} \\ &\leq d_B^P(s,t) \qquad \text{[Proposition 7.1.6]} \\ &= \boldsymbol{\mu}(\Theta_B^P)(s,t). \end{aligned}$$

□

The above theorem shows that if the simple partial policy iteration algorithm terminates, $\boldsymbol{\mu}(\Theta_B^P)$ computes the bisimilarity distances for those state pairs in $\mathrm{dom}(P)$. Since $Q \subseteq \mathrm{dom}(P)$, it computes the distances of all the state pairs in Q. It remains to prove that the algorithm does terminate. 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 $\langle S_?^2 \setminus B \setminus \mathrm{dom}(P), P \rangle$ 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 6.2.4. Assume that P is updated for (s,t) on line 6 of the current iteration of the loop. We distinguish two cases. If $(u,v) \notin \mathrm{dom}(P)$ for some $(u,v) \in \mathrm{support}(P(s,t)) \cap (S_?^2 \setminus B)$, then expand$(P,s,t)$ on line 7 will assign a value to $P(u,v)$ on line 9 of the expand function. As a consequence, $\mathrm{dom}(P)$ becomes bigger and, hence, the first component of $\langle S_?^2 \setminus B \setminus \mathrm{dom}(P), P \rangle$ becomes smaller. Note that in this case P may not become smaller as $\boldsymbol{\mu}(\Theta_B^P)(u,v)$ was zero and may

have become positive. Otherwise, $\text{support}(P(s,t)) \cap (S^2_? \setminus B) \subseteq \text{dom}(P)$. In that case, the expand function does not perform any assignment to P and, therefore, $\text{dom}(P)$ 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 6.2.5) for some π. As we show next, in this case the second component, that is, the partial policy, becomes smaller.

Theorem 7.1.9. *For all* $P \in \mathcal{P}$ *and* $(s,t) \in \text{dom}(P)$, *if* $\Delta(\boldsymbol{\mu}(\Theta^P_B))(s,t) < \boldsymbol{\mu}(\Theta^P_B)(s,t)$, *then* $P[(s,t)/\pi] \prec P$, *where* $\pi = \underset{\omega \in V(\Omega(\tau(s),\tau(t)))}{\text{argmin}} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta^P_B)(u,v)$.

Proof. Let $P \in \mathcal{P}$ and $(s,t) \in \text{dom}(P)$. Assume that $\Delta(\boldsymbol{\mu}(\Theta^P_B))(s,t) < \boldsymbol{\mu}(\Theta^P_B)(s,t)$. Let $\pi = \underset{\omega \in V(\Omega(\tau(s),\tau(t)))}{\text{argmin}} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta^P_B)(u,v)$. By Lemma 6.2.6, it suffices to prove that $\Theta^{P[(s,t)/\pi]}_B(\boldsymbol{\mu}(\Theta^P_B)) \sqsubset \boldsymbol{\mu}(\Theta^P_B)$. Let x, $y \in S$. We distinguish the following cases.

- Assume $(x,y) = (s,t)$. Then

$$\begin{aligned}
\Theta^{P[(s,t)/\pi]}_B(\boldsymbol{\mu}(\Theta^P_B))(x,y) &= \sum_{u,v \in S} P[(s,t)/\pi](x,y)(u,v)\ \boldsymbol{\mu}(\Theta^P_B)(u,v) \\
&= \sum_{u,v \in S} \pi(u,v)\ \boldsymbol{\mu}(\Theta^P_B)(u,v) \qquad [(x,y) = (s,t)] \\
&= \min_{\omega \in V(\Omega(\tau(s),\tau(t)))} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta^P_B)(u,v) \\
&= \Delta(\boldsymbol{\mu}(\Theta^P_B))(s,t) \\
&< \boldsymbol{\mu}(\Theta^P_B)(s,t) \\
&= \boldsymbol{\mu}(\Theta^P_B)(x,y).
\end{aligned}$$

- Assume that $(x, y) \in S_0^2$. Then

$$\Theta_B^{P[(s,t)/\pi]}(\boldsymbol{\mu}(\Theta_B^P))(x, y) = 0 = \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(x, y) = \boldsymbol{\mu}(\Theta_B^P)(x, y).$$

- Assume $(x, y) \in B$. Then

$$\Theta_B^{P[(s,t)/\pi]}(\boldsymbol{\mu}(\Theta_B^P))(x, y) = 1 = \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(x, y) = \boldsymbol{\mu}(\Theta_B^P)(x, y).$$

- Assume that $\ell(x) = \ell(y)$, $(x, y) \notin \mathrm{dom}(P)$ and $(x, y) \notin B$. Since $(x, y) \neq (s, t)$, we have that $(x, y) \notin \mathrm{dom}(P[(s, t)/\pi])$ and

$$\Theta_B^{P[(s,t)/\pi]}(\boldsymbol{\mu}(\Theta_B^P))(x, y) = 0 = \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(x, y) = \boldsymbol{\mu}(\Theta_B^P)(x, y).$$

- Otherwise, $(x, y) \in \mathrm{dom}(P)$ and

$$\begin{aligned}
&\Theta_B^{P[(s,t)/\pi]}(\boldsymbol{\mu}(\Theta_B^P))(x, y) \\
&= \sum_{u,v \in S} P[(s, t)/\pi](x, y)(u, v)\ \boldsymbol{\mu}(\Theta_B^P)(u, v) \\
&= \sum_{u,v \in S} P(x, y)(u, v)\ \boldsymbol{\mu}(\Theta_B^P)(u, v) \qquad [(x, y) \neq (s, t)] \\
&= \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(x, y) \\
&= \boldsymbol{\mu}(\Theta_B^P)(x, y).
\end{aligned}$$

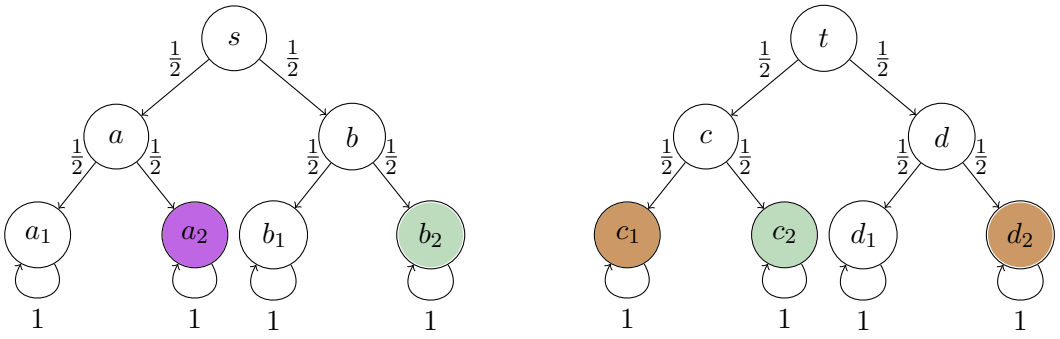
□

The on-the-fly algorithm of Bacci *et al.* differs in three major ways from our simple partial policy iteration algorithm. First of all, as we have already mentioned

in Section 5, they use Ψ^T while we use Θ_B^P. The main difference of these two functions is that Θ_B^P maps all the probabilistic bisimilar state pairs to zero. In the proof of Theorem 6.1.9 we give an example which shows that deciding probabilistic bisimilarity is essential for correctly computing the distances. Secondly, on line 5, instead of considering all the state pairs in $\mathrm{dom}(P)$, they consider only those state pairs that are reachable from the state pairs in Q in the coupled Markov chain $\langle S^2, P\rangle$. But, as we will show below, as a result they do not always correctly compute the distances. Thirdly, they map the state pairs in $S^2_? \setminus B \setminus \mathrm{dom}(P)$ to one, while we map them to zero.

We conclude this section with an example which shows that Bacci *et al.* do not always consider partial policies that are defined for sufficiently many state pairs.

Example 7.1.10. *Consider the labelled Markov chain presented below. 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

$$P(s,t) = \tfrac{1}{2}\mathrm{Dir}_{(a,d)} + \tfrac{1}{2}\mathrm{Dir}_{(b,c)}$$

$$P(a,d) = \tfrac{1}{2}\mathrm{Dir}_{(a_1,d_2)} + \tfrac{1}{2}\mathrm{Dir}_{(a_2,d_1)}$$

$$P(b,c) = \tfrac{1}{2}\mathrm{Dir}_{(b_1,c_2)} + \tfrac{1}{2}\mathrm{Dir}_{(b_2,c_1)}.$$

At this point, we have $\boldsymbol{\mu}(\Theta_B^P)(s,t) = 1$, $\boldsymbol{\mu}(\Theta_B^P)(a,d) = 1$ and $\boldsymbol{\mu}(\Theta_B^P)(b,c) = 1$. Note that (s,t) is not locally optimal, that is, $\Delta(\boldsymbol{\mu}(\Theta_B^P))(s,t) < \boldsymbol{\mu}(\Theta_B^P)(s,t)$. 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

$$P(a,c) = \tfrac{1}{2}\mathrm{Dir}_{(a_1,c_1)} + \tfrac{1}{2}\mathrm{Dir}_{(a_2,c_2)}$$

$$P(b,d) = \tfrac{1}{2}\mathrm{Dir}_{(b_1,d_1)} + \tfrac{1}{2}\mathrm{Dir}_{(b_2,d_2)}$$

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

$$P(a,d) = \tfrac{1}{2}\mathrm{Dir}_{(a_1,d_1)} + \tfrac{1}{2}\mathrm{Dir}_{(a_2,d_2)}$$

$$P(b,c) = \tfrac{1}{2}\mathrm{Dir}_{(b_1,c_1)} + \tfrac{1}{2}\mathrm{Dir}_{(b_2,c_2)}$$

At this point, we have $\boldsymbol{\mu}(\Theta_B^P)(s,t) = \frac{3}{4}$, $\boldsymbol{\mu}(\Theta_B^P)(a,d) = \frac{1}{2}$ and $\boldsymbol{\mu}(\Theta_B^P)(b,c) = \frac{1}{2}$. 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* $\boldsymbol{\mu}(\Theta_B^P)(s,t) = \frac{1}{2}$ *which is the probabilistic bisimilarity distance of* (s,t).

7.2 An Exponential Lower Bound of Simple Partial Policy Iteration

Below, we will prove the exponential lower bound for the simple partial policy iteration algorithm. We will reuse the labelled Markov chains defined in Definition 6.3.1.

Recall that the labelled Markov chain $\mathcal{M}_n$ has $4n+8$ states and $7n+14$ transitions. Next, we show that it takes at least $2^{n+1}-1$ iterations of the simple partial policy iteration algorithm to compute the distance of s_{n+1} and t_{n+1} in $\mathcal{M}_n$.

The partial policy iteration algorithm contains some nondeterminism. In particular, on line 3 and 9, an element of $V(\Omega(\tau(s),\tau(t)))$ and $V(\Omega(\tau(u),\tau(v)))$ is chosen. Furthermore, on line 5 a state pair $(s,t) \in \mathrm{dom}(P)$ with $\Delta(\boldsymbol{\mu}(\Theta_B^P))(s,t) < \boldsymbol{\mu}(\Theta_B^P)(s,t)$ is selected. Note that, if $2 \leq i \leq n+1$, then

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

Also,

$$V(\Omega(\tau(s_1),\tau(t_1))) = \{\tfrac{1}{2}\mathrm{Dir}_{(s_0,u_0)} + \tfrac{1}{2}\mathrm{Dir}_{(r_0,t_0)}, \tfrac{1}{2}\mathrm{Dir}_{(s_0,t_0)} + \tfrac{1}{2}\mathrm{Dir}_{(r_0,u_0)}\}.$$

Furthermore, if $1 \leq i < n+1$, then

$$V(\Omega(\tau(s_i),\tau(u_i))) = \{\tfrac{1}{2}\mathrm{Dir}_{(r_{i-1},u_{i-1})} + \tfrac{1}{4}\mathrm{Dir}_{(s_{i-1},t_{i-1})} + \tfrac{1}{4}\mathrm{Dir}_{(s_{i-1},u_{i-1})},$$

$$\tfrac{1}{2}\mathrm{Dir}_{(s_{i-1},u_{i-1})} + \tfrac{1}{4}\mathrm{Dir}_{(r_{i-1},u_{i-1})} + \tfrac{1}{4}\mathrm{Dir}_{(r_{i-1},t_{i-1})}\}.$$

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

Theorem 7.2.1. *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.*

Proof. Let $n \in \mathbb{N}$. After calling the function expand(P, s_{n+1}, t_{n+1}), all the vertices in $\mathcal{G}_n$ of Definition 6.3.2 will be added to the domain of P. The rest of the proof is the same as the proof of Theorem 6.3.19. □

7.3 General Partial Policy Iteration

The general policy iteration algorithm, which we presented in Section 6.4, can be generalized to use partial policies instead of total ones. The general partial policy iteration algorithm is as follows.

1 $P \leftarrow$ the partial function with empty domain
2 for each $(s,t) \in Q$
3 $P(s,t) \leftarrow$ an element of $V(\Omega(\tau(s), \tau(t)))$
4 expand(P, s, t)
5 while $\exists (s,t) \in \mathrm{dom}(P) : \Delta(\boldsymbol{\mu}(\Theta_B^P))(s,t) < \boldsymbol{\mu}(\Theta_B^P)$

6 $R \leftarrow P$

7 for each $(s,t) \in \mathrm{dom}(R)$ such that $\Delta(\boldsymbol{\mu}(\Theta^R))(s,t) < \boldsymbol{\mu}(\Theta^R)(s,t)$

8 $P(s,t) \leftarrow \underset{\omega \in V(\Omega(\tau(s),\tau(t)))}{\mathrm{argmin}} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta^R)(u,v)$

9 expand(P,s,t)

The proof of partial correctness is the same as the one for the general policy iteration algorithm. To prove termination we slightly generalize Theorem 7.1.9.

Theorem 7.3.1. *For all $P \in \mathcal{P}$ and distinct (s_1,t_1), ..., $(s_n,t_n) \in \mathrm{dom}(P)$, if $\Delta(\boldsymbol{\mu}(\Theta_B^P)(s_i,t_i) < \boldsymbol{\mu}(\Theta_B^P)(s_i,t_i)$ for all $1 \leq i \leq n$, then*

$$\boldsymbol{\mu}(\Theta^{P[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}) \sqsubset \boldsymbol{\mu}(\Theta_B^P),$$

where $\pi_i = \underset{\omega \in V(\Omega(\tau(s_i),\tau(t_i)))}{\mathrm{argmin}} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta_B^P)(u,v)$ for all $1 \leq i \leq n$.

Proof. Let $P \in \mathcal{P}$ and (s_1,t_1), ..., $(s_n,t_n) \in \mathrm{dom}(P)$. Assume that (s_1,t_1), ..., (s_n,t_n) are distinct and $\Delta(\boldsymbol{\mu}(\Theta_B^P))(s_i,t_i) < \boldsymbol{\mu}(\Theta_B^P)(s_i,t_i)$ for all $1 \leq i \leq n$. By Lemma 6.2.6, it suffices to prove that $\Theta_B^{P[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}(\boldsymbol{\mu}(\Theta_B^P)) \sqsubset \boldsymbol{\mu}(\Theta_B^P)$, where $\pi_i = \underset{\omega \in V(\Omega(\tau(s_i),\tau(t_i)))}{\mathrm{argmin}} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta_B^P)(u,v)$ for all $1 \leq i \leq n$. Let x, $y \in S$. We distinguish the following cases.

- Assume $(x,y) = (s_i,t_i)$ for some $1 \leq i \leq n$. Then

$$\begin{aligned}
&\Theta_B^{P[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}(\boldsymbol{\mu}(\Theta_B^P))(x,y) \\
&\quad = \sum_{u,v \in S} P[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n](x,y)(u,v)\ \boldsymbol{\mu}(\Theta_B^P)(u,v)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{u,v \in S} \pi_i(u,v)\ \boldsymbol{\mu}(\Theta_B^P)(u,v) \\
&= \min_{\omega \in V(\Omega(\tau(s_i),\tau(t_i)))} \sum_{u,v \in S} \omega(u,v)\ \boldsymbol{\mu}(\Theta_B^P)(u,v) \\
&= \Delta(\boldsymbol{\mu}(\Theta_B^P))(s_i,t_i) \\
&< \boldsymbol{\mu}(\Theta_B^P)(s_i,t_i) \\
&= \boldsymbol{\mu}(\Theta_B^P)(x,y).
\end{aligned}
$$

- Assume that $(x,y) \in S_0^2$. Then

$$\Theta_B^{P[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}(\boldsymbol{\mu}(\Theta_B^P))(x,y) = 0 = \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(x,y) = \boldsymbol{\mu}(\Theta_B^P)(x,y).$$

- Assume that $(x,y) \in B$. Then

$$\Theta_B^{P[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}(\boldsymbol{\mu}(\Theta_B^P))(x,y) = 1 = \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(x,y) = \boldsymbol{\mu}(\Theta_B^P)(x,y).$$

- Assume that $(x,y) \notin S_0^2$, $(x,y) \notin B$ and $(x,y) \notin \mathrm{dom}(P)$. Since $(x,y) \neq (s_i,t_i)$ for all $1 \leq i \leq n$, we have that $(x,y) \notin \mathrm{dom}(P[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n])$ and

$$\Theta_B^{P[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}(\boldsymbol{\mu}(\Theta_B^P))(x,y) = 0 = \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(x,y) = \boldsymbol{\mu}(\Theta_B^P)(x,y).$$

- Otherwise, $(x,y) \neq (s_i,t_i)$ for all i satisfying $1 \leq i \leq n$ and $(x,y) \in \mathrm{dom}(P)$. In this case,

$$\Theta_B^{P[(s_1,t_1)/\pi_1]\ldots[(s_n,t_n)/\pi_n]}(\boldsymbol{\mu}(\Theta_B^P))(x,y)$$

$$
\begin{aligned}
&= \sum_{u,v \in S} P[(s_1,t_1)/\pi_1] \dots [(s_n,t_n)/\pi_n](x,y)(u,v)\ \boldsymbol{\mu}(\Theta_B^P)(u,v) \\
&= \sum_{u,v \in S} P(x,y)(u,v)\ \boldsymbol{\mu}(\Theta_B^P)(u,v) \qquad [(x,y) \neq (s_i,t_i) \text{ for all } 1 \leq i \leq n] \\
&= \Theta_B^P(\boldsymbol{\mu}(\Theta_B^P))(x,y) \\
&= \boldsymbol{\mu}(\Theta_B^P)(x,y).
\end{aligned}
$$

□

Similar to the general policy iteration algorithm, it is unknown if there exists an exponential lower bound for the general partial policy iteration algorithm.

8 Distance One for Labelled Markov Chains

In this chapter, we present a characterization of D_1, the set of all state pairs which have distance one, as a greatest fixed point of a function. We then use this characterization to develop an algorithm that decides distance one in $O(m^2)$ time, where m is the number of transitions of the labelled Markov chain. Finally, we propose three new algorithms to compute the probabilistic bisimilarity distances, where all three have incorporated the new procedure of deciding distance one. The algorithms presented in Chapter 6 and Chapter 7 do not decide distance one at first and can only handle labelled Markov chains up to 150 states in a reasonable amount of time. For one such labelled Markov chain, our implementation of the algorithms takes more than 49 hours. It is shown in Chapter 9 that our new algorithms, with deciding zero and one at first, takes 13 milliseconds instead of 49 hours. Furthermore, these new algorithms can compute distances for labelled Markov chains with more than 10,000 states in less than 50 minutes.

8.1 Characterization of Distance One

Recall that D_1 is defined (see in Section 6.1) as the set of of all state pairs which have probabilistic bisimilarity distance one. In this section we present a characterization of D_1 as a greatest fixed point of the function of Definition 8.1.1.

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 $\text{support}(\omega) \subseteq D_1$.

Definition 8.1.1. The function $\Gamma : 2^{S^2} \to 2^{S^2}$ is defined by

$$\Gamma(X) = S_1^2 \cup \{\, (s,t) \in S_?^2 \mid \forall \omega \in \Omega(\tau(s), \tau(t)) : \text{support}(\omega) \subseteq X \,\}.$$

Next, we show that 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, for example, [28, Example 2.6(2)]) and the function Γ is monotone, we can conclude from the Knaster-Tarski fixed point theorem (Theorem 2.1.8) that Γ admits a greatest fixed point.

Proposition 8.1.2. *The function Γ is monotone.*

Proof. Let X, $Y \in 2^{S^2}$ with $X \subseteq Y$. Let $(s,t) \in \Gamma(X)$. We distinguish two cases.

- If $(s,t) \in S_1^2$ then obviously $(s,t) \in \Gamma(Y)$.
- If $(s,t) \in S_?^2$ then

$$\forall \omega \in \Omega(\tau(s), \tau(t)) : \text{support}(\omega) \subseteq X$$

$$\text{implies} \quad \forall \omega \in \Omega(\tau(s), \tau(t)) : \text{support}(\omega) \subseteq Y \qquad [X \subseteq Y]$$

$$\text{implies} \quad (s,t) \in \Gamma(Y).$$

□

We denote the greatest fixed point of Γ by $\boldsymbol{\nu}(\Gamma)$. Next, we show that D_1 is a fixed point of Γ.

Proposition 8.1.3. $D_1 = \Gamma(D_1)$.

Proof. For all s, $t \in S$,

$(s,t) \in \Gamma(D_1)$

$$\text{iff} \quad (s,t) \in S_1^2 \vee \forall \omega \in \Omega(\tau(s), \tau(t)) : \text{support}(\omega) \subseteq D_1$$

$$\text{iff} \quad \ell(s) \neq \ell(t) \vee \forall \omega \in \Omega(\tau(s), \tau(t)) : \text{support}(\omega) \subseteq D_1$$

$$\text{iff} \quad \ell(s) \neq \ell(t) \vee \forall \omega \in \Omega(\tau(s), \tau(t)) : \forall (u,v) \in \text{support}(\omega) : (u,v) \in D_1$$

$$\text{iff} \quad \ell(s) \neq \ell(t) \vee \forall \omega \in \Omega(\tau(s), \tau(t)) : \forall (u,v) \in \text{support}(\omega) : \boldsymbol{\mu}(\Delta)(u,v) = 1$$

$$\text{iff} \quad \ell(s) \neq \ell(t) \vee \forall \omega \in \Omega(\tau(s), \tau(t)) : \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) = 1$$

$$\text{iff} \quad \ell(s) \neq \ell(t) \vee \min_{\omega \in \Omega(\tau(s), \tau(t))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) = 1 \qquad \left[\text{since } \sum_{u,v \in S} \omega(u,v) = 1\right]$$

$$\text{iff} \quad \Delta(\boldsymbol{\mu}(\Delta))(s,t) = 1$$

$$\text{iff} \quad \boldsymbol{\mu}(\Delta)(s,t) = 1$$

$$\text{iff} \quad (s,t) \in D_1.$$

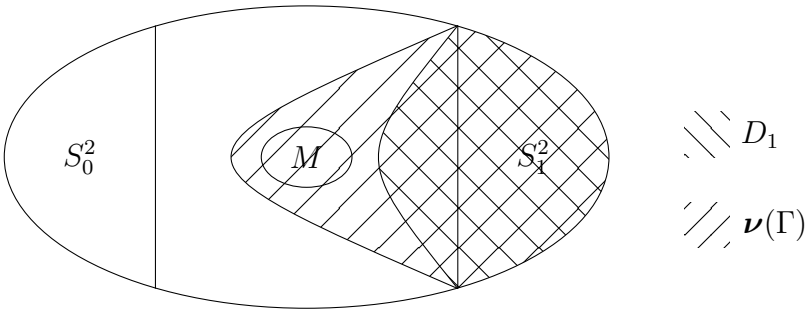
□

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 point of Γ, it remains to show that $\boldsymbol{\nu}(\Gamma) \subseteq D_1$, which is equivalent to the following.

Proposition 8.1.4. $\boldsymbol{\nu}(\Gamma) \setminus D_1 = \emptyset$.

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

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



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

$$M \subseteq \boldsymbol{\nu}(\Gamma) \setminus D_1. \tag{8.1}$$

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

$$M \subseteq \boldsymbol{\nu}(\Gamma) = \Gamma(\boldsymbol{\nu}(\Gamma)) \subseteq S_1^2 \cup S_?^2. \tag{8.2}$$

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

$$(s,t) \in \boldsymbol{\nu}(\Gamma) \wedge (s,t) \notin D_1 \qquad [(8.1)]$$

$$\text{implies} \quad (s,t) \in \Gamma(\boldsymbol{\nu}(\Gamma)) \wedge (s,t) \notin S_1^2$$

$$\text{implies} \quad \forall \omega \in \Omega(\tau(s), \tau(t)) : \text{support}(\omega) \subseteq \boldsymbol{\nu}(\Gamma). \tag{8.3}$$

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

$$\omega_{s,t} = \underset{\omega \in \Omega(\tau(s),\tau(t))}{\text{argmin}} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v). \tag{8.4}$$

We distinguish the following two cases.

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

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

  By (8.3), we have that $\text{support}(\omega_{s,t}) \subseteq \boldsymbol{\nu}(\Gamma)$. Since $\text{support}(\omega_{s,t}) \cap D_1 \neq \emptyset$ by assumption, we can conclude that $p > 0$. Again using the fact that $\text{support}(\omega_{s,t}) \subseteq \boldsymbol{\nu}(\Gamma)$, we have that

  $$\sum_{(u,v) \in \boldsymbol{\nu}(\Gamma) \setminus D_1} \omega_{s,t}(u,v) = 1 - p. \tag{8.5}$$

  Furthermore,

  $$\begin{aligned}
  m &= \boldsymbol{\mu}(\Delta)(s,t) \\
  &= \Delta(\boldsymbol{\mu}(\Delta))(s,t) \\
  &= \min_{\omega \in \Omega(\tau(s),\tau(t))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) \\
  &= \sum_{u,v \in S} \omega_{s,t}(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) \qquad [(8.4)] \\
  &= \sum_{(u,v) \in \boldsymbol{\nu}(\Gamma)} \omega_{s,t}(u,v)\, \boldsymbol{\mu}(\Delta)(u,v) \qquad [(8.3)]
  \end{aligned}$$

$$
\begin{aligned}
&= \sum_{(u,v)\in\boldsymbol{\nu}(\Gamma)\cap D_1} \omega_{s,t}(u,v)\,\boldsymbol{\mu}(\Delta)(u,v) + \sum_{(u,v)\in\boldsymbol{\nu}(\Gamma)\setminus D_1} \omega_{s,t}(u,v)\,\boldsymbol{\mu}(\Delta)(u,v)\\
&= p + \sum_{(u,v)\in\boldsymbol{\nu}(\Gamma)\setminus D_1} \omega_{s,t}(u,v)\,\boldsymbol{\mu}(\Delta)(u,v)\\
&\geq p + (1-p)m.
\end{aligned}
$$

The last step follows from (8.5) 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 (8.1).

- Otherwise, $\text{support}(\omega_{s,t}) \cap D_1 = \emptyset$ 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 2.1.14 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 (8.2).

  Next, we prove that M is a probabilistic bisimulation. Let $(s,t) \in M$. Since $M \subseteq \boldsymbol{\nu}(\Gamma) \setminus D_1$ by (8.1), 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 $\text{support}(\omega_{s,t}) \subseteq \boldsymbol{\nu}(\Gamma)$ by (8.3) and $\text{support}(\omega_{s,t}) \cap D_1 = \emptyset$ by the assumption, we have $\text{support}(\omega_{s,t}) \subseteq \boldsymbol{\nu}(\Gamma) \setminus D_1$. Since

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

and $\boldsymbol{\mu}(\Delta)(u,v) \geq m$ for all $(u,v) \in \boldsymbol{\nu}(\Gamma) \setminus D_1$, and $\text{support}(\omega_{s,t}) \subseteq \boldsymbol{\nu}(\Gamma) \setminus D_1$,

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

Proof. Immediate consequence of Proposition 8.1.3 and 8.1.4. □

8.2 An Algorithm of Deciding for Distance One

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

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 8.2.1. The function $\beth : 2^{S^2} \to 2^{S^2}$ is defined by

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

The next theorem follows from Theorem 8.1.5.

Theorem 8.2.2. $D_{<1} = \boldsymbol{\mu}(\beth)$.

Proof. First, we prove that for all $X \subseteq S^2$, X is a fixed point of $\Game$ if and only if $S^2 \setminus X$ is a fixed point of Γ. Let $X \subseteq S^2$. Then

$$\begin{aligned} \Game(X) = X \quad &\text{iff} \quad S^2 \setminus \Gamma(S^2 \setminus X) = X \\ &\text{iff} \quad \Gamma(S^2 \setminus X) = S^2 \setminus X. \end{aligned}$$

Next, we observe that $S^2 \setminus \boldsymbol{\nu}(\Gamma)$ is a fixed point of $\Game$, since $\boldsymbol{\nu}(\Gamma)$ is a fixed point of Γ.

Let X be a fixed point of $\Game$. Then $S^2 \setminus X$ is a fixed point of Γ. Hence, $S^2 \setminus X \subseteq \boldsymbol{\nu}(\Gamma)$. Therefore, $S^2 \setminus \boldsymbol{\nu}(\Gamma) \subseteq X$. Consequently, $\boldsymbol{\mu}(\Game) = S^2 \setminus \boldsymbol{\nu}(\Gamma)$.

Finally,

$$\begin{aligned} D_{<1} &= S^2 \setminus D_1 \\ &= S^2 \setminus \boldsymbol{\nu}(\Gamma) \qquad \text{[Theorem 8.1.5]} \\ &= \boldsymbol{\mu}(\Game) \qquad \text{[as proved above]} \end{aligned}$$

□

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. We present the graph induced by the labelled Markov chain as follows.

Definition 8.2.3. The directed graph $G = (V, E)$ is defined by

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

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 in the beginning of Section 8.1, 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 the following proposition in the proof of Proposition 8.2.5.

Proposition 8.2.4. *For all* μ, $\nu \in \text{Distr}(S)$ *and* $X \subseteq S^2$,

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

Proof. Let μ, $\nu \in \text{Distr}(S)$ and $X \subseteq S^2$. We prove two implications. We first show that $\text{support}(\mu) \times \text{support}(\nu) \not\subseteq X$ implies $\exists \omega \in \Omega(\mu,\nu) : \text{support}(\omega) \not\subseteq X$. Assume $\text{support}(\mu) \times \text{support}(\nu) \not\subseteq X$. Then there exists $(u,v) \in \text{support}(\mu) \times \text{support}(\nu)$ such that $(u,v) \notin X$. Hence, $\mu(u) > 0$ and $\nu(v) > 0$. Let

$$\mu'(s) = \begin{cases} \mu(u) - \min(\mu(u), \nu(v)) & \text{if } s = u \\ \mu(s) & \text{otherwise} \end{cases}$$

and

$$\nu'(s) = \begin{cases} \nu(v) - \min(\mu(u), \nu(v)) & \text{if } s = v \\ \nu(s) & \text{otherwise} \end{cases}$$

As

$$\sum_{s \in S} \mu'(s) = 1 - \min(\mu(u), \nu(v)) = \sum_{s \in S} \nu'(s),$$

we can find a $\omega' \in S^2 \to [0, 1]$ by applying Hitchcock's North West corner rule [51] such that $\sum_{t \in S} \omega'(s, t) = \mu'(s)$ and $\sum_{s \in S} \omega'(s, t) = \nu'(t)$. Let

$$\omega(s, t) = \begin{cases} \min(\mu(u), \nu(v)) & \text{if } s = u \text{ and } t = v \\ \omega'(s, t) & \text{otherwise} \end{cases}$$

By construction, $\omega \in \Omega(\mu, \nu)$. Since $\min(\mu(u), \nu(v)) > 0$, we have that $(u, v) \in \text{support}(\omega)$. As $(u, v) \notin X$, we obtain $\text{support}(\omega) \not\subseteq X$.

It remains to prove that if there exists an $\omega \in \Omega(\mu, \nu)$ such that $\text{support}(\omega) \not\subseteq X$, then $\text{support}(\mu) \times \text{support}(\nu) \not\subseteq X$. Assume there exists an $\omega \in \Omega(\mu, \nu)$ such that $\text{support}(\omega) \not\subseteq X$. There must be a pair of states $(u, v) \in \text{support}(\omega)$ and $(u, v) \notin X$. Thus, $\omega(u, v) > 0$. We have

$$\mu(u) = \sum_{s \in S} \omega(u, s) \geq \omega(u, v) > 0.$$

So $u \in \text{support}(\mu)$. Similarly, we can obtain that $v \in \text{support}(\nu)$. Thus, $(u, v) \in \text{support}(\mu) \times \text{support}(\nu)$. □

It is well known that the vertices reachable from S_0^2 can be expressed as the least fixed point of Υ (see, for example, [34]).

Proposition 8.2.5. $\boldsymbol{\mu}(\daleth) = \{\, (s,t) \mid (s,t) \textit{ is reachable from some } (u,v) \in S_0^2 \,\}$.

Proof. For all $X \subseteq S^2$,

$$
\begin{aligned}
&\daleth(X) \\
&\quad = S^2 \setminus \Gamma(S^2 \setminus X) \\
&\quad = S_0^2 \cup \{\, (s,t) \in S_?^2 \mid \exists \omega \in \Omega(\tau(s), \tau(t)) : \mathrm{support}(\omega) \not\subseteq S^2 \setminus X \,\} \\
&\quad = S_0^2 \cup \{\, (s,t) \in S_?^2 \mid \mathrm{support}(\tau(s)) \times \mathrm{support}(\tau(t)) \not\subseteq S^2 \setminus X \,\} \\
&\qquad\qquad \text{[Proposition 8.2.4]} \\
&\quad = S_0^2 \cup \{\, (s,t) \in S_?^2 \mid \mathrm{support}(\tau(s)) \times \mathrm{support}(\tau(t)) \cap X \neq \emptyset \,\} \\
&\quad = S_0^2 \cup \{\, (s,t) \in S_?^2 \mid \exists (u,v) \in X : (u,v) \in \mathrm{support}(\tau(s)) \times \mathrm{support}(\tau(t)) \,\} \\
&\quad = S_0^2 \cup \{\, (s,t) \in S_?^2 \mid \exists (u,v) \in X : \tau(s)(u) > 0 \wedge \tau(t)(v) > 0 \,\} \\
&\quad = S_0^2 \cup \{\, (s,t) \in V \mid \exists (u,v) \in X : \langle (u,v), (s,t) \rangle \in E \,\}.
\end{aligned}
$$

□

Theorem 8.2.6. *Distance smaller than one can be decided in $O(m^2)$ time.*

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

1. Decide distance zero.

2. Breadth-first search of the graph G defined in Definition 8.2.3, with the queue initially containing the pairs of states that have distance zero.

By Theorem 8.2.2 and Proposition 8.2.5, 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.

We have discussed in Chapter 4 that distance zero, that is, probabilistic bisimilarity, can be decided in $O(m \log n)$ time as shown by Derisavi, Hermanns and Sanders in [30]. The directed graph G has n^2 vertices and m^2 edges. Hence, breadth-first search takes $O(n^2 + m^2)$ time. Since each state of a labelled Markov chain has at least one transition, we have that $n \in O(m)$. Hence, breadth-first search takes $O(m^2)$ time. □

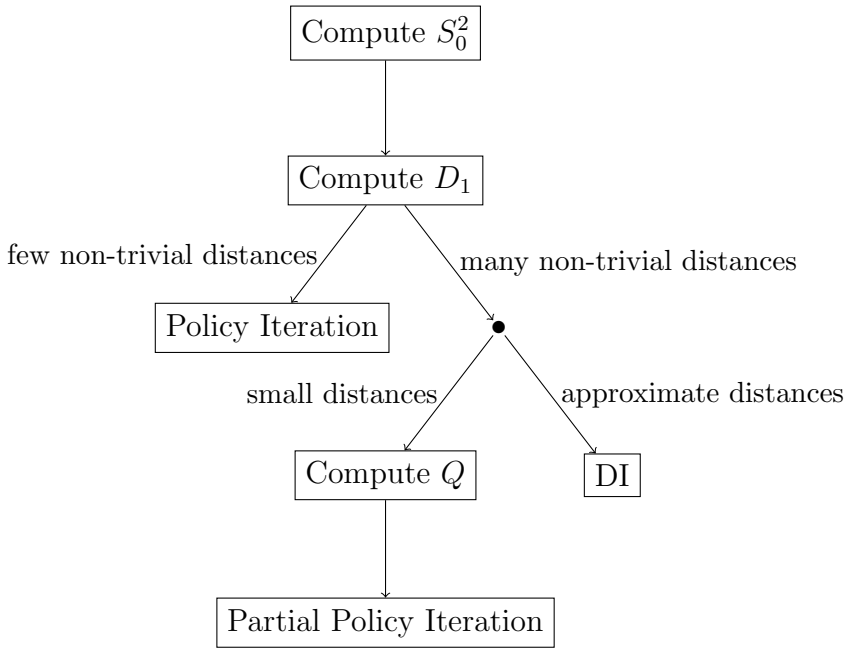
Theorem 8.2.7. *Distance one can be decided in $O(m^2)$ time.*

Proof. As we have shown in Theorem 8.2.6, distance smaller than one can be decided in $O(m^2)$ time. Hence, distance one can be decided in $O(m^2)$ time as well. □

8.3 Three New Algorithms

In this section, we present three new algorithms for which the decision procedure for distance one is a new ingredient.

The decision procedures for distance zero and one can be used to compute or approximate probabilistic bisimilarity distances as indicated below. We call the distances non-trivial if the distances are greater than zero and smaller than one.



Once we have computed the sets S_0^2 and D_1 of state pairs that have distance zero and 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 policy iteration algorithms introduced in Chapter 6 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 partial policy iteration algorithms introduced in Chapter 7 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 [12].

8.3.1 New Policy Iteration

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, Hermanns and Sanders [30] (step 1) and simple policy iteration algorithm due to Bacci, Bacci, Larsen and Mardare [3] (step 3), by incorporating our decision procedure (step 2) as follows.

1. Decide distance zero.
2. Decide distance one.
3. Policy iteration.

In this new algorithm, we not only decide distance zero, but also distance one, before running policy iteration. By substituting D_1 for B, Theorem 6.2.3 can be used to show that if the algorithm terminates then it computes the probabilistic bisimilarity distances. Similarly, Theorem 6.2.7, by initializing D_1 for B, can be used to show that the algorithm does terminate.

As we have already discussed in the previous section, step 1 and 2 are polynomial time. However, if we use the simple policy iteration in step 3, it may take at least

exponential time in the worst case, as we have shown in Section 6.3. Hence, the overall algorithm is exponential time.

In step 3, we can also initialize B with D_1 and run either the general policy iteration algorithm or the partial policy iteration algorithms. If the third step is a general (partial) policy iteration algorithm, of which the time complexity is unknown, the time complexity of the overall algorithm is unknown as well.

8.3.2 Algorithm for Small Distances

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 often 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

$$Q = \{\, (s,t) \in S^2 \setminus (S_0^2 \cup D_1) \mid \Delta(d)(s,t) \leq \varepsilon \,\}$$

where

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

4. 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 8.3.1. In step 4, we use this set as the query set and initialize B with D_1 to run the simple partial policy iteration algorithm by Bacci *et al.* [3].

Proposition 8.3.1. *Let d be the distance function defined in step 3. For all $(s,t) \in S^2 \setminus (S_0^2 \cup D_1)$, if $\boldsymbol{\mu}(\Delta)(s,t) \leq \varepsilon$, then $\Delta(d)(s,t) \leq \varepsilon$.*

Proof. Let $(s,t) \in S^2 \setminus (S_0^2 \cup D_1)$. Suppose $\boldsymbol{\mu}(\Delta)(s,t) \leq \varepsilon$. Hence,

$$\begin{aligned} & d \sqsubseteq \boldsymbol{\mu}(\Delta) \\ \text{implies} \quad & \Delta(d) \sqsubseteq \boldsymbol{\mu}(\Delta) \qquad \text{[Proposition 2.1.13]} \\ \text{implies} \quad & \Delta(d)(s,t) \leq \boldsymbol{\mu}(\Delta)(s,t) \\ \text{implies} \quad & \Delta(d)(s,t) \leq \varepsilon \qquad [\boldsymbol{\mu}(\Delta)(s,t) \leq \varepsilon] \end{aligned}$$

□

In this algorithm, we not only decide distance zero, but also distance one, before running simple partial policy iteration. Let B be D_1. Theorem 7.1.8 shows that if

the algorithm terminates then it computes the probabilistic bisimilarity distances. In Section 7.1, the explanation before Theorem 7.1.9, together with the theorem, show that the algorithm does terminate.

As we have seen in Section 4.2, step 1 and 2 take polynomial time. In step 3, computing $\Delta(d)$ corresponds to solving a minimum cost network flow problem as discussed in Section 4.2. Such a problem can be solved in polynomial time using, for example, Orlin's network simplex algorithm [70]. As we have shown in Section 7.2, step 4 takes at least exponential time in the worst case. Therefore, the overall algorithm is exponential time.

Note that we can also initialize B with D_1 and run the general partial policy iteration algorithm in step 4. As the complexity of the general partial policy iteration algorithm is unknown, the time complexity of this overall algorithm for small distances is unknown as well.

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

$$u(s,t) = \begin{cases} 0 & \text{if } (s,t) \in S_0^2 \\ 1 & \text{otherwise} \end{cases}$$

$$\begin{array}{l} \texttt{repeat} \\ \quad \texttt{for each } (s,t) \in S^2 \setminus (S_0^2 \cup D_1) \\ \quad\quad \texttt{if } l(s,t) \neq u(s,t) \\ \quad\quad\quad l(s,t) = \Delta(l)(s,t) \\ \quad\quad\quad u(s,t) = \Delta(u)(s,t) \\ \texttt{until } \|l - u\| \leq \alpha \end{array}$$

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 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 total correctness of the above algorithm, we annotate it with the following assertions.

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

2 $u(s,t) = \begin{cases} 0 & \text{if } (s,t) \in S_0^2 \\ 1 & \text{otherwise} \end{cases}$

$\{\mathbf{0} \sqsubseteq l \sqsubseteq \boldsymbol{\mu}(\Delta) \sqsubseteq u \sqsubseteq \mathbf{1}\}$

3 $n = 0$

4 repeat

$\{\Delta^n(\mathbf{0}) \sqsubseteq l \sqsubseteq \boldsymbol{\mu}(\Delta) \sqsubseteq u \sqsubseteq \Delta_1^n(\mathbf{1})\}$

5 for each $(s,t) \in S^2 \setminus (S_0^2 \cup D_1)$

$\{\Delta_1^n(\mathbf{0})(s,t) \leq l(s,t) \leq \boldsymbol{\mu}(\Delta)(s,t) \leq u(s,t) \leq \Delta_1^n(\mathbf{1})(s,t)\}$

6 if $l(s,t) \neq u(s,t)$

7 $l(s,t) = \Delta(l)(s,t)$

$\{\Delta_1^{n+1}(\mathbf{0})(s,t) \leq l(s,t) \leq \boldsymbol{\mu}(\Delta)(s,t) \leq u(s,t) \leq \Delta_1^n(\mathbf{1})(s,t)\}$

8 $u(s,t) = \Delta(u)(s,t)$

$\{\Delta_1^{n+1}(\mathbf{0})(s,t) \leq l(s,t) \leq \boldsymbol{\mu}(\Delta)(s,t) \leq u(s,t) \leq \Delta_1^{n+1}(\mathbf{1})(s,t)\}$

$\{\Delta_1^{n+1}(\mathbf{0}) \sqsubseteq l \sqsubseteq \boldsymbol{\mu}(\Delta) \sqsubseteq u \sqsubseteq \Delta_1^{n+1}(\mathbf{1})\}$

9 $n = n + 1$

10 until $\|l - u\| \leq \alpha$

$\{l \sqsubseteq \boldsymbol{\mu}(\Delta) \sqsubseteq u \wedge \|l - u\| \leq \alpha\}$

First, we prove that the above assertions hold. The assertion after line 2 follows

immediately from the definitions of $\mathbf{0}$, $\mathbf{1}$, u and l. This assertion also implies that the loop invariant of the outer loop holds the first time we reach line 5.

If the assertion before line 5 holds, then the assertion after line 5 holds as well. Note that the loop at line 5–8 iterates over $(s,t) \in S^2 \setminus (S_0^2 \cup D_1)$. In that case, $\Delta_1(d)(s,t) = \Delta(d)(s,t)$ for all $d \in [0,1]^{S^2}$ according to Definition 2.1.31. To prove that the assertion after line 8 holds, we distinguish the following two cases.

(i) If $l(s,t) = u(s,t)$ then $l(s,t) = \boldsymbol{\mu}(\Delta)(s,t)$ and $u(s,t) = \boldsymbol{\mu}(\Delta)(s,t)$ since $l(s,t) \leq \boldsymbol{\mu}(\Delta)(s,t) \leq u(s,t)$. Since Δ_1 is monotone (Theorem 2.1.32(a)), we can conclude from $\Delta_1^n(\mathbf{0})(s,t) \leq l(s,t) \leq \boldsymbol{\mu}(\Delta)(s,t)$ that

$$\Delta_1^{n+1}(\mathbf{0})(s,t) \leq \Delta_1(l)(s,t) \leq \Delta_1(\boldsymbol{\mu}(\Delta))(s,t) =$$

$$\Delta(\boldsymbol{\mu}(\Delta))(s,t) = \boldsymbol{\mu}(\Delta)(s,t) = l(s,t).$$

Similarly, $\boldsymbol{\mu}(\Delta)(s,t) \leq u(s,t) \leq \Delta^n(\mathbf{1})(s,t)$ implies that

$$u(s,t) = \boldsymbol{\mu}(\Delta)(s,t) = \Delta(\boldsymbol{\mu}(\Delta))(s,t) = \Delta_1(\boldsymbol{\mu}(\Delta))(s,t) \leq$$

$$\Delta_1(u)(s,t) \leq \Delta_1(\Delta^n(\mathbf{1}))(s,t) = \Delta^{n+1}(\mathbf{1})(s,t).$$

Their conjunction implies the assertion after line 8.

(ii) Assume that $l(s,t) \neq u(s,t)$. We have that $\Delta_1^n(\mathbf{0})(s,t) \leq l(s,t) \leq \boldsymbol{\mu}(\Delta)(s,t)$ implies

$$\Delta_1^{n+1}(\mathbf{0})(s,t) \leq \Delta_1(l)(s,t) \leq \Delta_1(\boldsymbol{\mu}(\Delta))(s,t) = \Delta(\boldsymbol{\mu}(\Delta))(s,t) = \boldsymbol{\mu}(\Delta)(s,t).$$

because Δ_1 is monotone (Theorem 2.1.32(a)). As a consequence, the assertion after line 7 is true. Using the monotonicity of Δ_1 (Theorem 2.1.32(a)),

$\boldsymbol{\mu}(\Delta)(s,t) \leq u(s,t) \leq \Delta_1^n(\mathbf{1})(s,t)$ implies $\boldsymbol{\mu}(\Delta)(s,t) = \Delta(\boldsymbol{\mu}(\Delta))(s,t) = \Delta_1(\boldsymbol{\mu}(\Delta))(s,t) \leq \Delta_1(u)(s,t) \leq \Delta_1^{n+1}(\mathbf{1})(s,t)$. Hence, the assertion after line 8 is true.

From the definition of u and l, we can conclude that $u(s,t) = l(s,t)$ for all $(s,t) \in S_0^2 \cup D_1$. As in case (i) above, the assertion before line 6 implies the one after line 8.

From the assertion after line 8 we can deduce the assertion on the next line. We can conclude that the loop invariant of the outer loop is maintained. The postcondition after line 10 easily follows if the loop terminates.

Next we prove that the outer loop terminates. According to Theorem 2.1.32(d) and 2.1.32(e), $\boldsymbol{\mu}(\Delta) = \sup_{m \in \mathbb{N}} \Delta_1^m(\mathbf{0})$. Since Δ_1 is monotone and, hence, the sequence $(\Delta_1^m(\mathbf{0}))_{m \in \mathbb{N}}$ is increasing,

$$\exists M \in \mathbb{N} : \forall m \geq M : \|\Delta_1^m(\mathbf{0}) - \boldsymbol{\mu}(\Delta)\| \leq \tfrac{\alpha}{2}.$$

By Theorem 2.1.32(c), 2.1.32(d) and 2.1.32(f), $\boldsymbol{\mu}(\Delta) = \inf_{n \in \mathbb{N}} \Delta_1^n(\mathbf{1})$. Since Δ_1 is monotone and, hence, the sequence $(\Delta_1^n(\mathbf{1}))_{n \in \mathbb{N}}$ is decreasing,

$$\exists N \in \mathbb{N} : \forall n \geq N : \|\Delta_1^n(\mathbf{1}) - \boldsymbol{\mu}(\Delta)\| \leq \tfrac{\alpha}{2}.$$

Hence,

$$\begin{aligned} & \|\Delta_1^{\max(M,N)}(\mathbf{0}) - \Delta_1^{\max(M,N)}(\mathbf{1})\| \\ \leq \; & \|\Delta_1^{\max(M,N)}(\mathbf{0}) - \boldsymbol{\mu}(\Delta)\| + \|\boldsymbol{\mu}(\Delta) - \Delta_1^{\max(M,N)}(\mathbf{1})\| \end{aligned}$$

$$\begin{aligned} &\leq \frac{\alpha}{2} + \frac{\alpha}{2} \\ &= \alpha. \end{aligned}$$

Because $\Delta_1^{\max(M,N)}(\mathbf{0}) \sqsubseteq l$ and $u \sqsubseteq \Delta_1^{\max(M,N)}(\mathbf{1})$, we can conclude that the outer loop terminates after at most $\max(M, N)$ iterations.

9 Experimental Results

The algorithms considered in this chapter are the following.

- The algorithms which applies the first order theory over the reals described in Section 3.1 and the ellipsoid method described in Section 4.2.
- $D_0 + D_1$: the decision procedure of the number of non-trivial distances (the first two steps of the new algorithms in Section 8.3).
- $D_0 + \text{SPI}$: the original state of the art algorithm which decides distance zero before running the simple policy iteration algorithm by Bacci *et al.* [3] (see Section 6.2).
- $D_0 + \text{GPI}$: the algorithm which decides distance zero before running the general policy iteration algorithm (see Section 6.4).
- $D_0 + D_1 + \text{SPI}$: the algorithm which decides both distance zero and distance one before running the simple policy iteration algorithm (see Section 8.3.1).
- $D_0 + D_1 + \text{GPI}$: the algorithm which decides both distance zero and distance

one before running the general policy iteration algorithm (see Section 8.3.1).

- $D_0 +$ SPPI: the modified algorithm by Bacci *et al.* [3] with the on-the-fly optimization which decides distance zero before running the simple partial policy iteration algorithm (see Section 7.1).

- $D_0 +$ GPPI: the algorithm which decides distance zero before running the general partial policy iteration algorithm (see Section 7.3).

- $D_0 + D_1 +$ SPPI: the algorithm which decides both distance zero and distance one before running the simple partial policy iteration algorithm (see Section 8.3.1).

- $D_0 + D_1 +$ GPPI: the algorithm which decides both distance zero and distance one before running the general partial policy iteration algorithm (see Section 8.3.1).

- $D_0 + D_1 + Q +$ SPPI: the algorithm which computes all distances smaller than a chosen $\varepsilon > 0$ (see Section 8.3.2).

- $D_0 + D_1 +$ DI: the algorithm which approximates the distances with accuracy α (see Section 8.3.3).

We implemented all the algorithms in Java[viii] and ran the implementations on several labelled Markov chains. These labelled Markov chains model randomized

[viii] https://bitbucket.org/discoveri/probabilistic-bisimilarity-distances

algorithms and probabilistic protocols that are part of the distribution of probabilistic model checkers such as PRISM [63] and jpf-probabilistic [93].

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. The cut-off time is set to be 60 hours. Our experiments were run on an Intel® Xeon® CPU X5660, using CentOS 7.5 and the Java 64-bit virtual machine version 1.8.0_101.

Whereas the original state of the art algorithm, $D_0 + \text{SPI}$, 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 hours, whereas our new algorithm takes only 13 milliseconds.

To decide distance zero, we implemented the algorithm to decide probabilistic bisimilarity due to Derisavi, Hermanns and Sanders [30] in Java. We implemented our algorithm to decide distance one, described in the proof of Theorem 8.2.7 and Theorem 8.2.6.

9.1 First Order Theory over the Reals and the Ellipsoid Method

We consider the randomized quicksort algorithm, an implementation of which is part of jpf-probabilistic [93]. 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.

The ellipsoid method takes more than 43 hours for the labelled Markov chain with 28 states, making it five orders of magnitude slower than the policy iteration algorithms, which take less than a dozen seconds. As the first order theory over the reals algorithm and the ellipsoid algorithm are not practical, we did not run them on the other labelled Markov chains.

9.2 Deciding Non-trivial Distances

We compute the number of non-trivial distances for three models: the bounded retransmission protocol by Helmink, Sellink and Vaandrager [48], the synchronous leader election protocol of Itai and Rodeh [54] and the randomized self-stabilising

algorithm due to Herman [50]. Since there are no non-trivial distances in the first two examples, deciding distance zero and one suffices to compute all the distances. The example of the randomized self-stabilising algorithm shows the importance of developing new algorithms when the number of non-trivial distances is so large that the policy iteration algorithms are infeasible.

9.2.1 Bounded Retransmission Protocol

N	M	$\|S\|$	$\|D_0\|$	$\|D_1\|$	$\|S_1^2\|$	$D_0 + D_1$	STD
16	2	677	456,977	1,352	1,352	3.0 s	0.2 s
16	3	886	783,226	1,770	1,770	8.6 s	0.7 s
16	4	1,095	1,196,837	2,188	2,188	17.5 s	1.2 s
16	5	1,304	1,697,810	2,606	2,606	22.8 s	1.7 s
32	2	1,349	1,817,105	2,696	2,696	24.7 s	1.2 s
32	3	1,766	3,115,226	3,530	3,530	69.7 s	4.7 s
32	4	2,183	4,761,125	4,364	4,364	141.0 s	6 s
32	5	2,600	6,754,802	5,198	5,198	208.6 s	10.7 s
64	2	2,693	7,246,865	5,384	5,384	235.2 s	12 s
64	3	3,526	12,425,626	7,050	7,050	616.4 s	9 s

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 above. The algorithm can handle systems up to 3,526 states within 11 minutes. 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. The standard deviation of the running time of $D_0 + D_1$ is denoted STD in the tables.

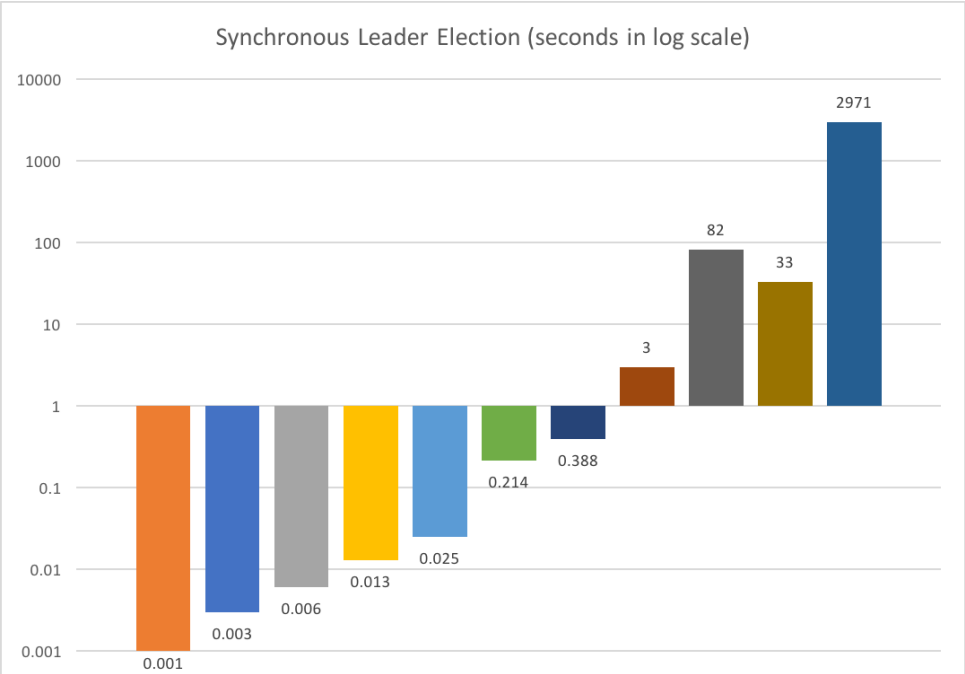
9.2.2 Synchronous Leader Election

The example we consider here is the synchronous leader election protocol. 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, which first decides distance zero and one, terminates without running the 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 table.

The biggest system the original simple policy iteration algorithm ($D_0 + \mathrm{SPI}$) can handle is the one with 147 states ($N = 3$ and $K = 4$) and it takes more than 49 hours. The algorithm ($D_0 + \mathrm{GPI}$) takes about 10 hours. In contrast, our procedure of deciding the non-trivial distances terminates within only 13 milliseconds.

N	K	$\lvert S\rvert$	$\lvert D_0\rvert$	$\lvert D_1\rvert$	$\lvert S_1^2\rvert$	$D_0 + D_1$	STD
3	2	26	122	554	50	1 ms	0.3 ms
3	4	147	7,419	14,190	292	13 ms	2.8 ms
3	6	459	88,671	122,010	916	214 ms	17 ms
3	8	1,059	508,851	612,630	2,116	3 s	98 ms
4	2	61	459	3,262	120	3 ms	0.4 ms
4	4	812	145,780	513,564	1,622	388 ms	13 ms
4	6	3,962	4,350,292	11,347,152	7,922	82 s	3 s
4	8	12,400	46,198,188	107,561,812	24,798	2,971 s	98 s
5	2	141	2,399	17,482	280	6 ms	1 ms
5	4	4,244	3,318,662	14,692,874	8,486	33 s	0.8 s
6	2	335	14,327	97,898	668	25 ms	2 ms

The graph on the next page plots the running time of $D_0 + D_1$ for each labelled Markov chains shown in the above table. The vertical axis indicates the running time in seconds in logarithmic scale and the number of states increases from left to right. It can be seen that as the number of states increases, the running time of deciding non-trivial distances increases, whereas the only exception is the labelled Markov chain with 4244 states ($N = 5$ and $K = 4$) indicated by the second bar to the right.



9.2.3 Randomized Self-stabilising

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. 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. 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	$\|S\|$	non-trivial	$\|D_0\|$	$\|D_1\|$	$\|S_1^2\|$	$D_0 + D_1$	STD
3	8	12	38	14	14	1.00 ms	0.46 ms
5	32	280	304	440	440	6.06 ms	2.16 ms
7	128	11,032	2,160	3,192	3,192	0.77 s	0.03 s
9	512	230,712	13,648	17,784	17,784	378.42 s	5.83 s

9.3 Policy Iteration Algorithms

In this section, we compare our new (partial) policy iteration algorithms which decide both distance zero and distance one first with the original policy iteration algorithms. We consider the randomized quicksort algorithm introduced in Section 9.1 and an example of two dies due to Knuth and Yao [62], one using only a fair coin and the other one using a biased coin. An implementation of the die algorithm is part of PRISM.

For the partial policy iteration algorithms we compute the distance for a single pair of states. From the results of the experiments, we make the following observations. Firstly, our new algorithms that decide the non-trivial distances first are faster than the ones that only decide distance zero first. Secondly, the general (partial) policy iteration algorithms are not necessarily faster than the simple (partial) policy iteration algorithms.

9.3.1 Randomized Quicksort

We denote the list size as L in the tables.

L	$\|S\|$	D_0	$D_0 + D_1$	non-trivial	$\|D_0\|$	$\|D_1\|$	$\|S_1^2\|$
4	10	0.17 ms	0.31 ms	28	42	30	18
5	28	0.39 ms	1 ms	262	332	190	54
6	82	0.7 ms	4 ms	2300	2750	1674	162

The list of size 4 gives rise to a labelled Markov chain with 10 states. We compare the running time of the new policy iteration algorithms which decide both distance zero and distance one first with the ones that only decide distance zero first. The last column of the table below is the total number of state pairs considered in the specific policy iteration algorithm.

Algorithm	Running time	STD	State pairs
$D_0 + \text{SPI}$	19 ms	4.7 ms	40
$D_0 + \text{GPI}$	15 ms	1 ms	40
$D_0 + D_1 + \text{SPI}$	14 ms	4 ms	28
$D_0 + D_1 + \text{GPI}$	11 ms	2 ms	28
$D_0 + \text{SPPI}$	11 ms	3 ms	10
$D_0 + \text{GPPI}$	13 ms	4ms	10
$D_0 + D_1 + \text{SPPI}$	10 ms	3 ms	10
$D_0 + D_1 + \text{GPPI}$	7 ms	0.7 ms	10

The list of size 5 gives rise to a labelled Markov chain with 28 states. The results for the policy iteration algorithms are collected in the following table.

Algorithm	Running time	STD	State pairs
$D_0 + \text{SPI}$	13 s	41 ms	398
$D_0 + \text{GPI}$	4 s	17 ms	398
$D_0 + D_1 + \text{SPI}$	9 s	57 ms	262
$D_0 + D_1 + \text{GPI}$	3 s	17 ms	262
$D_0 + \text{SPPI}$	49 ms	3 ms	12
$D_0 + \text{GPPI}$	49 ms	3 ms	12
$D_0 + D_1 + \text{SPPI}$	30 ms	4 ms	6
$D_0 + D_1 + \text{GPPI}$	27 ms	2 ms	6

The list of size 6 gives rise to a labelled Markov chain with 82 states. The results for the policy iteration algorithm are collected in the following table. The algorithm ($D_0 + \text{SPI}$) takes about 14 hours and our new algorithm which incorporates the decision procedure of distance one ($D_0 + D_1 + \text{SPI}$) takes less than 7 hours, while the general policy iteration algorithms ($D_0 + \text{GPI}$ and $D_0 + D_1 + \text{GPI}$) only take less than 30 minutes.

Algorithm	Running time	STD	State pairs
$D_0 + \text{SPI}$	14 hrs	2 min	3812
$D_0 + \text{GPI}$	27 min	2 s	3812
$D_0 + D_1 + \text{SPI}$	7 hrs	1 min	2300
$D_0 + D_1 + \text{GPI}$	21 min	3 s	2300
$D_0 + \text{SPPI}$	13 s	0.2 s	72
$D_0 + \text{GPPI}$	36 s	0.4 s	72
$D_0 + D_1 + \text{SPPI}$	9 s	0.2 s	44
$D_0 + D_1 + \text{GPPI}$	25 s	0.1 s	44

9.3.2 Dies

In the next experiment, we model two dies, one using only a fair coin and the other one using a biased coin with probability 0.51 for heads and 0.49 for tails. The goal is to compute the probabilistic bisimilarity distance between the two dies. The resulting labelled Markov chain has 20 states. There are 182 distances which are computed by the original policy iteration algorithms, while there are only 30 non-trivial distances.

Algorithm	Running time	STD	State pairs
$D_0 + \text{SPI}$	7 s	21 ms	182
$D_0 + \text{GPI}$	582 ms	2 ms	182
$D_0 + D_1 + \text{SPI}$	151 ms	18 ms	30
$D_0 + D_1 + \text{GPI}$	86 ms	3 ms	30
$D_0 + \text{SPPI}$	203 ms	19 ms	28
$D_0 + \text{GPPI}$	217 ms	3 ms	42
$D_0 + D_1 + \text{SPPI}$	52 ms	8 ms	14
$D_0 + D_1 + \text{GPPI}$	44 ms	4 ms	14

9.4 Large Number of Non-trivial Distances

For the cases when the number of non-trivial distances is large, we can either compute all distances smaller than a chosen $\varepsilon > 0$ or we can approximate the distances up to some chosen accuracy $\alpha > 0$.

Let us use randomized quicksort introduced introduced in Section 9.3.1 and the randomized self-stabilising algorithm due to Herman [50] introduced in Section 9.2.3 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 algorithm for small distances (D_0+D_1+Q+PPI) for the randomized

quicksort example with 82 states. The upper-bound ε is set to be 0.1. The algorithm terminates in about 5 minutes showing that no state pairs have distances smaller than 0.1.

We apply the approximation algorithm to the randomized self-stabilising algorithm with $N = 7$ 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 the randomized quicksort runs for about 14 minutes which makes it about 60 times faster than the original policy iteration algorithm ($D_0 + \text{SPI}$). For the randomized self-stabilising algorithm with 128 states, the approximation algorithm terminates in about 54 hours. 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	$\lvert S\rvert$	non-trivial	$D_0 + D_1 + \text{DI}$	STD
randomized quicksort	82	2,300	14 min	2 s
randomized self-stabilising algorithm	128	11,032	54 hrs	3 min

10 Simple Stochastic Games and Probabilistic Automata

We have discussed different algorithms to compute the probabilistic bisimilar distances for labelled Markov chains. Now we shift our focus to probabilistic automata. In Chapter 5, we presented a transformation from a labelled Markov chain to a stopping MDP. In this chapter, we will present a transformation from a probabilistic automaton to a simple stochastic game. This transformation was first proposed by Van Breugel and Worrell [21]. We will also present a new characterization of the probabilistic bisimilarity distances, in terms of the simple stochastic game, which might form a basis for a policy iteration algorithm.

10.1 Simple Stochastic Games

Stochastic games were introduced by Shapley [82]. We are interested in a simplified version of these games, called simple stochastic games, which were first studied by Condon [27]. We use the more general definition of Zwick and Paterson [94]. The

more general SSG can be converted to an SSG as defined in [27] in polynomial time [94, page 355].

A simple stochastic game is played with a single token by two players, called min and max, on a finite directed graph (V, E). The graph has five types of vertices: min, max and random vertices, 0-sinks and 1-sinks. The min, max and random vertices have several outgoing edges, whereas the 0-sinks and 1-sinks have no outgoing edges. Whenever the token is in a min (max) vertex, the token is moved to one of the successors of the vertex, chosen by the min (max) player. If the token is in a random vertex, the successor is chosen randomly. The min (max) player's objective is to minimize (maximize) the probability of reaching a 1-sink.

A *policy* I for the min player maps each min vertex to one of its two successors. Similarly, a *policy* A for the max player assigns to each max vertex one of its successors. Similar to the policies introduced for MDPs in Chapter 5, these policies are pure and stationary.

Now let us formally define simple stochastic games.

Definition 10.1.1. A *simple stochastic game (SSG)* is a tuple $\langle V, E, P \rangle$ consisting of

- a finite directed graph (V, E) such that
  - V is partitioned into the sets
    - $V_{\min}$ of *min vertices*,

* $V_{\max}$ of *max vertices*,

* V_{rnd} of *random vertices*,

* V_0 of *0-sinks*, and

* V_1 of *1-sinks*,

– the vertices in V_0 and V_1 have outdegree zero and all other vertices have outdegree at least one,

- a function $P : V_{\mathrm{rnd}} \to \mathrm{Distr}(V)$ such that for each vertex $v \in V_{\mathrm{rnd}}$, $P(v)(w) > 0$ iff $(v, w) \in E$.

For the rest of the section, we fix an SSG $\langle V, E, P\rangle$. The set of min and max polices are defined as follows.

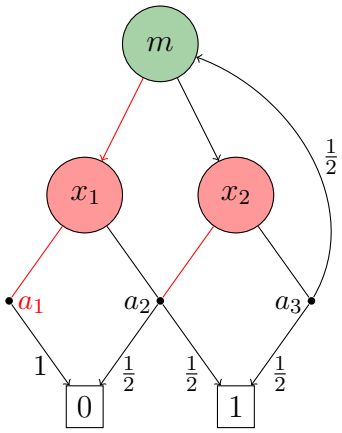
Definition 10.1.2. The set $\mathcal{I}$ of *min policies* is defined by

$$\mathcal{I} = \{I \in V_{\min} \to V \mid \forall x \in V_{\min} : (s, I(x)) \in E\}\,.$$

Definition 10.1.3. The set $\mathcal{A}$ of *max policies* is defined by

$$\mathcal{A} = \{A \in V_{\max} \to V \mid \forall x \in V_{\max} : (x, A(x)) \in E\}\,.$$

Example 10.1.4. *We consider an SSG on the finite directed graph shown below. Max vertices are red and min vertices are green. The random vertices a_1, a_2 and a_3 are denoted by bullets. The 0- and 1-sinks are square boxes which are labelled with zeroes and ones. The probabilities of the outgoing edges of the random vertices are denoted in the graph.*



A min policy maps the min vertex m to either x_1 or x_2. A max policy maps the max vertex x_1 to either a_1 or a_2 and maps the max vertex x_2 to either a_2 or a_3.

Let $I \in \mathcal{I}$ and $I(m) = x_1$. Let $A \in \mathcal{A}$ and $A(x_1) = a_1$ and $A(x_2) = a_2$. The pair of policies (I, A) is highlighted in red in the graph. □

Given a pair of min and max policies (I, A), we define a value function $v^{A,I}$ that maps each vertex x to the probability that the max player wins the game, provided that the game starts at vertex x and the min and max players play according to I and A. Condon [27, Lemma 1] shows that such a value function can be defined as the least fixed point of the following function.

Definition 10.1.5. Let $A \in \mathcal{A}$ and $I \in \mathcal{I}$. The function $\Gamma^{A,I} : [0,1]^V \to [0,1]^V$ is

defined by

$$\Gamma^{A,I}(v)(x) = \begin{cases} 0 & \text{if } x \in V_0 \\ 1 & \text{if } x \in V_1 \\ v(y) & \text{if } (x \in V_{\min} \text{ and } I(x) = y) \text{ or } \\ & \quad (x \in V_{\max} \text{ and } A(x) = y) \\ \displaystyle\sum_{(x,y)\in E} P(x)(y)\, v(y) & \text{if } x \in V_{\text{rnd}} \end{cases}$$

Example 10.1.6. *We consider the SSG and the pair of policies (A, I) of Example 10.1.4.*

At the min vertex m, the token moves to x_1 according to I. At the max vertex x_1, the token goes to a_1 according to A. There is only one outgoing edge of the random vertex a_1 which goes to a 0-sink. Thus, if we start at m, x_1 or a_1, the token reaches a 0-sink which means that the probability the max player wins the game under (I, A) is zero. If we start at the max vertex x_2, the token moves to a_2 according to A. At the random vertex a_2, the token moves to a 0-sink and 1-sink with probability $\frac{1}{2}$. Thus, the probability that the max player wins the game is $\frac{1}{2}$ if we start at x_2 or a_2. At the random vertex a_3, the token moves to m or a 1-sink, each with probability $\frac{1}{2}$. From the above analysis, we know that m will always reach a 0-sink. Thus, if we start at a_3, the probability that the max player wins the game is $\frac{1}{2}$. The following table shows the probabilities that the max player wins the game under the pair of policies (I, A) starting at each vertex.

m	x_1	x_2	a_1	a_2	a_3
0	0	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$

Alternatively, we can obtain the same values by computing the least fixed point of the function in Definition 10.1.5.

□

10.2 The Bisimulation Game for Probabilistic Automata

In this section, we introduce the transformation that maps each probabilistic automaton to an SSG. Note that this transformation was first presented by Van Breugel and Worrell [21].

Definition 10.2.1. Let $\langle S, L, \to, \ell \rangle$ be a probabilistic automaton. The SSG $\langle V, E, P \rangle$ consists of

- the set V of vertices, which is partitioned into the sets
  - $V_{\max} = \{(s,t) \in S^2 \mid s \not\sim t \wedge \ell(s) = \ell(t)\}$,
  - $V_{\min} = \{(s,\nu) \in S \times \mathrm{Distr}(S) \mid \exists t \in S : t \to \nu \wedge (s,t) \in V_{\max}\}$
  - $V_{\mathrm{rnd}} = \bigcup \{\, V(\Omega(\mu,\nu)) \mid \exists s \in S : (s,\nu) \in V_{\min} \wedge s \to \mu \,\}$,
  - $V_0 = S_0^2$,
  - $V_1 = S_1^2$,

- the set E of edges which is defined by

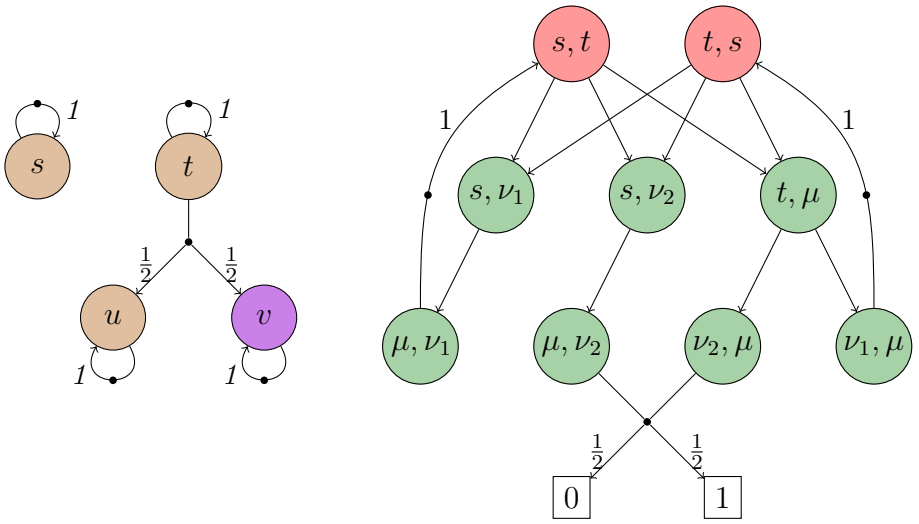
$$\begin{aligned} E \;=\; & \{\, \langle (s,t),(s,\nu)\rangle \mid (s,t) \in V_{\max} \wedge t \to \nu \} \cup \\ & \{\, \langle (s,t),(t,\mu)\rangle \mid (s,t) \in V_{\max} \wedge s \to \mu \} \cup \\ & \{ \langle (s,\nu),\omega\rangle \mid (s,\nu) \in V_{\min} \wedge s \to \mu \wedge \omega \in V(\Omega(\mu,\nu)) \,\} \cup \\ & \{\, \langle \omega,(u,v)\rangle \mid \omega \in V_{\mathrm{rnd}} \wedge (u,v) \in S^2 \wedge \omega(u,v) > 0 \,\}, \end{aligned}$$

  and

- the function $P : V_{\mathrm{rnd}} \to \mathrm{Distr}(V)$ is defined by $P(\omega)(u,v) = \omega(u,v)$.

Example 10.2.2. *We consider a probabilistic automaton with four states shown on the left. The labels are represented by the colours of the states. The only state that has multiple transitions is t: one goes back to t with probability one, while the other one takes the automaton to u and v with equal probabilities.*

Let μ, ν_1, ν_2 be defined as follows. $\mu = \mathrm{Dir}_s$, $\nu_1 = \mathrm{Dir}_t$ and $\nu_2 = \frac{1}{2}\mathrm{Dir}_u + \frac{1}{2}\mathrm{Dir}_v$. The transition relation $\to$ is defined as $\{s \to \mu, t \to \nu_1, t \to \nu_2\}$.



The SSG on the right is transformed from the probabilistic automaton on the left. □

From Definition 10.1.2 and Definition 10.1.3 we can obtain the set of min and max policies for the transformed SSG as follows.

Definition 10.2.3. The set $\mathcal{I}$ of *min policies* is defined by

$$\mathcal{I} = \left\{ I \in (S \times \mathrm{Distr}(S)) \to \mathrm{Distr}(S^2) \;\middle|\; \begin{array}{l} \forall (s,\nu) \in S \times \mathrm{Distr}(S) : \exists \mu \in \mathrm{Distr}(S) : \\ I(s,\nu) \in V(\Omega(\mu,\nu)) \wedge s \to \mu \end{array} \right\}.$$

The set $\mathcal{A}$ of *max policies* is defined by

$$\mathcal{A} = \left\{ A \in S^2_? \to (S \times \mathrm{Distr}(S)) \;\middle|\; \begin{array}{l} \forall (s,t) \in S^2_? : \\ (\exists \nu \in \mathrm{Distr}(S) : A(s,t) = (s,\nu) \wedge t \to \nu) \vee \\ (\exists \mu \in \mathrm{Distr}(S) : A(s,t) = (t,\mu) \wedge s \to \mu) \end{array} \right\}.$$

10.3 An Alternative Characterization of Probabilistic Bisimilarity Distances

In the previous section, we have shown that a probabilistic automaton can be transformed into an SSG. In this section, we will show that there exists a pair of optimal policies for the transformed SSG. We will explain what we mean by optimal policies later in this section. Furthermore, we will show that the values under the optimal policies of the SSG correspond to the probabilistic bisimilarity distances of the probabilistic automaton.

We slightly modify the function Δ of Definition 2.2.6 and introduce a discount factor c. Note that we already introduced the special case when the discount factor is 1 in Section 2.2, that is, Δ_1. This extra parameter c is needed for the proofs in this section.

Definition 10.3.1. Let $c \in (0,1]$. The function $\Delta_c : [0,1]^{S^2} \to [0,1]^{S^2}$ is defined as follows.

$$\Delta_c(d)(s,t) = \begin{cases} 1 & \text{if } \ell(s) \neq \ell(t) \\ 0 & \text{if } s \sim t \\ c\,\Delta(d)(s,t) & \text{otherwise.} \end{cases}$$

We collect some properties of Δ_c in the following theorem.

Proposition 10.3.2. *For all* $c \in (0,1]$,

(a) the function Δ_c is monotone, and

(b) the function Δ_c is c-Lipschitz.

Proof. First, we prove part (a). Let $c \in (0, 1]$. Let d, $e \in [0, 1]^{S^2}$ with $d \sqsubseteq e$. Let $s, t \in S$. We distinguish three cases.

- If $\ell(s) \neq \ell(t)$ then

$$\Delta_c(d)(s, t) = 1 = \Delta_c(e)(s, t).$$

- If $s \sim t$ then

$$\Delta_c(d)(s, t) = 0 = \Delta_c(e)(s, t).$$

- Otherwise,

$$\begin{aligned} \Delta_c(d)(s, t) &= c\,\Delta(d)(s, t) \\ &\leq c\,\Delta(e)(s, t) \\ &\qquad [d \sqsubseteq e \text{ and } \Delta(d)(s, t) \leq \Delta(e)(s, t) \text{ by Proposition 2.2.7}] \\ &= \Delta_c(e)(s, t). \end{aligned}$$

Next, we prove part (b). Let $c \in (0, 1]$. Let d, $e \in [0, 1]^{S^2}$. It suffices to show that for all s, $t \in S$, $|\Delta_c(d)(s, t) - \Delta_c(e)(s, t)| \leq c\,\|d - e\|$. Let s, $t \in S$. We distinguish three cases.

- If $\ell(s) \neq \ell(t)$ then

$$|\Delta_c(d)(s, t) - \Delta_c(e)(s, t)| = |1 - 1| = 0 \leq c\,\|d - e\|.$$

- If $s \sim t$ then

$$|\Delta_c(d)(s,t) - \Delta_c(e)(s,t)| = |0 - 0| = 0 \leq c\,\|d - e\|.$$

- Let $\ell(s) = \ell(t)$ and $s \not\sim t$. In this case, $\Delta_c(d)$ is the composition of the Hausdorff distance H (Definition 2.2.10) and the Kantorovich distance K (Definition 2.1.24). Hence,

$$\begin{aligned}
&|\Delta_c(d)(s,t) - \Delta_c(e)(s,t)| \\
&= |c\,H(K(d))(\{\mu \mid s \to \mu\}, \{\nu \mid t \to \nu\}) - \\
&\qquad\qquad c\,H(K(e))(\{\mu \mid s \to \mu\}, \{\nu \mid t \to \nu\})| \\
&= c\,|H(K(d))(\{\mu \mid s \to \mu\}, \{\nu \mid t \to \nu\}) - \\
&\qquad\qquad H(K(e))(\{\mu \mid s \to \mu\}, \{\nu \mid t \to \nu\})| \\
&\leq c\,\|H(K(d)) - H(K(e))\| \\
&\leq c\,\|K(d) - K(e)\| \qquad \text{[Proposition 2.2.11]} \\
&\leq c\,\|d - e\| \qquad \text{[Proposition 2.1.25]}
\end{aligned}$$

□

By Definition 10.1.5, we can define the values of the SSG under the pair of policies (A, I) as the least fixed point of $\Gamma_1^{A,I}$. This least fixed point captures the expectation of reaching a state pair with different labels if both players use the

given policies. Note that here we introduce a discount factor c to the function, as we will prove things about $c = 1$ by taking limits as $c \to 1$ later.

Definition 10.3.3. Let $A \in \mathcal{A}$, $I \in \mathcal{I}$ and $c \in (0, 1]$. The function $\Gamma_c^{A,I} : [0,1]^{S^2} \to [0,1]^{S^2}$ is defined as follows.

$$\Gamma_c^{A,I}(d)(s,t) = \begin{cases} 1 & \text{if } (s,t) \in S_1^2 \\ 0 & \text{if } (s,t) \in S_0^2 \\ c \sum_{u,v \in S} I(A(s,t))(u,v)\, d(u,v) & \text{otherwise.} \end{cases}$$

We collect some properties of $\Gamma_c^{A,I}$ in the following theorem.

Proposition 10.3.4. *For all $A \in \mathcal{A}$, $I \in \mathcal{I}$ and $c \in (0, 1]$,*

(a) the function $\Gamma_c^{A,I}$ is monotone and

(b) the function $\Gamma_c^{A,I}$ is c-Lipschitz.

Proof.

(a) Let $A \in \mathcal{A}$, $I \in \mathcal{I}$ and $c \in (0, 1]$. Let d, $e \in [0,1]^{S^2}$ with $d \sqsubseteq e$. We distinguish three cases.

- If $\ell(s) \neq \ell(t)$ then

$$\Gamma_c^{A,I}(d)(s,t) = 1 = \Gamma_c^{A,I}(e)(s,t).$$

- If $s \sim t$ then

$$\Gamma_c^{A,I}(d)(s,t) = 0 = \Gamma_c^{A,I}(e)(s,t).$$

– Otherwise,

$$\begin{aligned}\Gamma_c^{A,I}(d)(s,t) &= c \sum_{u,v\in S} I(A(s,t))(u,v)\ d(u,v) \\ &\leq c \sum_{u,v\in S} I(A(s,t))(u,v)\ e(u,v) \qquad [d \sqsubseteq e] \\ &= \Gamma_c^{A,I}(e)(s,t).\end{aligned}$$

(b) Let $A \in \mathcal{A}$, $I \in \mathcal{I}$ and $c \in (0,1]$. Let $d,\ e \in [0,1]^{S^2}$. Let $s,\ t \in S$. We distinguish three cases.

– If $\ell(s) \neq \ell(t)$ then

$$|\Gamma_c^{A,I}(d)(s,t) - \Gamma_c^{A,I}(e)(s,t)| = |1-1| = 0 \leq c\,\|d-e\|.$$

– If $s \sim t$ then

$$|\Gamma_c^{A,I}(d)(s,t) - \Gamma_c^{A,I}(e)(s,t)| = |0-0| = 0 \leq c\,\|d-e\|.$$

– Otherwise,

$$\begin{aligned}&|\Gamma_c^{A,I}(d)(s,t) - \Gamma_c^{A,I}(e)(s,t)| \\ &= \left| c \sum_{u,v\in S} I(A(s,t))(u,v)\,d(u,v) - c \sum_{u,v\in S} I(A(s,t))(u,v)\,e(u,v) \right| \\ &= c \left| \sum_{u,v\in S} I(A(s,t))(u,v)\,d(u,v) - \sum_{u,v\in S} I(A(s,t))(u,v)\,e(u,v) \right| \\ &= c \sum_{u,v\in S} I(A(s,t))(u,v)\,|d(u,v) - e(u,v)| \\ &\leq c \sum_{u,v\in S} I(A(s,t))(u,v)\,\|d-e\| \\ &= c\,\|d-e\|.\end{aligned}$$

□

By the Knaster-Tarski fixed point theorem (Theorem 2.1.8(a)), $\Gamma_c^{A,I}$ has a least fixed point, which we denote by $\boldsymbol{\mu}(\Gamma_c^{A,I})$. In the remainder of this section, we show that there exists a max policy A^* and a min policy I^* such that the corresponding value function captures the probabilistic bisimilarity distances, that is $\boldsymbol{\mu}(\Delta_1) = \boldsymbol{\mu}(\Gamma_1^{A^*,I^*})$. We call these policies A^* and I^* optimal.

The proof consists of two parts. First, we prove that there exists an optimal min policy $I^* \in \mathcal{I}$ such that

$$\forall A \in \mathcal{A} : \boldsymbol{\mu}(\Gamma_1^{A,I^*}) \sqsubseteq \boldsymbol{\mu}(\Delta_1). \tag{10.1}$$

Lemma 10.3.5. *There exists $I \in \mathcal{I}$ such that for all $A \in \mathcal{A}$, $\boldsymbol{\mu}(\Gamma_1^{A,I}) \sqsubseteq \boldsymbol{\mu}(\Delta_1)$.*

Proof. Towards the construction of $I^* \in \mathcal{I}$, let $s \in S$ and $\nu \in \mathrm{Distr}(S)$. Since we restrict our attention to finitely branching probabilistic automata,

$$\mu_{s,\nu} = \operatorname*{argmin}_{s \to \mu} \min_{\omega \in V(\Omega(\mu,\nu))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta_1(u,v)) \tag{10.2}$$

exists. Because the set $V(\Omega(\mu_{s,\nu}, \nu))$ is nonempty and finite, we can define

$$I^*(s,\nu) = \operatorname*{argmin}_{\omega \in V(\Omega(\mu_{s,\nu},\nu))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta_1(u,v)). \tag{10.3}$$

By construction $I^* \in \mathcal{I}$.

Let $A \in \mathcal{A}$. Since $\boldsymbol{\mu}(\Gamma_1^{A,I^*})$ is the least pre-fixed point of Γ_1^{A,I^*} according to Theorem 2.1.8(c), to conclude that $\boldsymbol{\mu}(\Gamma_1^{A,I^*}) \sqsubseteq \boldsymbol{\mu}(\Delta_1)$ it suffices to show that $\boldsymbol{\mu}(\Delta_1)$

is a pre-fixed point of Γ_1^{A,I^*}, that is, $\Gamma_1^{A,I^*}(\boldsymbol{\mu}(\Delta_1)) \sqsubseteq \boldsymbol{\mu}(\Delta_1)$. Let $s, t \in S$. We distinguish three cases.

- If $\ell(s) \neq \ell(t)$ then $\Gamma_1^{A,I^*}(\boldsymbol{\mu}(\Delta_1))(s,t) = 1 = \Delta_1(\boldsymbol{\mu}(\Delta_1))(s,t) = \boldsymbol{\mu}(\Delta_1)(s,t)$.

- If $s \sim t$ then $\Gamma_1^{A,I^*}(\boldsymbol{\mu}(\Delta_1))(s,t) = 0 = \Delta_1(\boldsymbol{\mu}(\Delta_1))(s,t) = \boldsymbol{\mu}(\Delta_1)(s,t)$.

- Otherwise, $\ell(s) = \ell(t)$ and $s \not\sim t$. Without any loss of generality, we assume that $A(s,t) = (s,\nu)$ with $t \to \nu$. The case that $A(s,t) = (t,\mu)$ with $s \to \mu$ can be dealt with similarly. Then

$$
\begin{aligned}
&\Gamma_1^{A,I^*}(\boldsymbol{\mu}(\Delta_1))(s,t) \\
&= \sum_{u,v \in S} I^*(A(s,t))(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \\
&= \sum_{u,v \in S} I^*(s,\nu)(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \qquad [A(s,t) = (s,\nu)] \\
&= \min_{\omega \in V(\Omega(\mu_{s,\nu},\nu))} \sum_{u,v \in S} \omega(u,v)\, (\boldsymbol{\mu}\Delta_1)(u,v) \qquad [(10.3)] \\
&= \min_{s \to \mu} \min_{\omega \in V(\Omega(\mu,\nu))} \sum_{u,v \in S} \omega(u,v)\, (\boldsymbol{\mu}\Delta_1)(u,v) \qquad [(10.2)] \\
&\leq \max_{t \to \nu} \min_{s \to \mu} \min_{\omega \in V(\Omega(\mu,\nu))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \\
&\leq \max \left\{ \max_{s \to \mu} \min_{t \to \nu} \min_{\omega \in V(\Omega(\mu,\nu))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v), \right. \\
&\qquad \left. \max_{t \to \nu} \min_{s \to \mu} \min_{\omega \in V(\Omega(\mu,\nu))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \right\} \\
&= \Delta_1(\boldsymbol{\mu}(\Delta_1))(s,t) \\
&= \boldsymbol{\mu}(\Delta_1)(s,t).
\end{aligned}
$$

□

In the remainder of this dissertation, we denote the optimal min policy constructed in the above proof by I^*.

As we will see in the proof of Theorem 10.3.20, it remains to prove that there exists an optimal max policy $A^* \in \mathcal{A}$ such that

$$\forall I \in \mathcal{I} : \boldsymbol{\mu}(\Delta_1) \sqsubseteq \boldsymbol{\mu}(\Gamma_1^{A^*,I}).$$

The proof of this second part turns out to be much more involved than the proof of the first part contained in the above proposition. The proof has the following three major components.

- The probabilistic bisimilarity distances captured by $\boldsymbol{\mu}(\Delta_1)$ are the limit of their discounted counterparts represented by $\boldsymbol{\mu}(\Delta_c)$.
- Similarly, for all $A \in \mathcal{A}$ and $I \in \mathcal{I}$, the value function $\boldsymbol{\mu}(\Gamma_1^{A,I})$ is the limit of the discounted value functions $\boldsymbol{\mu}(\Gamma_c^{A,I})$. This result is inspired by [38, Theorem 4.4.1].
- There exists an optimal max policy in the discounted setting.

Combining the above three components, we arrive at an optimal max policy. The first two components are proved in Proposition 10.3.11 and Proposition 10.3.17. The third major component of the proof, Proposition 10.3.18, consists of showing that there exists an optimal max policy in the discounted setting.

In the next few propositions, we show the first component, $\lim_{c\uparrow 1} \boldsymbol{\mu}(\Delta_c) = \boldsymbol{\mu}(\Delta_1)$.

Proposition 10.3.6. *For all b, $c \in (0,1]$, if $b \leq c$ then $\boldsymbol{\mu}(\Delta_b) \sqsubseteq \boldsymbol{\mu}(\Delta_c)$.*

Proof. Let b, $c \in (0,1]$ with $b \leq c$. To conclude that $\boldsymbol{\mu}(\Delta_b) \sqsubseteq \boldsymbol{\mu}(\Delta_c)$ it suffices to show $\Delta_b(\boldsymbol{\mu}(\Delta_c)) \sqsubseteq \boldsymbol{\mu}(\Delta_c)$ according to Theorem 2.1.8(c). Let s, $t \in S$. We distinguish three cases.

- If $\ell(s) \neq \ell(t)$ then

$$\Delta_b(\boldsymbol{\mu}(\Delta_c))(s,t) = 1 = \Delta_c(\boldsymbol{\mu}(\Delta_c))(s,t) = \boldsymbol{\mu}(\Delta_c)(s,t).$$

- If $s \sim t$ then

$$\Delta_b(\boldsymbol{\mu}(\Delta_c))(s,t) = 0 = \Delta_c(\boldsymbol{\mu}(\Delta_c))(s,t) = \boldsymbol{\mu}(\Delta_c)(s,t).$$

- Otherwise,

$$\begin{aligned}&\Delta_b(\boldsymbol{\mu}(\Delta_c))(s,t) = b\,\Delta_1(\boldsymbol{\mu}(\Delta_c))(s,t) \leq c\,\Delta_1(\boldsymbol{\mu}(\Delta_c))(s,t)\\ &= \Delta_c(\boldsymbol{\mu}(\Delta_c))(s,t) = \boldsymbol{\mu}(\Delta_c)(s,t).\end{aligned}$$

□

We define a new function δ_1 which helps to prove that the probabilistic bisimilarity distances captured by $\boldsymbol{\mu}(\Delta_1)$ are the limit of their discounted counterparts represented by $\boldsymbol{\mu}(\Delta_c)$, that is, $\lim_{c\uparrow 1} \boldsymbol{\mu}(\Delta_c) = \boldsymbol{\mu}(\Delta_1)$.

Definition 10.3.7. The function $\delta_1 : S^2 \to [0,1]$ is defined by

$$\delta_1(s,t) = \sup_{c \in (0,1)} \boldsymbol{\mu}(\Delta_c)(s,t).$$

Proposition 10.3.8. $\lim_{c \uparrow 1} \boldsymbol{\mu}(\Delta_c) = \delta_1$.

Proof. It suffices to prove that

$$\forall \epsilon > 0 : \exists b \in (0,1) : \forall c \in [b,1) : \|\boldsymbol{\mu}(\Delta_c) - \delta_1\| < \epsilon.$$

From Proposition 10.3.6 and the definition of δ_1 we can conclude that for each $(s,t) \in S^2$ there exists $b_{(s,t)} \in (0,1)$ such that $|\boldsymbol{\mu}(\Delta_c)(s,t) - \delta_1(s,t)| < \epsilon$ for all $b_{(s,t)} \leq c < 1$. Let $b = \max_{(s,t) \in S^2} b_{(s,t)}$. Then $\|\boldsymbol{\mu}(\Delta_c) - \delta_1\| < \epsilon$ for all $b \leq c < 1$. $\square$

The next two propositions show that $\delta_1 = \boldsymbol{\mu}(\Delta_1)$.

Proposition 10.3.9. $\delta_1 \sqsubseteq \boldsymbol{\mu}(\Delta_1)$.

Proof. Next, we show that for all $c \in (0,1)$, $\boldsymbol{\mu}(\Delta_c) \sqsubseteq \boldsymbol{\mu}(\Delta_1)$. From this we can immediately deduce that $\delta_1 \sqsubseteq \boldsymbol{\mu}(\Delta_1)$.

Let $c \in (0,1)$. To conclude that $\boldsymbol{\mu}(\Delta_c) \sqsubseteq \boldsymbol{\mu}(\Delta_1)$, it suffices to show that $\Delta_c(\boldsymbol{\mu}(\Delta_1)) \sqsubseteq \boldsymbol{\mu}(\Delta_1)$ according to Theorem 2.1.8(d). Let s, $t \in S$. We distinguish three cases.

- If $\ell(s) \neq \ell(t)$ then

$$\Delta_c(\boldsymbol{\mu}(\Delta_1))(s,t) = 1 = \Delta_1(\boldsymbol{\mu}(\Delta_1))(s,t) = \boldsymbol{\mu}(\Delta_1)(s,t).$$

- If $s \sim t$ then

$$\Delta_c(\boldsymbol{\mu}(\Delta_1))(s,t) = 0 = \Delta_1(\boldsymbol{\mu}(\Delta_1))(s,t) = \boldsymbol{\mu}(\Delta_1)(s,t).$$

- Otherwise,

$$\Delta_c(\boldsymbol{\mu}(\Delta_1))(s,t) = c\,\Delta_1(\boldsymbol{\mu}(\Delta_1))(s,t) \leq \Delta_1(\boldsymbol{\mu}(\Delta_1))(s,t) = \boldsymbol{\mu}(\Delta_1)(s,t).$$

□

Proposition 10.3.10. $\boldsymbol{\mu}(\Delta_1) \sqsubseteq \delta_1$.

Proof. To conclude that $\boldsymbol{\mu}(\Delta_1) \sqsubseteq \delta_1$, it suffices to show that $\Delta_1(\delta_1) = \delta_1$ according to Theorem 2.1.8(d). Let s, $t \in S$. We distinguish three cases.

- If $\ell(s) \neq \ell(t)$ then

$$\Delta_1(\delta_1)(s,t) = 1 = \lim_{c\uparrow 1} \Delta_c(\boldsymbol{\mu}(\Delta_c))(s,t) = \lim_{c\uparrow 1} \boldsymbol{\mu}(\Delta_c)(s,t) = \delta_1(s,t)$$

  by Proposition 10.3.8.

- If $s \sim t$ then

$$\Delta_1(\delta_1)(s,t) = 0 = \lim_{c\uparrow 1} \Delta_c(\boldsymbol{\mu}(\Delta_c))(s,t) = \lim_{c\uparrow 1} \boldsymbol{\mu}(\Delta_c)(s,t) = \delta_1(s,t)$$

  by Proposition 10.3.8.

- Otherwise

$$\Delta_1(\delta_1)(s,t) \;=\; \Delta_1(\lim_{c\uparrow 1} \boldsymbol{\mu}(\Delta_c))(s,t) \qquad \text{[Proposition 10.3.8]}$$

$$
\begin{aligned}
&= \lim_{c\uparrow 1} \Delta_1(\boldsymbol{\mu}(\Delta_c))(s,t) \\
&\qquad [\Delta_1 \text{ is 1-Lipschitz and, hence, continuous}] \\
&\qquad [\text{by Proposition 2.2.12}] \\
&= \lim_{c\uparrow 1} c \lim_{c\uparrow 1} \Delta_1(\boldsymbol{\mu}(\Delta_c))(s,t) \\
&= \lim_{c\uparrow 1} c\, \Delta_1(\boldsymbol{\mu}(\Delta_c))(s,t) \qquad [\text{multiplication is continuous}] \\
&= \lim_{c\uparrow 1} \Delta_c(\boldsymbol{\mu}(\Delta_c))(s,t) \\
&= \lim_{c\uparrow 1} \boldsymbol{\mu}(\Delta_c)(s,t) \\
&= \delta_1(s,t) \qquad [\text{Proposition 10.3.8}]
\end{aligned}
$$

□

Proposition 10.3.11. $\lim_{c\uparrow 1} \boldsymbol{\mu}(\Delta_c) = \boldsymbol{\mu}(\Delta_1)$.

Proof.

$$
\begin{aligned}
\lim_{c\uparrow 1} \boldsymbol{\mu}(\Delta_c) &= \delta_1 \qquad [\text{Proposition 10.3.8}] \\
&= \boldsymbol{\mu}(\Delta_1) \qquad [\text{Proposition 10.3.9 and 10.3.10}]
\end{aligned}
$$

□

We have completed the first component of the proof. In the next few propositions, we show that for all $A \in \mathcal{A}$ and $I \in \mathcal{I}$, the value function $\boldsymbol{\mu}(\Gamma_1^{A,I})$ is the limit of the discounted value functions $\boldsymbol{\mu}(\Gamma_c^{A,I})$, that is, $\lim_{c\uparrow 1} \boldsymbol{\mu}(\Gamma_c^{A,I}) = \boldsymbol{\mu}(\Gamma_1^{A,I})$. This result is inspired by [38, Theorem 4.4.1].

Proposition 10.3.12. *For all $A \in \mathcal{A}$, $I \in \mathcal{I}$ and b, $c \in (0,1]$, if $b \leq c$ then $\boldsymbol{\mu}(\Gamma_b^{A,I}) \sqsubseteq \boldsymbol{\mu}(\Gamma_c^{A,I})$.*

Proof. Let $A \in \mathcal{A}$ and $I \in \mathcal{I}$. Let b, $c \in (0,1]$ with $b \leq c$. To conclude that $\boldsymbol{\mu}(\Gamma_b^{A,I}) \sqsubseteq \boldsymbol{\mu}(\Gamma_c^{A,I})$ it suffices to show $\Gamma_b^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I})) \sqsubseteq \boldsymbol{\mu}(\Gamma_c^{A,I})$ according to Theorem 2.1.8(d). Let s, $t \in S$. We distinguish three cases.

- If $\ell(s) \neq \ell(t)$ then

$$\Gamma_b^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) = 1 = \Gamma_c^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) = \boldsymbol{\mu}(\Gamma_c^{A,I})(s,t).$$

- If $s \sim t$ then

$$\Gamma_b^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) = 0 = \Gamma_c^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) = \boldsymbol{\mu}(\Gamma_c^{A,I})(s,t).$$

- Otherwise,

$$\begin{aligned} \Gamma_b^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) &= b\,\Gamma_1^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) \\ &\leq c\,\Gamma_1^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) \\ &= \Gamma_c^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) \\ &= \boldsymbol{\mu}(\Gamma_c^{A,I})(s,t). \end{aligned}$$

□

To prove $\lim_{c \uparrow 1} \boldsymbol{\mu}(\Delta_c) = \boldsymbol{\mu}(\Delta_1)$, we have defined a function δ_1 . Similarly, to prove $\lim_{c \uparrow 1} \boldsymbol{\mu}(\Gamma_c^{A,I}) = \boldsymbol{\mu}(\Gamma_1^{A,I})$, we define a function $\gamma_1^{A,I}$.

Definition 10.3.13. Let $A \in \mathcal{A}$ and $I \in \mathcal{I}$. The function $\gamma_1^{A,I} : S^2 \to [0,1]$ is defined by

$$\gamma_1^{A,I}(s,t) = \sup_{c \in (0,1)} \boldsymbol{\mu}(\Gamma_c^{A,I})(s,t).$$

The next proposition shows that $\gamma_1^{A,I} = \lim_{c \uparrow 1} \boldsymbol{\mu}(\Gamma_c^{A,I})$.

Proposition 10.3.14. *For all $A \in \mathcal{A}$ and $I \in \mathcal{I}$, $\lim_{c \uparrow 1} \boldsymbol{\mu}(\Gamma_c^{A,I}) = \gamma_1^{A,I}$.*

Proof. Let $A \in \mathcal{A}$ and $I \in \mathcal{I}$. It suffices to prove that

$$\forall \epsilon > 0 : \exists b \in (0,1) : \forall c \in [b,1) : \|\boldsymbol{\mu}(\Gamma_c^{A,I}) - \gamma_1^{A,I}\| < \epsilon.$$

From Proposition 10.3.12 and the definition of $\gamma_1^{A,I}$ we can conclude that for each $(s,t) \in S^2$ there exists $b_{(s,t)} \in (0,1)$ such that $|\boldsymbol{\mu}\Gamma_c^{A,I}(s,t) - \gamma_1^{A,I}(s,t)| < \epsilon$ for all $b_{(s,t)} \leq c < 1$. Let $b = \max_{(s,t) \in S^2} b_{(s,t)}$. Then $\|\boldsymbol{\mu}(\Gamma_c^{A,I}) - \gamma_1^{A,I}\| < \epsilon$ for all $b \leq c < 1$. $\square$

The next two propositions indicate that $\gamma_1^{A,I} = \boldsymbol{\mu}(\Gamma_1^{A,I})$.

Proposition 10.3.15. *For all $A \in \mathcal{A}$ and $I \in \mathcal{I}$, $\gamma_1^{A,I} \sqsubseteq \boldsymbol{\mu}(\Gamma_1^{A,I})$.*

Proof. Let $A \in \mathcal{A}$ and $I \in \mathcal{I}$. We show that for all $c \in (0,1)$, $\boldsymbol{\mu}(\Gamma_c^{A,I}) \sqsubseteq \boldsymbol{\mu}(\Gamma_1^{A,I})$. From this we can immediately deduce that $\gamma_1^{A,I} \sqsubseteq \boldsymbol{\mu}(\Gamma_1^{A,I})$.

Let $c \in (0,1)$. To conclude that $\boldsymbol{\mu}(\Gamma_c^{A,I}) \sqsubseteq \boldsymbol{\mu}(\Gamma_1^{A,I})$, it suffices to show that $\Gamma_c^{A,I}(\boldsymbol{\mu}(\Gamma_1^{A,I})) \sqsubseteq \boldsymbol{\mu}(\Gamma_1^{A,I})$ according to Theorem 2.1.8(d). Let $s, t \in S$. We distinguish three cases.

- If $\ell(s) \neq \ell(t)$ then

$$\Gamma_c^{A,I}(\boldsymbol{\mu}(\Gamma_1^{A,I}))(s,t) = 1 = \Gamma_1^{A,I}(\boldsymbol{\mu}(\Gamma_1^{A,I}))(s,t) = \boldsymbol{\mu}(\Gamma_1^{A,I})(s,t).$$

- If $s \sim t$ then

$$\Gamma_c^{A,I}(\boldsymbol{\mu}(\Gamma_1^{A,I}))(s,t) = 0 = \Gamma_1^{A,I}(\boldsymbol{\mu}(\Gamma_1^{A,I}))(s,t) = \boldsymbol{\mu}(\Gamma_1^{A,I})(s,t).$$

- Otherwise,

$$\Gamma_c^{A,I}(\boldsymbol{\mu}(\Gamma_1^{A,I}))(s,t) = c\,\Gamma_1^{A,I}(\boldsymbol{\mu}(\Gamma_1^{A,I}))(s,t) \leq \Gamma_1^{A,I}(\boldsymbol{\mu}(\Gamma_1^{A,I}))(s,t) = \boldsymbol{\mu}(\Gamma_1^{A,I})(s,t).$$

□

Proposition 10.3.16. *For all $A \in \mathcal{A}$ and $I \in \mathcal{I}$, $\boldsymbol{\mu}(\Gamma_1^{A,I}) \sqsubseteq \gamma_1^{A,I}$.*

Proof. Let $A \in \mathcal{A}$ and $I \in \mathcal{I}$. To conclude that $\boldsymbol{\mu}(\Gamma_1^{A,I}) \sqsubseteq \gamma_1^{A,I}$, it suffices to show that $\Gamma_1^{A,I}(\gamma_1^{A,I}) = \gamma_1^{A,I}$ according to Theorem 2.1.8(d). Let s, $t \in S$. We distinguish three cases.

- If $\ell(s) \neq \ell(t)$ then

$$\Gamma_1^{A,I}(\gamma_1^{A,I})(s,t) = 1 = \lim_{c\uparrow 1} \Gamma_c^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) =$$

$$\lim_{c\uparrow 1} \boldsymbol{\mu}(\Gamma_c^{A,I})(s,t) = \gamma_1^{A,I}(s,t) \qquad \text{[Proposition 10.3.14]}.$$

- If $s \sim t$ then

$$\Gamma_1^{A,I}(\gamma_1^{A,I})(s,t) = 0 = \lim_{c\uparrow 1} \Gamma_c^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) =$$

$$\lim_{c\uparrow 1} \boldsymbol{\mu}(\Gamma_c^{A,I})(s,t) = \gamma_1^{A,I}(s,t) \qquad \text{[Proposition 10.3.14]}.$$

- Otherwise,

$$
\begin{aligned}
&\Gamma_1^{A,I}(\gamma_1^{A,I})(s,t) \\
&\quad = \Gamma_1^{A,I}(\lim_{c\uparrow 1} \boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) \qquad \text{[Proposition 10.3.14]} \\
&\quad = \lim_{c\uparrow 1} \Gamma_1^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) \\
&\qquad\qquad \text{[by Proposition 10.3.4, } \Gamma_1^{A,I} \text{ is 1-Lipschitz and, hence, continuous]} \\
&\quad = \lim_{c\uparrow 1} c \lim_{c\uparrow 1} \Gamma_1^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) \\
&\quad = \lim_{c\uparrow 1} c\, \Gamma_1^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) \qquad \text{[multiplication is continuous]} \\
&\quad = \lim_{c\uparrow 1} \Gamma_c^{A,I}(\boldsymbol{\mu}(\Gamma_c^{A,I}))(s,t) \\
&\quad = \lim_{c\uparrow 1} \boldsymbol{\mu}(\Gamma_c^{A,I})(s,t) \\
&\quad = \gamma_1^{A,I}(s,t) \qquad \text{[Proposition 10.3.14]}
\end{aligned}
$$

□

Combining the above propositions, we can conclude $\lim_{c\uparrow 1} \boldsymbol{\mu}(\Gamma_c^{A,I}) = \boldsymbol{\mu}(\Gamma_1^{A,I})$, which completes the second component of the proof.

Proposition 10.3.17. *For all $A \in \mathcal{A}$ and $I \in \mathcal{I}$, $\lim_{c\uparrow 1} \boldsymbol{\mu}(\Gamma_c^{A,I}) = \boldsymbol{\mu}(\Gamma_1^{A,I})$.*

Proof. Let $A \in \mathcal{A}$ and $I \in \mathcal{I}$.

$$
\begin{aligned}
\lim_{c\uparrow 1} \boldsymbol{\mu}(\Gamma_c^{A,I}) &= \gamma_1^{A,I} \qquad \text{[Proposition 10.3.14]} \\
&= \boldsymbol{\mu}(\Gamma_1^{A,I}) \qquad \text{[Proposition 10.3.15 and 10.3.16]}
\end{aligned}
$$

□

For the case when the discount factor c is smaller than one, we construct a max policy $A_c^* \in \mathcal{A}$ in the next proposition so that $\forall I \in \mathcal{I} : \boldsymbol{\mu}(\Delta_c) \sqsubseteq \boldsymbol{\mu}(\Gamma_c^{A_c^*,I})$.

Proposition 10.3.18. *For all $c \in (0,1)$, $\exists A \in \mathcal{A} : \forall I \in \mathcal{I} : \boldsymbol{\mu}(\Delta_c) \sqsubseteq \boldsymbol{\mu}(\Gamma_c^{A,I})$.*

Proof. Let $c \in (0,1)$. Let s, $t \in S$. If

$$\max_{s \to \mu} \min_{t \to \nu} K(\boldsymbol{\mu}(\Delta_c))(\mu,\nu) \geq \max_{t \to \nu} \min_{s \to \mu} K(\boldsymbol{\mu}(\Delta_c))(\mu,\nu) \tag{10.4}$$

then we define $A_c^*(s,t)$ by

$$A_c^*(s,t) = \left(t, \operatorname*{argmax}_{s \to \mu} \min_{t \to \nu} K(\boldsymbol{\mu}(\Delta_c))(\mu,\nu)\right).$$

Because the probabilistic automaton is finitely branching, the above exists. Otherwise, we define $A_c^*(s,t)$ by

$$A_c^*(s,t) = \left(s, \operatorname*{argmax}_{t \to \nu} \min_{s \to \mu} K(\boldsymbol{\mu}(\Delta_c))(\mu,\nu)\right).$$

By construction, $A_c^* \in \mathcal{A}$.

Let $I \in \mathcal{I}$. Since $\langle [0,1]^{S \times S}, \|\cdot - \cdot\| \rangle$ is a nonempty complete metric space according to Proposition 2.1.22 and the function $\Gamma_c^{A_c^*,I}$ is contractive by Proposition 10.3.4, we can conclude from Theorem 2.1.23 that $\Gamma_c^{A_c^*,I}$ has a unique fixed point. Therefore, $\boldsymbol{\mu}(\Gamma_c^{A_c^*,I})$ is not only the least fixed point but also the greatest fixed point of $\Gamma_c^{A_c^*,I}$. According to Theorem 2.1.8(b), $\boldsymbol{\mu}(\Gamma_c^{A_c^*,I})$ is the greatest post-fixed point of $\Gamma_c^{A_c^*,I}$. Hence, to conclude that $\boldsymbol{\mu}(\Delta_c) \sqsubseteq \boldsymbol{\mu}(\Gamma_c^{A_c^*,I})$ it suffices to show that $\boldsymbol{\mu}(\Delta_c)$ is a post-fixed point of $\Gamma_c^{A_c^*,I}$, that is, $\boldsymbol{\mu}(\Delta_c) \sqsubseteq \Gamma_c^{A_c^*,I}(\boldsymbol{\mu}(\Delta_c))$. Let s, $t \in S$. We distinguish three cases.

- If $(s,t) \in S_0^2$, then

$$\begin{aligned}
\boldsymbol{\mu}(\Delta_c)(s,t) &\leq \boldsymbol{\mu}(\Delta_1)(s,t) \qquad \text{[Proposition 10.3.6]} \\
&= \boldsymbol{\mu}(\Delta)(s,t) \qquad \text{[Theorem 2.2.14]} \\
&= 0 \qquad \text{[Theorem 2.2.8]} \\
&= \Gamma_c^{A_c^*,I}(\boldsymbol{\mu}(\Delta_c))(s,t).
\end{aligned}$$

- If $(s,t) \in S_1^2$, then

$$\begin{aligned}
\boldsymbol{\mu}(\Delta_c)(s,t) &= \Delta_c(\boldsymbol{\mu}(\Delta_c))(s,t) \\
&= 1 \\
&= \Gamma_c^{A_c^*,I}(\boldsymbol{\mu}(\Delta_c))(s,t).
\end{aligned}$$

- Otherwise, $(s,t) \in S_?^2$. Without loss of any generality, assume that $A_c^*(s,t) = (t,\mu)$. This assumption implies (10.4) and

$$\Delta_1(\boldsymbol{\mu}(\Delta_c))(s,t) = \min_{t \to \nu} K(\boldsymbol{\mu}(\Delta_c))(\mu,\nu). \tag{10.5}$$

Hence,

$$\begin{aligned}
\boldsymbol{\mu}(\Delta_c)(s,t) &= \Delta_c(\boldsymbol{\mu}(\Delta_c))(s,t) \\
&= c\,\Delta_1(\boldsymbol{\mu}(\Delta_c))(s,t) \\
&= c \min_{t \to \nu} K(\boldsymbol{\mu}(\Delta_c))(\mu,\nu) \qquad \text{[(10.5)]} \\
&= c \min_{t \to \nu} \min_{\omega \in V(\Omega(\nu,\mu))} \sum_{u,v \in S} \omega(u,v)\,\boldsymbol{\mu}(\Delta_c)(u,v)
\end{aligned}$$

[Definition 2.1.24 of Kantorovich metric]

$$\leq c \sum_{u,v \in S} I(A_c^*(s,t))(u,v)\, \boldsymbol{\mu}(\Delta_c)(u,v)$$

$$= c\, \Gamma_1^{A_c^*, I}(\boldsymbol{\mu}(\Delta_c))(s,t)$$

$$= \Gamma_c^{A_c^*, I}(\boldsymbol{\mu}(\Delta_c))(s,t).$$

□

Combining the three components, we obtain the second part of the proof. The next lemma shows the existence of a policy $A^* \in \mathcal{A}$ such that

$$\forall I \in \mathcal{I} : \boldsymbol{\mu}(\Delta_1) \sqsubseteq \boldsymbol{\mu}(\Gamma_1^{A^*, I}).$$

Lemma 10.3.19. $\exists A \in \mathcal{A} : \forall I \in \mathcal{I} : \boldsymbol{\mu}(\Delta_1) \sqsubseteq \boldsymbol{\mu}(\Gamma_1^{A,I})$.

Proof. According to Proposition 10.3.18,

$$\forall n \in \mathbb{N} : \exists A_n \in \mathcal{A} : \forall I \in \mathcal{I} : \boldsymbol{\mu}(\Delta_{\frac{n}{n+1}}) \sqsubseteq \boldsymbol{\mu}(\Gamma_{\frac{n}{n+1}}^{A_n, I}). \tag{10.6}$$

Since the set $\mathcal{A}$ is finite, the sequence $(A_n)_{n \in \mathbb{N}}$ has a subsequence $(A_{\sigma(n)})_{n \in \mathbb{N}}$ that is constant, that is, there exists $A^* \in \mathcal{A}$ such that for all $n \in \mathbb{N}$, $A_{\sigma(n)} = A^*$. From Proposition 10.3.11 and 10.3.17 we can conclude that

$$\lim_{n \in \mathbb{N}} \boldsymbol{\mu}(\Delta_{\frac{\sigma(n)}{\sigma(n)+1}}) = \boldsymbol{\mu}(\Delta_1) \text{ and } \lim_{n \in \mathbb{N}} \boldsymbol{\mu}(\Gamma_{\frac{\sigma(n)}{\sigma(n)+1}}^{A,I}) = \boldsymbol{\mu}(\Gamma_1^{A,I}).$$

From (10.6) we can deduce that $\exists A \in \mathcal{A} : \forall I \in \mathcal{I} : \boldsymbol{\mu}(\Delta_1) \sqsubseteq \boldsymbol{\mu}(\Gamma_1^{A,I})$. □

In the remainder of this dissertation, we denote the optimal max policy that satisfies Lemma 10.3.19 by A^*. As a consequence,

$$\forall I \in \mathcal{I} : \boldsymbol{\mu}(\Delta_1) \sqsubseteq \boldsymbol{\mu}(\Gamma_1^{A^*,I}). \tag{10.7}$$

Theorem 10.3.20. $\boldsymbol{\mu}(\Delta_1) = \boldsymbol{\mu}(\Gamma_1^{A^*,I^*})$.

Proof. Since

$$\begin{aligned} \boldsymbol{\mu}(\Delta_1) &\sqsubseteq \boldsymbol{\mu}(\Gamma_1^{A^*,I^*}) \qquad [(10.7)] \\ &\sqsubseteq \boldsymbol{\mu}(\Delta_1) \qquad [(10.1)] \end{aligned}$$

we can conclude that $\boldsymbol{\mu}(\Delta_1) = \boldsymbol{\mu}(\Gamma_1^{A^*,I^*})$. □

The above theorem characterizes the probabilistic bisimilarity distances for probabilistic automata in terms of a game. This characterization could form the basis for a policy iteration algorithm to compute the distances for probabilistic automata, just as the similar characterization presented in Section 6.1 forms the basis for the algorithm to compute the probabilistic bisimilarity distances for labelled Markov chains by Bacci *et al.* [3].

In the next chapter, we will introduce an algorithm which decides distance one for probabilistic automata. The game characterization of the probabilistic bisimilarity distances will be used in the correctness proof of this algorithm.

11 Distance One for Probabilistic Automata

The complexity of computing the probabilistic bisimilarity distances for probabilistic automata a la Deng *et al.* [29] was first studied by Fu [41]. He showed that these probabilistic bisimilarity distances are rational. Furthermore, he proved that the problem of deciding whether the distance of two states is smaller than a given rational is in $\mathbf{NP} \cap \mathbf{coNP}$. The proof can be adapted to show that the decision problem is in $\mathbf{UP} \cap \mathbf{coUP}$ [42]. Recall that **UP** contains those problems in **NP** with a unique accepting computation. Van Breugel and Worrell [21] have shown that the problem of computing the probabilistic bisimilarity distances is in **PPAD**, which is short for polynomial parity argument in a directed graph.

However, we are not aware of any practical algorithms to compute the distances for probabilistic automata. In this chapter, we present a characterization of distance one, which is an interplay of the greatest and least fixed point of a function. We then present a polynomial time algorithm that decides distance one for probabilistic automata. As a consequence, we can determine in polynomial time how many, if any, distances are non-trivial, that is, greater than zero and smaller than one. As

we have already shown in Chapter 9 in the context of labelled Markov chains, being able to decide distance zero and distance one in polynomial time has significant impact on computing probabilistic bisimilarity distances for labelled Markov chains.

11.1 First Attempt

As a first attempt towards capturing the set D_1, we mimick our approach taken in Section 8.1 to characterize D_1 for labelled Markov chains. However, as we will see later, this attempt will fail.

By Theorem 8.1.5, the set D_1 for a labelled Markov chain can be characterized as the greatest fixed point of the function Γ. In our first attempt, we try to mimic this idea and define D_1 as either the greatest or the least fixed point of a function. 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 there exists $s \to \mu$ such that for all $t \to \nu$ and for all couplings $\omega \in \Omega(\mu,\nu)$ we have that $\text{support}(\omega) \subseteq D_1$, or there exists $t \to \nu$ such that for all $s \to \mu$ and for all couplings $\omega \in \Omega(\mu,\nu)$ we have $\text{support}(\omega) \subseteq D_1$. This analysis suggests the following definition.

Definition 11.1.1. The function $\Xi : 2^{S^2} \to 2^{S^2}$ is defined by

$$\Xi(X) = S_1^2 \cup \left\{ (s,t) \in S_?^2 \;\middle|\; \begin{array}{l} \exists s \to \mu : \forall t \to \nu : \forall \omega \in V(\Omega(\mu,\nu)) : \text{support}(\omega) \subseteq X \vee \\ \\ \exists t \to \nu : \forall s \to \mu : \forall \omega \in V(\Omega(\nu,\mu)) : \text{support}(\omega) \subseteq X \end{array} \right\}.$$

Proposition 11.1.2. *The function Ξ is monotone.*

Proof. Let X, $Y \in 2^{S^2}$ with $X \subseteq Y$. Let $(s,t) \in \Xi(X)$. We distinguish two cases.

- If $(s,t) \in S_1^2$ then obviously $(s,t) \in \Xi(Y)$.
- If $(s,t) \in S_?^2$ then

$$\begin{aligned} & \exists s \to \mu : \forall t \to \nu : \forall \omega \in V(\Omega(\mu,\nu)) : \mathrm{support}(\omega) \subseteq X \vee \\ & \exists t \to \nu : \forall s \to \mu : \forall \omega \in V(\Omega(\nu,\mu)) : \mathrm{support}(\omega) \subseteq X \\ \text{implies} \quad & \exists s \to \mu : \forall t \to \nu : \forall \omega \in V(\Omega(\mu,\nu)) : \mathrm{support}(\omega) \subseteq Y \vee \\ & \exists t \to \nu : \forall s \to \mu : \forall \omega \in V(\Omega(\nu,\mu)) : \mathrm{support}(\omega) \subseteq Y \\ \text{implies} \quad & (s,t) \in \Xi(Y). \end{aligned}$$

□

By Knaster-Tarski's fixed point theorem (Theorem 2.1.8(a, b)), Ξ has a least and a greatest fixed point, which we denote by $\boldsymbol{\mu}(\Xi)$ and $\boldsymbol{\nu}(\Xi)$, respectively. We first show that D_1 is a fixed point of Ξ.

Proposition 11.1.3. $\Xi(D_1) = D_1$.

Proof. For all s, $t \in S$,

$$\begin{aligned} & (s,t) \in \Xi(D_1) \\ \text{iff} \quad & \ell(s) \neq \ell(t) \vee \end{aligned}$$

$$
\begin{aligned}
&(s \not\sim t \wedge \\
&\quad (\exists s \to \mu : \forall t \to \nu : \forall \omega \in V(\Omega(\mu,\nu)) : \mathrm{support}(\omega) \subseteq D_1 \vee \\
&\quad\ \exists t \to \nu : \forall s \to \mu : \forall \omega \in V(\Omega(\nu,\mu)) : \mathrm{support}(\omega) \subseteq D_1)) \\
\text{iff}\quad &(s \not\sim t \wedge \ell(s) \neq \ell(t)) \vee \\
&(s \not\sim t \wedge \\
&\quad (\exists s \to \mu : \forall t \to \nu : \forall \omega \in V(\Omega(\mu,\nu)) : \mathrm{support}(\omega) \subseteq D_1 \vee \\
&\quad\ \exists t \to \nu : \forall s \to \mu : \forall \omega \in V(\Omega(\nu,\mu)) : \mathrm{support}(\omega) \subseteq D_1)) \\
\text{iff}\quad &s \not\sim t \wedge \\
&(\ell(s) \neq \ell(t) \vee \\
&\quad (\exists s \to \mu : \forall t \to \nu : \forall \omega \in V(\Omega(\mu,\nu)) : \mathrm{support}(\omega) \subseteq D_1 \vee \\
&\quad\ \exists t \to \nu : \forall s \to \mu : \forall \omega \in V(\Omega(\nu,\mu)) : \mathrm{support}(\omega) \subseteq D_1)) \\
\text{iff}\quad &s \not\sim t \wedge \\
&(\ell(s) \neq \ell(t) \vee \\
&\quad (\exists s \to \mu : \forall t \to \nu : \forall \omega \in V(\Omega(\mu,\nu)) : \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) = 1 \vee \\
&\quad\ \exists t \to \nu : \forall s \to \mu : \forall \omega \in V(\Omega(\nu,\mu)) : \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) = 1)) \\
\text{iff}\quad &s \not\sim t \wedge \Delta_1(\boldsymbol{\mu}(\Delta_1))(s,t) = 1 \\
\text{iff}\quad &s \not\sim t \wedge \boldsymbol{\mu}(\Delta_1)(s,t) = 1 \\
\text{iff}\quad &\boldsymbol{\mu}(\Delta_1)(s,t) = 1
\end{aligned}
$$

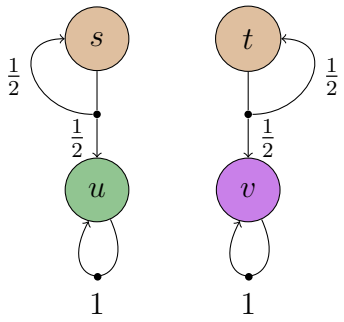
iff $(s,t) \in D_1$.

□

Proposition 11.1.4. *There exists a probabilistic automaton such that* $D_1 \neq \boldsymbol{\mu}(\Xi)$.

Proof. As we have seen in Proposition 11.1.3, D_1 is a fixed point of Ξ. We will show that for the probabilistic automaton depicted below, S_1^2 is a fixed point of Ξ as well. Obviously, states with different labels have distance one, that is, $S_1^2 \subseteq D_1$. We will demonstrate that the states s and t have distance one. Hence, $S_1^2 \subset D_1$. Therefore, D_1 is not the least fixed point of Ξ.

Consider the following probabilistic automaton.



Note that

$$
\begin{aligned}
S_0^2 &= \{(s,s),(t,t),(u,u),(v,v)\} \\
S_1^2 &= \{(s,u),(s,v),(t,u),(t,v),(u,s),(u,t),(u,v),(v,s),(v,t),(v,u)\} \\
S_?^2 &= \{(s,t),(t,s)\}
\end{aligned}
$$

Also note that

$$V(\Omega(\tfrac{1}{2}\mathrm{Dir}_s + \tfrac{1}{2}\mathrm{Dir}_u, \mathrm{Dir}_t + \tfrac{1}{2}\mathrm{Dir}_v)) \;=$$

$$\{\tfrac{1}{2}\mathrm{Dir}_{(s,t)} + \tfrac{1}{2}\mathrm{Dir}_{(u,v)}, \tfrac{1}{2}\mathrm{Dir}_{(s,v)} + \tfrac{1}{2}\mathrm{Dir}_{(u,t)}\} \tag{11.1}$$

Let us first show that S_1^2 is a fixed point of Ξ, that is, $\Xi(S_1^2) = S_1^2$. From the definition of Ξ we can immediately conclude that $S_1^2 \subseteq \Xi(S_1^2)$. To conclude that $\Xi(S_1^2) \subseteq S_1^2$, it suffices to show that $(s,t) \notin \Xi(S_1^2)$ and $(t,s) \notin \Xi(S_1^2)$. Since $(s,t) \in S_?^2$ and

$$\mathrm{support}(\tfrac{1}{2}\mathrm{Dir}_{(s,t)} + \tfrac{1}{2}\mathrm{Dir}_{(u,v)}) = \{(s,t),(u,v)\} \not\subseteq S_1^2,$$

we can conclude from (11.1) that $(s,t) \notin \Xi(S_1^2)$. Similarly, we can show that $(t,s) \notin \Xi(S_1^2)$.

It remains to show that the states s and t have distance one, that is, $(s,t) \in D_1$. First, we prove that for all $n \in \mathbb{N}$,

$$\Delta_1^n(\mathbf{0})(s,t) = 1 - 2^{-n} \tag{11.2}$$

by induction n. We distinguish two cases.

- If $n = 0$ then

$$\Delta_1^n(\mathbf{0})(s,t) = \mathbf{0}(s,t) = 0 = 1 - 2^0 = 1 - 2^{-n}.$$

- If $n > 0$ then

$\Delta_1^n(\mathbf{0})(s,t)$

$$
\begin{aligned}
&= \min\{\tfrac{1}{2}\Delta_1^{n-1}(\mathbf{0})(s,t) + \tfrac{1}{2}\Delta_1^{n-1}(\mathbf{0})(u,v), \tfrac{1}{2}\Delta_1^{n-1}(\mathbf{0})(s,v) + \tfrac{1}{2}\Delta_1^{n-1}(\mathbf{0})(u,t)\} \\
&\qquad [(11.1)] \\
&= \min\{\tfrac{1}{2}\Delta_1^{n-1}(\mathbf{0})(s,t) + \tfrac{1}{2}, \tfrac{1}{2} + \tfrac{1}{2}\} \\
&= \tfrac{1}{2}\Delta_1^{n-1}(\mathbf{0})(s,t) + \tfrac{1}{2} \\
&= \tfrac{1}{2}(1 - 2^{-(n-1)}) + \tfrac{1}{2} \qquad \text{[induction hypothesis]} \\
&= 1 - 2^{-n}.
\end{aligned}
$$

From the above, we can conclude that

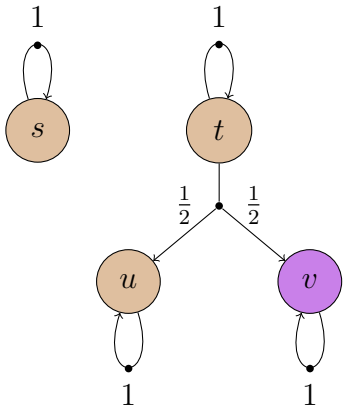
$$
\begin{aligned}
&\boldsymbol{\mu}(\Delta_1)(s,t) \\
&\quad = \sup_{n\in\mathbb{N}} \Delta_1^n(\mathbf{0})(s,t) \qquad \text{[Theorem 2.1.21]} \\
&\quad = \sup_{n\in\mathbb{N}} 1 - 2^{-n} \\
&\quad = 1.
\end{aligned}
$$

Hence, $(s,t) \in D_1$. □

Proposition 11.1.5. *There exists a probabilistic automaton such that* $D_1 \neq \boldsymbol{\nu}(\Xi)$.

Proof. We will demonstrate that the states s and t in the probabilistic automaton depicted below have distance at most $\frac{1}{2}$. As a consequence, we conclude that $D_1 = S_1^2$. As we have seen in Proposition 11.1.3, D_1 is a fixed point of Ξ. Furthermore, we will show that $D_1 \cup \{(s,t),(t,s)\}$ is a fixed point of Ξ as well. Therefore, D_1 is not the greatest fixed point of Ξ.

In this proof, we consider the following probabilistic automaton. Note that it is the probabilistic automaton in Example 10.2.2.



Note that

$$\begin{aligned} S_0^2 &= \{(s,s),(t,t),(u,u),(v,v),(s,u),(u,s)\} \\ S_1^2 &= \{(s,v),(t,u),(t,v),(u,t),(u,v),(v,s),(v,t),(v,u)\} \\ S_?^2 &= \{(s,t),(t,s)\} \end{aligned}$$

Also note that

$$\begin{aligned} V(\Omega(\mathrm{Dir}_s, \tfrac{1}{2}\mathrm{Dir}_u + \tfrac{1}{2}\mathrm{Dir}_v)) &= \{\tfrac{1}{2}\mathrm{Dir}_{(s,u)} + \tfrac{1}{2}\mathrm{Dir}_{(s,v)}\} \\ V(\Omega(\mathrm{Dir}_s, \mathrm{Dir}_t)) &= \{\mathrm{Dir}_{(s,t)}\} \end{aligned} \tag{11.3}$$

First, we will show that the states s and t have distance at most $\frac{1}{2}$. We define the function $d : S^2 \to [0,1]$ by

$$d(x,y) = \begin{cases} \frac{1}{2} & \text{if } (x,y) = (s,t) \text{ or } (x,y) = (t,s) \\ \boldsymbol{\mu}(\Delta_1)(x,y) & \text{otherwise.} \end{cases}$$

Next, we will show that d is a fixed point of Δ_1. Since $\boldsymbol{\mu}(\Delta_1)$ is the least fixed point of Δ_1, we can conclude that $\boldsymbol{\mu}(\Delta_1)(s,t) \leq d(s,t) = \frac{1}{2}$ and $\boldsymbol{\mu}(\Delta_1)(t,s) \leq d(t,s) = \frac{1}{2}$.

It remains to show that for all x, $y \in S$, $\Delta_1(d)(x,y) = d(x,y)$. We distinguish three cases.

- If $(x,y) = (s,t)$ then

$$\begin{aligned}
&\Delta_1(d)(s,t) \\
&\quad = \max\{\min\{\tfrac{1}{2}d(s,u) + \tfrac{1}{2}d(s,v), d(s,t)\}, \max\{\tfrac{1}{2}d(s,u) + \tfrac{1}{2}d(s,v), d(s,t)\}\} \\
&\quad = \max\{\min\{\tfrac{1}{2}\boldsymbol{\mu}(\Delta_1)(s,u) + \tfrac{1}{2}\boldsymbol{\mu}(\Delta_1)(s,v), \tfrac{1}{2}\}, \\
&\qquad\qquad \max\{\tfrac{1}{2}\boldsymbol{\mu}(\Delta_1)(s,u) + \tfrac{1}{2}\boldsymbol{\mu}(\Delta_1)(s,v), \tfrac{1}{2}\}\} \\
&\quad = \max\{\min\{0 + \tfrac{1}{2}, \tfrac{1}{2}\}, \max\{0 + \tfrac{1}{2}, \tfrac{1}{2}\}\} \\
&\quad = \tfrac{1}{2} \\
&\quad = d(s,t).
\end{aligned}$$

- The case that $(x,y) = (t,s)$ is similar to the previous case.

- Assume $(x,y) \neq (s,t)$ and $(x,y) \neq (t,s)$. Note that if $x \to \mu$ and $y \to \nu$ and $\omega \in V(\Omega(\mu,\nu))$ then $\mathrm{support}(\omega) \cap \{(s,t),(t,s)\} = \emptyset$. As a consequence,

$$\Delta_1(d)(x,y) = \Delta_1(\boldsymbol{\mu}(\Delta_1))(x,y) = \boldsymbol{\mu}(\Delta_1)(x,y) = d(x,y).$$

It remains to show that $D_1 \cup \{(s,t),(t,s)\}$ is a fixed point of Ξ, that is,

$$\Xi(D_1 \cup \{(s,t),(t,s)\}) = D_1 \cup \{(s,t),(t,s)\}.$$

We prove two inclusions. First of all, we observe that

$$D_1 \;=\; \Xi(D_1) \qquad \text{[Proposition 11.1.3]}$$

$$\subseteq \quad \Xi(D_1 \cup \{(s,t),(t,s)\}) \qquad \text{[Proposition 11.1.2]}$$

Because

$$\begin{aligned} \text{support}(\tfrac{1}{2}\text{Dir}_{(s,u)} + \tfrac{1}{2}\text{Dir}_{(s,v)}) &= \{(s,u),(s,v)\} &\subseteq (D_1 \cup \{(s,t),(t,s)\} \\ \text{support}(\text{Dir}_{(s,t)}) &= \{(s,t)\} &\subseteq (D_1 \cup \{(s,t),(t,s)\} \end{aligned}$$

we can conclude from the transitions of the above probabilistic automaton, the definition of Ξ and (11.3) that $(s,t) \in \Xi(D_1 \cup \{(s,t),(t,s)\})$. Similarly, we can conclude that $(t,s) \in \Xi(D_1 \cup \{(s,t),(t,s)\})$. Hence,

$$D_1 \cup \{(s,t),(t,s)\} \subseteq \Xi(D_1 \cup \{(s,t),(t,s)\}).$$

To prove the other inclusion, we conclude from the definition of Ξ that

$$\Xi(D_1 \cup \{(s,t),(t,s)\}) \subseteq S_1^2 \cup S_?^2 = D_1 \cup \{(s,t),(t,s)\}.$$

□

11.2 Deciding Distance One

In the previous section, we have shown that the set D_1 of state pairs that have distance one is neither the least fixed point of Ξ nor the greatest fixed point of Ξ. As we will prove in Section 11.3, the set D_1 can be characterized as an interplay of the greatest and the least fixed points of the function Λ of Definition 11.2.1. In this section, we present an algorithm to compute the set D_1.

Definition 11.2.1. The function $\Lambda : 2^{S^2} \times 2^{S^2} \to 2^{S^2}$ is defined by

$$\Lambda(X,Y) = S_1^2 \cup \left\{ (s,t) \in S_?^2 \;\middle|\; \begin{array}{l} \exists s \to \mu : \forall t \to \nu : \forall \omega \in V(\Omega(\mu,\nu)) : \\ \qquad \mathrm{support}(\omega) \subseteq X \wedge \mathrm{support}(\omega) \cap Y \neq \emptyset \vee \\ \exists t \to \nu : \forall s \to \mu : \forall \omega \in V(\Omega(\mu,\nu)) : \\ \qquad \mathrm{support}(\omega) \subseteq X \wedge \mathrm{support}(\omega) \cap Y \neq \emptyset \end{array} \right\}.$$

Recall from Chapter 10 that the probabilistic bisimilarity distances can be characterized in terms of an SSG. The set $\Lambda(X,Y)$ contains all state pairs with different labels and those state pairs for which there exists a move by the max player so that every subsequent move of the min player always ends up in X and with some positive probability in Y.

Let $Y \subseteq S^2$. The function $\boldsymbol{\lambda} X.\Lambda(X,Y)$ is the function that maps X to $\Lambda(X,Y)$. As we will see in Proposition 11.2.2(a), this function is monotone. Since $\langle 2^{S^2}, \subseteq \rangle$ is a complete lattice, by Theorem 2.1.8(a, b), it admits a least and a greatest fixed point, denoted by $\boldsymbol{\mu} X.\Lambda(X,Y)$ and $\boldsymbol{\nu} X.\Lambda(X,Y)$, respectively. Similarly, let $X \subseteq S^2$. The function $\boldsymbol{\lambda} Y.\Lambda(X,Y)$ is the function that maps Y to $\Lambda(X,Y)$. As we will see in Proposition 11.2.2(b), this function is monotone. Since $\langle 2^{S^2}, \subseteq \rangle$ is a complete lattice, by Theorem 2.1.8(a, b), it admits a least and a greatest fixed point, denoted by $\boldsymbol{\mu} Y.\Lambda(X,Y)$ and $\boldsymbol{\nu} Y.\Lambda(X,Y)$, respectively.

Proposition 11.2.2. *For all X, Y, $Z \subseteq S^2$ with $X \subseteq Y$,*

(a) $\Lambda(X,Z) \subseteq \Lambda(Y,Z)$.

(b) $\Lambda(Z, X) \subseteq \Lambda(Z, Y)$.

(c) $\boldsymbol{\mu} Z.\Lambda(X, Z) \subseteq \boldsymbol{\mu} Z.\Lambda(Y, Z)$.

Proof. (a) Let $(s, t) \in \Lambda(X, Z)$. We distinguish two cases.

– If $(s, t) \in S_1^2$ then obviously $(s, t) \in \Lambda(Y, Z)$.

– If $(s, t) \in S_?^2$ then

$$\begin{aligned} \exists s \to \mu : \quad & \forall t \to \nu : \forall \omega \in V(\Omega(\mu, \nu)) : \\ & \mathrm{support}(\omega) \subseteq X \wedge \mathrm{support}(\omega) \cap Z \neq \emptyset \vee \\ \exists t \to \nu : \quad & \forall s \to \mu : \forall \omega \in V(\Omega(\nu, \mu)) : \\ & \mathrm{support}(\omega) \subseteq X \wedge \mathrm{support}(\omega) \cap Z \neq \emptyset \end{aligned}$$

implies

$$\begin{aligned} \exists s \to \mu : \quad & \forall t \to \nu : \forall \omega \in V(\Omega(\mu, \nu)) : \\ & \mathrm{support}(\omega) \subseteq Y \wedge \mathrm{support}(\omega) \cap Z \neq \emptyset \vee \\ \exists t \to \nu : \quad & \forall s \to \mu : \forall \omega \in V(\Omega(\nu, \mu)) : \\ & \mathrm{support}(\omega) \subseteq Y \wedge \mathrm{support}(\omega) \cap Z \neq \emptyset \end{aligned}$$

implies

$(s, t) \in \Lambda(Y, Z)$.

(b) Let $(s, t) \in \Lambda(Z, X)$. We distinguish two cases.

– If $(s, t) \in S_1^2$ then obviously $(s, t) \in \Lambda(Z, Y)$.

– If $(s,t) \in S^2_?$ then

$$\begin{aligned}
&\exists s \to \mu : && \forall t \to \nu : \forall \omega \in V(\Omega(\mu,\nu)) : \\
& && \text{support}(\omega) \subseteq Z \wedge \text{support}(\omega) \cap X \neq \emptyset \vee \\
&\exists t \to \nu : && \forall s \to \mu : \forall \omega \in V(\Omega(\nu,\mu)) : \\
& && \text{support}(\omega) \subseteq Z \wedge \text{support}(\omega) \cap X \neq \emptyset \\
& \text{implies} \\
&\exists s \to \mu : && \forall t \to \nu : \forall \omega \in V(\Omega(\mu,\nu)) : \\
& && \text{support}(\omega) \subseteq Z \wedge \text{support}(\omega) \cap Y \neq \emptyset \vee \\
&\exists t \to \nu : && \forall s \to \mu : \forall \omega \in V(\Omega(\nu,\mu)) : \\
& && \text{support}(\omega) \subseteq Z \wedge \text{support}(\omega) \cap Y \neq \emptyset \\
& \text{implies}
\end{aligned}$$

$(s,t) \in \Lambda(Z,Y)$.

(c) We have that

$$\begin{aligned}
\Lambda(X, \boldsymbol{\mu} Z.\Lambda(Y,Z)) &\subseteq \Lambda(Y, \boldsymbol{\mu} Z.\Lambda(Y,Z)) && \text{[part (b)]} \\
&= \boldsymbol{\mu} Z.\Lambda(Y,Z) && [\boldsymbol{\mu} Z.\Lambda(Y,Z) \text{ is a fixed point of } \Lambda(Y,\cdot)]
\end{aligned}$$

that is, $\boldsymbol{\mu} Z.\Lambda(Y,Z)$ is a pre-fixed point of $\Lambda(X,\cdot)$. Since $\boldsymbol{\mu} Z.\Lambda(X,Z)$ is the least pre-fixed point of $\Lambda(X,\cdot)$ according to Theorem 2.1.8(c), we can conclude that $\boldsymbol{\mu} Z.\Lambda(X,Z) \subseteq \boldsymbol{\mu} Z.\Lambda(Y,Z)$.

□

The set $\boldsymbol{\mu}Y.\Lambda(X,Y)$ contains all state pairs (s,t) for which there exists a max policy such that for all min policies, (s,t) can reach a state pair with different labels and all state pairs reachable from (s,t) are element of X.

Since the function $\boldsymbol{\lambda}X.\boldsymbol{\mu}Y.\Lambda(X,Y)$ is monotone as well, we can conclude from Theorem 2.1.8(b) that the greatest fixed point $\boldsymbol{\nu}X.\boldsymbol{\mu}Y.\Lambda(X,Y)$ exists. The set $\boldsymbol{\nu}X.\boldsymbol{\mu}Y.\Lambda(X,Y)$ contains all state pairs (s,t) for which there exists a max policy such that for all min policies, all state pairs reachable from (s,t) can reach a state pair with different labels. In the next section, we will prove that $\boldsymbol{\nu}X.\boldsymbol{\mu}Y.\Lambda(X,Y)$ captures the set D_1. According to Theorem 2.1.9(a) and (b), these greatest and least fixed points can be obtained iteratively as follows.

```
1  X_c = S^2
2  do
3     Y_c = ∅
4     do
5        Y_p = Y_c
6        Y_c = Λ(X_c, Y_p)
7     while Y_p ≠ Y_c
8     X_p = X_c
9     X_c = Y_c
10 while X_p ≠ X_c
```

The inner loop (line 3–7) computes the least fixed point $\boldsymbol{\mu}Y.\Lambda(X_c, Y)$. The outer loop (line 1–10) computes the greatest fixed point $\boldsymbol{\nu}X.\boldsymbol{\mu}Y.\Lambda(X, Y)$, which equals D_1 as we will prove in the next section. Due to the monotonicity of Λ we can conclude that both the inner and outer loop terminate after at most $|S|^2$ iterations. To conclude that the above algorithm is polynomial time, it remains to show that $\Lambda(X_c, Y_p)$ on line 6 can be computed in polynomial time.

Proposition 11.2.3. *For all* μ, $\nu \in \mathrm{Distr}(S)$ *and* $X \subseteq S^2$,

(a)

$$\begin{aligned}
&\forall \omega \in V(\Omega(\mu,\nu)) : \mathrm{support}(\omega) \subseteq X \\
&\quad \textit{iff} \quad K(d)(\mu,\nu) = 1 \\
&\quad \textit{iff} \quad \forall \omega \in \Omega(\mu,\nu) : \mathrm{support}(\omega) \subseteq X \\
&\quad \textit{iff} \quad \mathrm{support}(\mu) \times \mathrm{support}(\nu) \subseteq X
\end{aligned}$$

(b)

$$\begin{aligned}
&\forall \omega \in V(\Omega(\mu,\nu)) : \mathrm{support}(\omega) \cap X \neq \emptyset \\
&\quad \textit{iff} \quad K(d)(\mu,\nu) > 0 \\
&\quad \textit{iff} \quad \forall \omega \in \Omega(\mu,\nu) : \mathrm{support}(\omega) \cap X \neq \emptyset
\end{aligned}$$

where

$$d(s,t) = \begin{cases} 1 & \textit{if } (s,t) \in X \\ 0 & \textit{otherwise.} \end{cases}$$

Proof. Let μ, $\nu \in \mathrm{Distr}(S)$ and $X \subseteq S^2$. Let

$$\pi = \operatorname*{argmin}_{\omega \in V(\Omega(\mu,\nu))} \sum_{(u,v)\in S^2} \omega(u,v)\, d(u,v).$$

(a) The proof consists of several parts.

– We first show that

$$\forall \omega \in V(\Omega(\mu,\nu)) : \mathrm{support}(\omega) \subseteq X$$

implies $K(d)(\mu,\nu) = 1$. Assume $\forall \omega \in V(\Omega(\mu,\nu)) : \mathrm{support}(\omega) \subseteq X$. We have

$$\begin{aligned}
& K(d)(\mu,\nu) \\
&= \min_{\omega \in V(\Omega(\mu,\nu))} \sum_{(u,v)\in S^2} \omega(u,v)\, d(u,v) \\
&= \sum_{(u,v)\in S^2} \pi(u,v)\, d(u,v) \\
&= \sum_{(u,v)\in X} \pi(u,v)\, d(u,v) + \sum_{(u,v)\in S^2\setminus X} \pi(u,v)\, d(u,v) \\
&= \sum_{(u,v)\in X} \pi(u,v) \times 1 + \sum_{(u,v)\in S^2\setminus X} \pi(u,v) \times 0 \qquad [\text{definition of } d] \\
&= \sum_{(u,v)\in X} \pi(u,v) \\
&= 1 \qquad [\mathrm{support}(\pi) \subseteq X]
\end{aligned}$$

– Next we prove that if $K(d)(\mu,\nu) = 1$, then

$$\forall \omega \in V(\Omega(\mu,\nu)) : \mathrm{support}(\omega) \subseteq X.$$

Assume that $K(d)(\mu,\nu) = 1$. Let $\omega \in V(\Omega(\mu,\nu))$. Then

$$\begin{aligned}
1 &= K(d)(\mu,\nu) \\
&\leq \sum_{(u,v)\in S^2} \omega(u,v)\, d(u,v) \\
&= \sum_{(u,v)\in X} \omega(u,v)\, d(u,v) + \sum_{(u,v)\in S^2\setminus X} \omega(u,v)\, d(u,v) \\
&= \sum_{(u,v)\in X} \omega(u,v) \times 1 + \sum_{(u,v)\in S^2\setminus X} \omega(u,v) \times 0 \qquad \text{[definition of } d\text{]} \\
&= \sum_{(u,v)\in X} \omega(u,v).
\end{aligned}$$

Hence, $\sum_{(u,v)\in X} \omega(u,v) = 1$ and, therefore, $\text{support}(\omega) \subseteq X$.

– The proof that $\forall \omega \in V(\Omega(\mu,\nu)) : \text{support}(\omega) \subseteq X$ if and only if $K(d)(\mu,\nu) = 1$ is similar to the above proof, relying on Proposition 2.1.12.

– The fact that $\forall \omega \in V(\Omega(\mu,\nu)) : \text{support}(\omega) \subseteq X$ if and only if $\text{support}(\mu) \times \text{support}(\nu) \subseteq X$ is Proposition 8.2.4.

(b) The proof consists of several parts.

– First we show that $\forall \omega \in V(\Omega(\mu,\nu)) : \text{support}(\omega) \cap X \neq \emptyset$ implies $K(d)(\mu,\nu) > 0$. Assume $\forall \omega \in V(\Omega(\mu,\nu)) : \text{support}(\omega) \cap X \neq \emptyset$. We have

$$\begin{aligned}
&K(d)(\mu,\nu) \\
&= \min_{\omega\in V(\Omega(\mu,\nu))} \sum_{(u,v)\in S^2} \omega(u,v)\, d(u,v)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{(u,v)\in S^2} \pi(u,v)\, d(u,v) \\
&= \sum_{(u,v)\in X} \pi(u,v)\, d(u,v) + \sum_{(u,v)\in S^2\setminus X} \pi(u,v)\, d(u,v) \\
&= \sum_{(u,v)\in X} \pi(u,v) \times 1 + \sum_{(u,v)\in S^2\setminus X} \pi(u,v) \times 0 \qquad [\text{definition of } d] \\
&= \sum_{(u,v)\in X} \pi(u,v) \\
&> 0 \qquad [\text{support}(\pi) \cap X \neq \emptyset]
\end{aligned}
$$

– Next, we prove that if $K(d)(\mu,\nu)>0$, then $\forall \omega \in V(\Omega(\mu,\nu)) : \text{support}(\omega) \cap X \neq \emptyset$. Assume that $K(d)(\mu,\nu) > 0$. Let $\omega \in V(\Omega(\mu,\nu))$. Then

$$
\begin{aligned}
0 &< K(d)(\mu,\nu) \\
&\leq \sum_{(u,v)\in S^2} \omega(u,v)\, d(u,v) \\
&= \sum_{(u,v)\in X} \omega(u,v)\, d(u,v) + \sum_{(u,v)\in S^2\setminus X} \omega(u,v)\, d(u,v) \\
&= \sum_{(u,v)\in X} \omega(u,v) \times 1 + \sum_{(u,v)\in S^2\setminus X} \omega(u,v) \times 0 \qquad [\text{definition of } d] \\
&= \sum_{(u,v)\in X} \omega(u,v).
\end{aligned}
$$

Hence, $\sum_{(u,v)\in X} \omega(u,v) > 0$ and, therefore, $\text{support}(\omega) \cap X \neq \emptyset$.

– The proof that $\forall \omega \in \Omega(\mu,\nu) : \text{support}(\omega) \cap X \neq \emptyset$ if and only if $K(d)(\mu,\nu) > 0$ is similar to the above proof, relying on Proposition 2.1.12.

□

Computing $K(d)(\mu,\nu)$ boils down to solving a minimum cost network flow problem, where d captures the cost. This problem can be solved in polynomial time

using, for example, Orlin's network simplex algorithm [70]. Hence, $\Lambda(X_c, Y_p)$ can be computed in polynomial time.

11.3 Correctness Proof

11.3.1 The Λ Function and the Game Characterization

This subsection collects some properties of the function Λ and relates it with the game characterization of the probabilistic bisimilarity distances for probabilistic automata.

Proposition 11.3.1. $\Lambda(D_1, D_1) = D_1$.

Proof. We will show that $\Lambda(D_1, D_1) = \Xi(D_1)$. Since $\Xi(D_1) = D_1$ according to Proposition 11.1.3, it follows that $\Lambda(D_1, D_1) = D_1$.

Assume $\omega \in \text{Distr}(S^2)$. By definition, $\text{support}(\omega) \neq \emptyset$. Hence, $\text{support}(\omega) \subseteq D_1 \wedge \text{support}(\omega) \cap D_1 \neq \emptyset$ is equivalent to $\text{support}(\omega) \subseteq D_1$. Therefore, $\Lambda(D_1, D_1) = \Xi(D_1)$. □

Recall that in Section 10.3 we have shown that $\boldsymbol{\mu}(\Delta_1) = \boldsymbol{\mu}(\Gamma_1^{A^*,I^*})$ for the optimal policies A^* and I^*. The next proposition is technical and is only used in the proof of Proposition 11.3.3.

Proposition 11.3.2. *For all $I \in \mathcal{I}$ and $(s,t) \in D_1 \setminus S_1^2$,*

$$\boldsymbol{\mu}(\Delta_1)(s,t) \leq \Gamma_1^{A^*,I}(\boldsymbol{\mu}(\Delta_1))(s,t).$$

Proof. Let $I \in \mathcal{I}$ and $(s,t) \in D_1 \setminus S_1^2$. Without any loss of generality, we may assume that $A^*(s,t) = (s,\nu)$. Let $I(s,\nu) = \pi$. Then $s \to \mu$, $t \to \nu$ and $\pi \in V(\Omega(\mu,\nu))$.

$$
\begin{aligned}
\boldsymbol{\mu}(\Delta_1)(s,t) &= \boldsymbol{\mu}(\Gamma_1^{A^*,I^*})(s,t) \qquad \text{[Theorem 10.3.20]} \\
&= \Gamma_1^{A^*,I^*}(\boldsymbol{\mu}(\Gamma_1^{A^*,I^*}))(s,t) \\
&= \Gamma_1^{A^*,I^*}(\boldsymbol{\mu}(\Delta_1))(s,t) \qquad \text{[Theorem 10.3.20]} \\
&= \sum_{u,v \in S} I^*(A^*(s,t))(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \\
&= \sum_{u,v \in S} I^*(s,\nu)(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \qquad [A^*(s,t) = (s,\nu)] \\
&= \min_{\omega \in V(\Omega(\mu_{s,\nu},\nu))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \qquad [(10.3)] \\
&= \min_{s \to \mu} \min_{\omega \in V(\Omega(\mu,\nu))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \qquad [(10.2)] \\
&\leq \min_{\omega \in V(\Omega(\mu,\nu))} \sum_{u,v \in S} \omega(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \qquad [s \to \mu] \\
&\leq \sum_{u,v \in S} \pi(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \qquad [\pi \in V(\Omega(\mu,\nu))] \\
&= \sum_{u,v \in S} I(s,\nu)(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \qquad [I(s,\nu) = \pi] \\
&= \sum_{u,v \in S} I(A^*(s,t))(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \qquad [A^*(s,t) = (s,\nu)] \\
&= \Gamma_1^{A^*,I}(\boldsymbol{\mu}(\Delta_1))(s,t).
\end{aligned}
$$

□

The next proposition is the key result in this subsection. It illustrates that in the transformed SSG, the vertices in $D_1 \setminus S_1^2$ can only reach the vertices in D_1 if the max player plays according to the optimal max policy A^* and the min player

plays an arbitrary min policy I.

Proposition 11.3.3. *For all* $(s,t) \in D_1 \setminus S_1^2$ *and* $I \in \mathcal{I}$, $\text{support}(I(A^*(s,t))) \subseteq D_1$.

Proof. Let $(s,t) \in D_1 \setminus S_1^2$. Towards a contradiction, assume there exist $I \in \mathcal{I}$ and $(x,y) \in \text{support}(I(A^*(s,t)))$ such that $\boldsymbol{\mu}(\Delta_1)(x,y) < 1$. Then

$$\begin{aligned}
\boldsymbol{\mu}(\Delta_1)(s,t) &\leq \Gamma_1^{A^*,I}(\boldsymbol{\mu}(\Delta_1))(s,t) \qquad \text{[Proposition 11.3.2]} \\
&= \sum_{u,v \in S} I(A^*(s,t))(u,v)\, \boldsymbol{\mu}(\Delta_1)(u,v) \\
&< 1 \qquad [(x,y) \in \text{support}(I(A^*(s,t))) \text{ and } \boldsymbol{\mu}(\Delta_1)(x,y) < 1]
\end{aligned}$$

This contradicts $(s,t) \in D_1$. □

Next we show that D_1 is the least fixed point of $\boldsymbol{\lambda} Y.\Lambda(D_1, Y)$. We assume $\mathbf{Y} = \boldsymbol{\mu} Y.\Lambda(D_1, Y)$. Recall that we have presented a game characterization of the distances in Section 10.3. We will use this characterization to show that $D_1 \setminus \mathbf{Y} = \emptyset$. Together with the fact that D_1 is a fixed point of $\boldsymbol{\lambda} Y.\Lambda(Y, Y)$, we can conclude that $D_1 = \mathbf{Y}$. The next proposition is technical and will only be used in the proof of Proposition 11.3.5.

Proposition 11.3.4. *If* $\mathbf{Y} = \boldsymbol{\mu} Y.\Lambda(D_1, Y)$ *and* $\mathbf{Y} \subset D_1$, *then* $\exists I \in \mathcal{I} : \forall (s,t) \in D_1 \setminus \mathbf{Y} : \boldsymbol{\mu}(\Gamma_1^{A^*,I})(s,t) = 0$.

Proof. Assume that $\mathbf{Y} = \boldsymbol{\mu} Y.\Lambda(D_1, Y)$ and $\mathbf{Y} \subset D_1$. First, we show that there exists $I \in \mathcal{I}$ such that

$$\forall (s,t) \in D_1 \setminus \mathbf{Y} : \text{support}(I(A^*(s,t))) \subseteq D_1 \setminus \mathbf{Y}$$

and then we prove

$$\forall (s,t) \in D_1 \setminus \mathbf{Y} : \boldsymbol{\mu}(\Gamma_1^{A^*,I})(s,t) = 0.$$

Let $(s,t) \in D_1 \setminus \mathbf{Y}$. Because $(s,t) \in D_1$ we can deduce that $(s,t) \notin S_0^2$. Since $(s,t) \notin \mathbf{Y} = \Lambda(D_1, \mathbf{Y})$, we have that $(s,t) \notin S_1^2$. Hence, $(s,t) \in S_?^2$.

Because $(s,t) \in S_?^2$ and $(s,t) \notin \Lambda(D_1, \mathbf{Y})$, we can conclude that

$$\begin{aligned} &\forall s \to \mu : \exists t \to \nu : \exists \omega \in V(\Omega(\mu,\nu)) : \\ &\qquad \text{support}(\omega) \not\subseteq D_1 \vee \text{support}(\omega) \cap \mathbf{Y} = \emptyset \wedge \\ &\forall t \to \nu : \exists s \to \mu : \exists \omega \in V(\Omega(\mu,\nu)) : \\ &\qquad \text{support}(\omega) \not\subseteq D_1 \vee \text{support}(\omega) \cap \mathbf{Y} = \emptyset \end{aligned} \tag{11.4}$$

Without loss of generality, assume that $A^*(s,t) = (t,\mu)$ with $s \to \mu$. By (11.4), we have that

$$\exists t \to \nu : \exists \omega \in V(\Omega(\mu,\nu)) : \text{support}(\omega) \not\subseteq D_1 \vee \text{support}(\omega) \cap \mathbf{Y} = \emptyset.$$

In the remainder of this proof we denote the coupling ω satisfying the above by $I(A^*(s,t))$. Hence,

$$\text{support}(I(A^*(s,t))) \not\subseteq D_1 \vee \text{support}(I(A^*(s,t))) \cap \mathbf{Y} = \emptyset.$$

Since $(s,t) \in D_1$ and $(s,t) \notin S_1^2$, we can conclude from Proposition 11.3.3 that

$$\text{support}(I(A^*(s,t))) \subseteq D_1$$

Combining the above, we obtain

$$\text{support}(I(A^*(s,t))) \subseteq D_1 \setminus \mathbf{Y} \tag{11.5}$$

Next, we prove for all $(s,t) \in D_1 \setminus \mathbf{Y}$, $\boldsymbol{\mu}(\Gamma_1^{A^*,I})(s,t) = 0$. To prove this, it suffices to show that $\forall n \in \mathbb{N} : (\Gamma_1^{A^*,I})^n(\mathbf{0})(s,t) = 0$ according to Theorem 2.1.21. We prove this by induction on n. The base case $n = 0$ is immediate. Let $n > 0$. Then

$$\begin{aligned}
(\Gamma_1^{A^*,I})^n(\mathbf{0})(s,t) &= \Gamma_1^{A^*,I}((\Gamma_1^{A^*,I})^{n-1}(\mathbf{0}))(s,t) \\
&= \sum_{u,v \in S} I(A^*(s,t))(u,v)\,(\Gamma_1^{A^*,I})^{n-1}(\mathbf{0})(u,v) \\
&= \sum_{(u,v) \in D_1 \setminus \mathbf{Y}} I(A^*(s,t))(u,v)\,(\Gamma_1^{A^*,I})^{n-1}(\mathbf{0})(u,v) \qquad [(11.5)] \\
&= 0 \\
&\qquad [\forall (u,v) \in D_1 \setminus \mathbf{Y} : (\Gamma^{A^*,I})^{n-1}(\mathbf{0})(u,v) = 0 \text{ by induction}]
\end{aligned}$$

□

The proposition below shows that D_1 is the least fixed point of $\boldsymbol{\lambda} Y.\Lambda(D_1, Y)$.

Proposition 11.3.5. $D_1 = \boldsymbol{\mu} Y.\Lambda(D_1, Y)$.

Proof. Let $\mathbf{Y} = \boldsymbol{\mu} Y.\Lambda(D_1, Y)$. From Proposition 11.3.1, we can conclude $\mathbf{Y} \subseteq D_1$.

It remains to prove that $D_1 \subseteq \mathbf{Y}$. Towards a contradiction, assume $\mathbf{Y} \subset D_1$. We complete the proof by showing that this assumption implies $\emptyset \subset D_1 \setminus \mathbf{Y} \subseteq D_0$.

Let $(s,t) \in D_1 \setminus \mathbf{Y}$. Then

$$\boldsymbol{\mu}(\Delta_1)(s,t) \leq \boldsymbol{\mu}(\Gamma_1^{A^*,I})(s,t) \qquad [(10.7)]$$

$$= \quad 0 \qquad \text{[Proposition 11.3.4]}$$

□

11.3.2 Iterative Characterization of $\boldsymbol{\nu} X.\boldsymbol{\mu} Y.\Lambda(X, Y)$

To conclude that the algorithm presented in Section 11.2 is correct, it remains to show that $\boldsymbol{\nu} X.\boldsymbol{\mu} Y.\Lambda(X, Y)$ equals D_1. In this subsection, we provide an iterative characterization of $\boldsymbol{\nu} X.\boldsymbol{\mu} Y.\Lambda(X, Y)$.

Definition 11.3.6. For each $i \in \mathbb{N}$, the set $X_i \subseteq S^2$ is defined by

$$X_i = \begin{cases} S^2 & \text{if } i = 0 \\ \boldsymbol{\mu} Y.\Lambda(X_{i-1}, Y) & \text{otherwise.} \end{cases}$$

For each i, $j \in \mathbb{N}$, the set $Y_i^j \subseteq S^2$ is defined by

$$Y_i^j = \begin{cases} D_1 & \text{if } j = 0 \\ \Lambda(X_i, Y_i^{j-1}) & \text{otherwise.} \end{cases}$$

The above definition differs from the iterative algorithm presented in the previous section in that $Y_i^0 = D_1$ whereas the algorithm starts its iteration towards the least fixed point from $\emptyset$.

Next, we prove a key property of the sets X_i..

Proposition 11.3.7. *For all $i \in \mathbb{N}$, $D_1 \subseteq X_i$.*

Proof. We prove it by induction on i.

- In the base case, $i = 0$, we have that

$$\begin{aligned} X_0 &= \boldsymbol{\mu} Y.\Lambda(S^2, Y) \\ &\supseteq \boldsymbol{\mu} Y.\Lambda(D_1, Y) \qquad [D_1 \subseteq S^2 \text{ and Proposition 11.2.2(c)}] \\ &= D_1 \qquad [\text{Proposition 11.3.5}] \end{aligned}$$

- Let $i > 0$. Then

$$\begin{aligned} X_i &= \boldsymbol{\mu} Y.\Lambda(X_{i-1}, Y) \\ &\supseteq \boldsymbol{\mu} Y.\Lambda(D_1, Y) \qquad [\text{by induction } D_1 \subseteq X_{i-1} \text{ and Proposition 11.2.2(c)}] \\ &= D_1 \qquad [\text{Proposition 11.3.5}] \end{aligned}$$

□

The proposition below collects two properties of Y_i^j, which will be used in the proofs later.

Proposition 11.3.8.

(a) For all $i, j \in \mathbb{N}$, $D_1 \subseteq Y_i^j$.

(b) For all $i, j \in \mathbb{N}$, $Y_i^j \subseteq Y_i^{j+1}$.

Proof. (a) Let $i \in \mathbb{N}$. We prove this proposition by induction on j. The base case, $j = 0$, is vacuously true. Let $j > 0$. Then

$$Y_i^j = \Lambda(X_i, Y_i^{j-1})$$

$$\supseteq \quad \Lambda(X_i, D_1) \qquad \text{[by induction, } D_1 \subseteq Y_i^{j-1} \text{ and Proposition 11.2.2(b)]}$$

$$\supseteq \quad \Lambda(D_1, D_1) \qquad \text{[Proposition 11.3.7 and Proposition 11.2.2(a)]}$$

$$= \quad D_1 \qquad \text{[Proposition 11.3.1]}$$

(b) Let $i \in \mathbb{N}$. We prove this proposition by induction on j.

– If $j = 0$ then

$$\begin{aligned} Y_i^0 &= D_1 \\ &= \Lambda(D_1, D_1) \qquad \text{[Proposition 11.3.1]} \\ &\subseteq \Lambda(X_i, D_1) \qquad \text{[Proposition 11.3.7 and Proposition 11.2.2(b)]} \\ &= \Lambda(X_i, Y_i^0) \\ &= Y_i^1. \end{aligned}$$

– If $j > 0$ then

$$\begin{aligned} Y_i^j &= \Lambda(X_i, Y_i^{j-1}) \\ &\subseteq \Lambda(X_i, Y_i^j) \\ &\qquad \text{[by induction, } Y_i^{j-1} \subseteq Y_i^j \text{ and Proposition 11.2.2(b)]} \\ &= Y_i^{j+1}. \end{aligned}$$

□

We conclude this subsection with the key proposition below.

Proposition 11.3.9.

(a) $X_m = \boldsymbol{\nu} X.\boldsymbol{\mu} Y.\Lambda(X, Y)$ *for some* $m \in \mathbb{N}$.

(b) $Y_m^n = \boldsymbol{\mu} Y.\Lambda(X_m, Y)$ *for some* $n \in \mathbb{N}$.

(c) $X_m = Y_m^n$.

Proof. (a) It follows from Theorem 2.1.9(b) and Proposition 11.2.2(c).

(b) First, we have

$$D_1 \subseteq X_m = \boldsymbol{\mu} Y.\Lambda(X_m, Y)$$

by Proposition 11.3.7 and part (a). The desired result follows from the latter fact and Theorem 2.1.9(c) and Proposition 11.2.2(b).

(c) It follows from part (a) and (b).

□

From Proposition 11.3.9(a) and Proposition 11.3.7, we can conclude that it suffices to prove $X_m \subseteq D_1$.

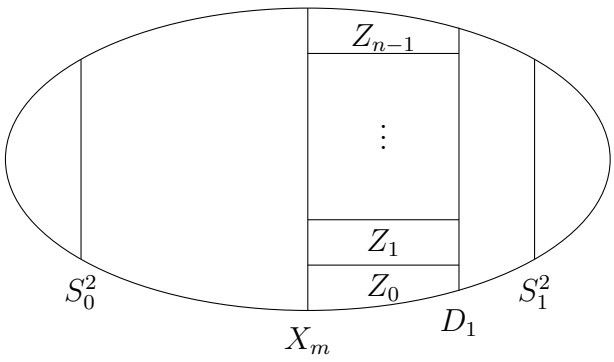
11.3.3 Max Policy A'

In this subsection, we will construct a max policy A'. The construction of A' relies on partitioning $X_m \setminus D_1$ into n disjoint subsets $Z_0, \cdots, Z_{n-1}$, which is shown by Proposition 11.3.11(b) and (d). Note that m and n are the constants from

Proposition 11.3.9. The key result in this section is Proposition 11.3.11 which collects some properties of Z_i.

Definition 11.3.10. For each $0 \leq i < n$, the set $Z_i \subseteq S^2$ is defined by

$$Z_i = Y_m^{i+1} \setminus Y_m^i.$$



We collect some properties of Z_i in the proposition below which is the key result in this subsection.

Proposition 11.3.11.

(a) For all $0 \leq i < n$, $Z_i \subseteq S^2_?$.

(b) For all $0 \leq i < j < n$, $Z_i \cap Z_j = \emptyset$.

(c) For all $1 \leq j \leq n$, $\bigcup_{0 \leq i < j} Z_i = Y_m^j \setminus D_1$.

(d) $\bigcup_{0 \leq i < n} Z_i = X_m \setminus D_1$.

(e) For all $0 \leq i \leq n$, $Y_m^i = D_1 \cup \bigcup_{0 \leq j < i} Z_j$.

Proof.

(a) Let $0 \leq i < n$. By Proposition 11.3.8(a), $S_1^2 \subseteq Y_m^i$. Furthermore, $Y_m^{i+1} = \Lambda(X_m, Y_m^i) \subseteq S_1^2 \cup S_?^2$. Hence, $Z_i \subseteq S_?^2$.

(b) Let $0 \leq i < j < n$. Because $i < j$, $Y_m^{i+1} \subseteq Y_m^j$ due to Proposition 11.3.8(b). Since also $Z_i = Y_m^{i+1} \setminus Y_m^i \subseteq Y_m^{i+1}$ and $Z_j = Y_m^{j+1} \setminus Y_m^j$, we can conclude that $Z_i \cap Z_j = \emptyset$.

(c) We prove this part by induction on j.

- If $j = 1$ then

$$Z_0 = Y_m^1 \setminus Y_m^0 = Y_m^1 \setminus D_1.$$

- If $j > 1$ then

$$\begin{aligned}
\bigcup_{0 \leq i < j} Z_i &= Z_{j-1} \cup \bigcup_{0 \leq i < j-1} Z_i \\
&= Z_{j-1} \cup (Y_m^{j-1} \setminus D_1) \qquad \text{[induction hypothesis]} \\
&= (Y_m^j \setminus Y_m^{j-1}) \cup (Y_m^{j-1} \setminus D_1) \\
&= Y_m^j \setminus D_1 \qquad [Y_m^{j-1} \subseteq Y_m^j \text{ by Proposition 11.3.8(b)}]
\end{aligned}$$

(d) Follows from part(c) and the observation that $X_m = Y_m^n$ (Proposition 11.3.9(c)).

(e) We prove that for all $Y_m^i = D_1 \cup \bigcup_{0 \leq j < i} Z_j$ by induction on i.

- The base case $i = 0$ holds by the definition of Y_m^i.

– If $i > 0$ then

$$\begin{aligned} Y_m^i &= Y_m^{i-1} \cup (Y_m^i \setminus Y_m^{i-1}) \\ &= Y_m^{i-1} \cup Z_{i-1} \\ &= (D_1 \cup \bigcup_{0 \le j < i-1} Z_j) \cup Z_{i-1} \qquad \text{[induction hypothesis]} \\ &= D_1 \cup \bigcup_{0 \le j < i} Z_j \end{aligned}$$

□

Based on the proposition below, we will construct the max policy A' of Definition 11.3.13.

Proposition 11.3.12. *For all* $0 \le i < n$ *and* $(s,t) \in Z_i$,

$$\exists s \to \mu : \forall t \to \nu : \forall \omega \in V(\Omega(\mu,\nu)) : \text{support}(\omega) \subseteq X_m \wedge \text{support}(\omega) \cap Y_m^i \neq \emptyset \vee \tag{11.6}$$

$$\exists t \to \nu : \forall s \to \mu : \forall \omega \in V(\Omega(\nu,\mu)) : \text{support}(\omega) \subseteq X_m \wedge \text{support}(\omega) \cap Y_m^i \neq \emptyset \tag{11.7}$$

Proof. Let $0 \le i < n$ and $(s,t) \in Z_i$. Then

$$\begin{aligned} (s,t) \in Z_i \qquad &\text{iff} \qquad (s,t) \in Y_m^{i+1} \setminus Y_m^i \\ &\text{implies} \quad (s,t) \in Y_m^{i+1} \\ &\text{iff} \qquad (s,t) \in \Lambda(X_m, Y_m^i) \end{aligned}$$

$$
\begin{aligned}
\text{iff} \quad & \exists s \to \mu : \forall t \to \nu : \forall \omega \in V(\Omega(\mu,\nu)) : \\
& \qquad \text{support}(\omega) \subseteq X_m \wedge \text{support}(\omega) \cap Y_m^i \neq \emptyset \vee \\
& \exists t \to \nu : \forall s \to \mu : \forall \omega \in V(\Omega(\nu,\mu)) : \\
& \qquad \text{support}(\omega) \subseteq X_m \wedge \text{support}(\omega) \cap Y_m^i \neq \emptyset \\
& \quad [(s,t) \in Z_i \subseteq S_?^2 \text{ by Proposition 11.3.11(a)}]
\end{aligned}
$$

□

Now we are ready to construct the max policy A'.

Definition 11.3.13. The function $A' : S_?^2 \to (S \times \text{Distr}(S))$ is defined by

$$
A'(s,t) = \begin{cases} (t,\mu) & \text{if } (s,t) \in Z_i \text{ and (11.6)} \\ (s,\nu) & \text{if } (s,t) \in Z_i \text{ and (11.7)} \\ A^*(s,t) & \text{if } (s,t) \in S_?^2 \setminus (X_m \setminus D_1). \end{cases}
$$

In the transformed SSG of Definition 10.2.1, the vertices in $X_m \setminus S_1^2$ can only reach the vertices in X_m, if the min player plays according to the optimal min policy I^* and the max player plays according to the max policy A'.

Proposition 11.3.14. *For all* $(s,t) \in X_m \setminus S_1^2$, $\text{support}(I^*(A'(s,t))) \subseteq X_m$.

Proof. Let $(s,t) \in X_m \setminus S_1^2$. We distinguish two cases.

- If $(s,t) \in D_1 \setminus S_1^2$, then

$$
\text{support}(I^*(A'(s,t))) \quad = \quad \text{support}(I^*(A^*(s,t)))
$$

$$[(s,t) \notin X_m \setminus D_1 \text{ since } (s,t) \in D_1]$$

$$\subseteq \quad D_1 \qquad \text{[Proposition 11.3.3]}$$

$$\subseteq \quad X_m \qquad \text{[Proposition 11.3.7]}$$

- Otherwise, $(s,t) \in X_m \setminus D_1$ since $S_1^2 \subseteq D_1$. Hence, $(s,t) \in Z_i$ for some $0 \leq i < n$ according to Proposition 11.3.11(b) and (d). Without loss of generality, assume that $A'(s,t) = (t,\mu)$. Then $s \to \mu$ according to (11.6). Assume that $I^*(t,\mu) = \omega$. Then $t \to \nu$ and $\omega \in V(\Omega(\mu,\nu))$. From (11.6) we can conclude that $\text{support}(\omega) \subseteq X_m$. Therefore, $\text{support}(I^*(A'(s,t))) \subseteq X_m$.

□

The value of the vertices in D_1 is one if the min player plays according to the optimal min policy I^* and the max player plays according to A'.

Proposition 11.3.15. *For all* $(s,t) \in D_1$, $\boldsymbol{\mu}(\Gamma_1^{A',I^*})(s,t) = 1$.

Proof. We will show that for all $i \in \mathbb{N}$ and $(s,t) \in D_1$, $(\Gamma_1^{A',I^*})^i(\mathbf{0})(s,t) = (\Gamma_1^{A^*,I^*})^i(\mathbf{0})(s,t)$. From this fact we can conclude that for all $(s,t) \in D_1$,

$$\begin{aligned}
\boldsymbol{\mu}(\Gamma_1^{A',I^*})(s,t) &= \boldsymbol{\mu}(\Gamma_1^{A^*,I^*})(s,t) && \text{[Proposition 10.3.4 and Theorem 2.1.21]} \\
&= \boldsymbol{\mu}(\Delta_1)(s,t) && \text{[Theorem 10.3.20]} \\
&= 1 && [(s,t) \in D_1]
\end{aligned}$$

The base case, $i = 0$, is vacuously true. Let $i > 0$. Let $(s,t) \in D_1$. We distinguish two cases.

- If $\ell(s) \neq \ell(t)$ then

$$\begin{aligned}(\Gamma_1^{A',I^*})^i(\mathbf{0})(s,t) &= \Gamma_1^{A',I^*}((\Gamma_1^{A',I^*})^{i-1}(\mathbf{0}))(s,t)\\ &= 1\\ &= \Gamma_1^{A^*,I^*}((\Gamma_1^{A^*,I^*})^{i-1}(\mathbf{0}))(s,t)\\ &= (\Gamma_1^{A^*,I^*})^i(\mathbf{0})(s,t).\end{aligned}$$

- If $\ell(s) = \ell(t)$ then

$$\begin{aligned}(\Gamma_1^{A',I^*})^i(\mathbf{0})(s,t) &= \Gamma_1^{A',I^*}((\Gamma_1^{A',I^*})^{i-1}(\mathbf{0}))(s,t)\\ &= \sum_{u,v\in S} I^*(A'(s,t))(u,v)\,(\Gamma_1^{A',I^*})^{i-1}(\mathbf{0})(u,v)\\ &= \sum_{(u,v)\in X_m} I^*(A'(s,t))(u,v)\,(\Gamma_1^{A',I^*})^{i-1}(\mathbf{0})(u,v)\\ &\qquad \text{[Proposition 11.3.14]}\\ &= \sum_{(u,v)\in X_m} I^*(A^*(s,t))(u,v)\,(\Gamma_1^{A',I^*})^{i-1}(\mathbf{0})(u,v)\\ &\qquad [(s,t)\in D_1\setminus S_1^2]\\ &= \sum_{(u,v)\in D_1} I^*(A^*(s,t))(u,v)\,(\Gamma_1^{A',I^*})^{i-1}(\mathbf{0})(u,v)\\ &\qquad \text{[Proposition 11.3.3]}\\ &= \sum_{(u,v)\in D_1} I^*(A^*(s,t))(u,v)\,(\Gamma_1^{A^*,I^*})^{i-1}(\mathbf{0})(u,v)\\ &\qquad \text{[by induction]}\\ &= \sum_{u,v\in S} I^*(A^*(s,t))(u,v)\,(\Gamma_1^{A^*,I^*})^{i-1}(\mathbf{0})(u,v)\\ &\qquad \text{[Proposition 11.3.3]}\end{aligned}$$

$$
\begin{aligned}
&= \Gamma_1^{A^*,I^*}((\Gamma_1^{A^*,I^*})^{i-1}(\mathbf{0}))(s,t) \\
&= (\Gamma_1^{A^*,I^*})^i(\mathbf{0})(s,t).
\end{aligned}
$$

□

The next proposition is technical and is only used in the proof of Proposition 11.3.20.

Proposition 11.3.16. *For all* $\emptyset \neq M \subseteq X_m$ *with* $\text{support}(I^*(A'(s,t))) \subseteq M$ *for all* $(s,t) \in M$*, if* $M \cap (X_m \setminus D_1) \neq \emptyset$ *then* $M \cap D_1 \neq \emptyset$*.*

Proof. Let $\emptyset \neq M \subseteq X_m$. Assume that for all $(s,t) \in M$,

$$\text{support}(I^*(A'(s,t))) \subseteq M. \tag{11.8}$$

Next, we show that

$$\forall 0 \leq i < n : M \cap Z_i \neq \emptyset \text{ implies } M \cap D_1 \neq \emptyset. \tag{11.9}$$

From Proposition 11.3.11(d) the desired result follows. We prove (11.9) by induction on i.

- Let $i = 0$. Let $(s,t) \in M \cap Z_0$. Without any loss of generality, we may assume that $A'(s,t) = (t,\mu)$. From (11.6) and the fact that $Y_m^0 = D_1$ we can conclude that

$$\text{support}(I^*(A'(s,t))) \subseteq X_m \wedge \text{support}(I^*(A'(s,t))) \cap D_1 \neq \emptyset. \tag{11.10}$$

Since $(s,t) \in M$, we have (11.8). From (11.8) and (11.10), we can conclude that $M \cap D_1 \neq \emptyset$.

- Let $i > 0$. Let $(s,t) \in M \cap Z_i$. Without any loss of generality, we may assume that $A'(s,t) = (t,\mu)$. From (11.6) we can conclude that

$$\text{support}(I^*(A'(s,t))) \subseteq X_m \wedge \text{support}(I^*(A'(s,t))) \cap Y_m^i \neq \emptyset. \qquad (11.11)$$

  Since $(s,t) \in M$, we have (11.8). From (11.8) and (11.11), we can conclude that $M \cap Y_m^i \neq \emptyset$. Let $(u,v) \in M \cap Y_m^i$. We distinguish two cases.

  - Assume $(u,v) \in D_1$. Since $(u,v) \in M$, we can conclude $M \cap D_1 \neq \emptyset$.
  - Otherwise, $(u,v) \notin D_1$. Since $(u,v) \in Y_m^i$, we have $(u,v) \in Y_m^i \setminus D_1$. From Proposition 11.3.11(c) we can deduce that $(u,v) \in Z_j$ for some $0 \leq j < i$. By the induction hypothesis, since $(u,v) \in M \cap Z_j$ for some $0 \leq j < i$, we can conclude $M \cap D_1 \neq \emptyset$.

□

11.3.4 The Function Ψ

In this subsection, we define a function Ψ to help us prove Proposition 11.3.21, that is, $X_m \subseteq D_1$. Since $X_m = \boldsymbol{\nu} X.\boldsymbol{\mu} Y.\Lambda(X,Y)$ by Proposition 11.3.9(a) and $D_1 \subseteq X_m$ by Proposition 11.3.7, we can conclude $D_1 = \boldsymbol{\nu} X.\boldsymbol{\mu} Y.\Lambda(X,Y)$.

Given the max policy A' and an arbitrary min policy I, from Proposition 11.3.11(e) and 11.3.12 we can conclude that each state pair in Z_i can reach a state pair in D_1 or Z_j with $j < i$. Consequently, each state pair in Z_i can reach a state pair in D_1. Given the max policy A' and the optimal min policy I^*, we define the function Ψ as follows.

Definition 11.3.17. The function $\Psi : [0,1]^{S^2} \to [0,1]^{S^2}$ is defined by

$$\Psi(d)(s,t) = \begin{cases} \Gamma_1^{A',I^*}(d)(s,t) & \text{if } (s,t) \in X_m \\ 0 & \text{otherwise} \end{cases}$$

Proposition 11.3.18.

(a) The function Ψ is monotone.

(b) The function Ψ is nonexpansive.

Proof.

(a) Let d, $e \in [0,1]^{S^2}$ with $d \sqsubseteq e$. Let s, $t \in S$. We distinguish two cases.

- If $(s,t) \in X_m$ then

$$\begin{aligned} \Psi(d)(s,t) &= \Gamma_1^{A',I^*}(d)(s,t) \\ &\leq \Gamma_1^{A',I^*}(e)(s,t) \qquad \text{[Proposition 10.3.4]} \\ &= \Psi(e)(s,t). \end{aligned}$$

– Otherwise,

$$\Psi(d)(s,t) = 0 = \Psi(e)(s,t).$$

(b) Let d, $e \in [0,1]^{S^2}$. Let s, $t \in S$. We distinguish two cases.

– If $(s,t) \in X_m$ then

$$|\Psi(d)(s,t) - \Psi(e)(s,t)| = |\Gamma_1^{A',I^*}(d)(s,t) - \Gamma_1^{A',I^*}(e)(s,t)| \leq \|d - e\|$$

by Proposition 10.3.4(b).

– Otherwise,

$$|\Psi(d)(s,t) - \Psi(e)(s,t)| = |0 - 0| = 0 \leq \|d - e\|.$$

□

Since $\langle [0,1]^{S^2}, \sqsubseteq \rangle$ is a complete lattice and Ψ is monotone, Ψ has a least fixed point $\boldsymbol{\mu}(\Psi)$ and a greatest fixed point $\boldsymbol{\nu}(\Psi)$ by Theorem 2.1.8(a) and (b). Next, we will show that $\boldsymbol{\mu}(\Psi)$ and $\boldsymbol{\mu}(\Gamma_1^{A',I^*})$ coincide on the state pairs in X_m.

Proposition 11.3.19. *For all* $(s,t) \in X_m$, $\boldsymbol{\mu}(\Psi)(s,t) = \boldsymbol{\mu}(\Gamma_1^{A',I^*})(s,t)$.

Proof. According to the facts that the functions Ψ and Γ_1^{A',I^*} are monotone and non-expansive (Proposition 11.3.18 and Proposition 10.3.4), we can conclude from Theorem 2.1.21 that it suffices to prove that for all $(s,t) \in X_m$ and $i \in \mathbb{N}$, $\Psi^i(\mathbf{0})(s,t) = (\Gamma_1^{A',I^*})^i(\mathbf{0})(s,t)$. We prove this by induction on i. The base case, $i = 0$, is vacuously true. Let $i > 0$. Let $(s,t) \in X_m$. We distinguish two cases.

- If $\ell(s) \neq \ell(t)$ then

$$\begin{aligned}
\Psi^i(\mathbf{0})(s,t) &= \Psi(\Psi^{i-1}(\mathbf{0}))(s,t) \\
&= \Gamma_1^{A',I^*}(\Psi^{i-1}(\mathbf{0}))(s,t) \\
&= 1 \\
&= \Gamma_1^{A',I^*}((\Gamma_1^{A',I^*})^{i-1}(\mathbf{0}))(s,t) \\
&= (\Gamma_1^{A',I^*})^i(\mathbf{0})(s,t).
\end{aligned}$$

- If $\ell(s) = \ell(t)$ then

$$\begin{aligned}
\Psi^i(\mathbf{0})(s,t) &= \Psi(\Psi^{i-1}(\mathbf{0}))(s,t) \\
&= \Gamma_1^{A',I^*}(\Psi^{i-1}(\mathbf{0}))(s,t) \\
&= \sum_{u,v \in S} I^*(A'(s,t))(u,v)\, \Psi^{i-1}(\mathbf{0})(u,v) \\
&= \sum_{(u,v) \in X_m} I^*(A'(s,t))(u,v)\, \Psi^{i-1}(\mathbf{0})(u,v) \\
&\qquad \text{[Proposition 11.3.14]} \\
&= \sum_{(u,v) \in X_m} I^*(A'(s,t))(u,v)\, (\Gamma_1^{A',I^*})^{i-1}(\mathbf{0})(u,v) \\
&\qquad \text{[induction hypothesis]} \\
&= \sum_{u,v \in S} I^*(A'(s,t))(u,v)\, (\Gamma_1^{A',I^*})^{i-1}(\mathbf{0})(u,v) \\
&\qquad \text{[Proposition 11.3.14]} \\
&= \Gamma_1^{A',I^*}(\Gamma_1^{A',I^*})^{i-1}(\mathbf{0}))(s,t) \\
&= (\Gamma_1^{A',I^*})^i(\mathbf{0})(s,t).
\end{aligned}$$

□

Proposition 11.3.20. *Ψ has a unique fixed point.*

Proof. It is sufficient to prove that $\boldsymbol{\mu}(\Psi) = \boldsymbol{\nu}(\Psi)$. Let

$$\begin{aligned} m &= \max\{\, \boldsymbol{\nu}(\Psi)(s,t) - \boldsymbol{\mu}(\Psi)(s,t) \mid (s,t) \in S^2 \,\} \\ M &= \{\, (s,t) \in S^2 \mid \boldsymbol{\nu}(\Psi)(s,t) - \boldsymbol{\mu}(\Psi)(s,t) = m \,\} \end{aligned}$$

To conclude that $\boldsymbol{\mu}(\Psi) = \boldsymbol{\nu}(\Psi)$, it suffices to show that $m = 0$. We distinguish four cases.

- Assume that $M \not\subseteq X_m$. Let $(s,t) \in M$ and $(s,t) \notin X_m$. Then

  $$\boldsymbol{\mu}(\Psi)(s,t) = \Psi(\boldsymbol{\mu}(\Psi))(s,t) = 0 = \Psi(\boldsymbol{\nu}(\Psi))(s,t) = \boldsymbol{\nu}(\Psi)(s,t).$$

  Hence,

  $$m = \boldsymbol{\nu}(\Psi)(s,t) - \boldsymbol{\mu}(\Psi)(s,t) = 0 - 0 = 0.$$

- Assume that $M \cap D_1 \neq \emptyset$. Let $(s,t) \in M$ and $(s,t) \in D_1$. By Proposition 11.3.7, $(s,t) \in X_m$. Hence,

  $$\begin{aligned} \boldsymbol{\mu}(\Psi)(s,t) &= \boldsymbol{\mu}(\Gamma_1^{A',I^*})(s,t) \qquad \text{[Proposition 11.3.19]} \\ &= 1 \qquad \text{[Proposition 11.3.15]} \end{aligned}$$

  Since

  $$\boldsymbol{\nu}(\Psi)(s,t) \geq \boldsymbol{\mu}(\Psi)(s,t) = 1,$$

we can conclude that

$$m = \boldsymbol{\nu}(\Psi)(s,t) - \boldsymbol{\mu}(\Psi)(s,t) = 1 - 1 = 0.$$

- As we will show next, it cannot be the case that $M \subseteq X_m$ and $M \cap D_1 = \emptyset$. Towards a contradiction, assume that $M \subseteq X_m$ and $M \cap D_1 = \emptyset$. Hence, we can conclude by Proposition 11.3.7 that $M \subseteq X_m \setminus D_1$.

$$\begin{aligned}
m &= \boldsymbol{\nu}(\Psi)(s,t) - \boldsymbol{\mu}(\Psi)(s,t) \\
&= \Psi(\boldsymbol{\nu}(\Psi))(s,t) - \Psi(\boldsymbol{\mu}(\Psi))(s,t) \\
&= \Gamma_1^{A',I^*}(\boldsymbol{\nu}(\Psi))(s,t) - \Gamma_1^{A',I^*}(\boldsymbol{\mu}(\Psi))(s,t) \qquad [(s,t) \in X_m] \\
&= \sum_{u,v \in S} I^*(A'(s,t))(u,v)\, \boldsymbol{\nu}(\Psi)(u,v) - \sum_{u,v \in S} I^*(A'(s,t))(u,v)\, \boldsymbol{\mu}(\Psi)(u,v) \\
&\qquad [(s,t) \in S_?^2] \\
&= \sum_{u,v \in S} I^*(A'(s,t))(u,v)\, (\boldsymbol{\nu}(\Psi)(u,v) - \boldsymbol{\mu}(\Psi)(u,v))
\end{aligned}$$

and $\boldsymbol{\nu}(\Psi)(u,v) - \boldsymbol{\mu}(\Psi)(u,v) \leq m$ for all $u,\ v \in S$, we can conclude that $\text{support}(I^*(A'(s,t))) \subseteq M$. From the above and Proposition 11.3.16, we can conclude $M \cap D_1 \neq \emptyset$. This contradicts the assumption that $M \cap D_1 = \emptyset$.

□

From the fact that Ψ has a unique fixed point and the alternative characterization of the probabilistic bisimilarity distances presented in Section 10.3, we can infer the key result in this section, that is, $X_m \subseteq D_1$.

Proposition 11.3.21. $X_m \subseteq D_1$.

Proof. We define the function $d \in S^2 \to [0,1]$ by

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

Next, we will show that d is a fixed point of Ψ, that is, for all s, $t \in S$, $\Psi(d)(s,t) = d(s,t)$. Let s, $t \in S$. We distinguish three cases.

- If $(s,t) \notin X_m$, then

$$\Psi(d)(s,t) = 0 = d(s,t).$$

- If $\ell(s) \neq \ell(t)$ then

$$\begin{aligned} \Psi(d)(s,t) &= \Gamma_1^{A',I^*}(d)(s,t) \qquad [S_1^2 \subseteq X_m] \\ &= 1 \\ &= d(s,t) \qquad [S_1^2 \subseteq X_m] \end{aligned}$$

- Otherwise, $(s,t) \in X_m$ and $\ell(s) = \ell(t)$. Then

$$\begin{aligned} \Psi(d)(s,t) &= \Gamma_1^{A',I^*}(d)(s,t) \\ &= \sum_{u,v \in S} I^*(A'(s,t))(u,v)\, d(u,v) \\ &= \sum_{(u,v) \in X_m} I^*(A'(s,t))(u,v)\, d(u,v) \qquad \text{[Proposition 11.3.14]} \\ &= 1 \qquad [d(u,v) = 1 \text{ for all } (u,v) \in X_m] \\ &= d(s,t). \end{aligned}$$

Let $(s,t) \in X_m$. Then

$$\begin{aligned}
\boldsymbol{\mu}(\Delta_1)(s,t) &\geq \boldsymbol{\mu}(\Gamma_1^{A',I^*})(s,t) \qquad [(10.1)] \\
&= \boldsymbol{\mu}(\Psi)(s,t) \qquad [\text{Proposition 11.3.19}] \\
&= d(s,t) \qquad [\text{Proposition 11.3.20}] \\
&= 1.
\end{aligned}$$

Hence, $(s,t) \in D_1$. □

Theorem 11.3.22. $D_1 = \boldsymbol{\nu} X.\boldsymbol{\mu} Y.\Lambda(X,Y)$.

Proof. Immediately consequence of Proposition 11.3.9(a), Proposition 11.3.7 and Proposition 11.3.21. □

12 Conclusion

In this dissertation, we have presented our work on algorithms to compute the probabilistic bisimilarity distances for labelled Markov chains and probabilistic automata. In particular, we have focused on the policy iteration algorithms for labelled Markov chains.

12.1 Algorithms for Labelled Markov Chains

The first step of our work was reviewing the algorithms in the literature. We have reviewed the algorithm which uses the first order theory over the reals and the one which uses Khachiyan's ellipsoid method. We also have reviewed the (partial) policy iteration algorithm by Bacci *et al.* [3].

To compute the distances correctly, we have slightly modified the algorithm by Bacci *et al.* [3], that is running the procedure of deciding distance zero before running the simple policy iteration algorithm. We have shown that it is a small, yet essential modification. We have also presented the general policy iteration algorithm and have proved the correctness of this algorithm.

The basic algorithm of Bacci *et al.* with the on-the-fly optimization is to compute the distances for only a few state pairs. We have provided a counterexample showing that the original algorithm does not always consider sufficiently many state pairs. We have modified the algorithm and proved our modification correct. Furthermore, we have generalized the general policy iteration algorithm to use partial policies.

We have proved an exponential lower bound for the simple (partial) policy iteration algorithm. Note that although the simple (partial) policy algorithm is exponential time in the worst case, in practice it is much faster than the polynomial-time algorithm which uses the ellipsoid method, as can be seen in Chapter 9.

As shown by Derisavi, Hermanns and Sanders in [30] and also by Valmari and Franceschinis [92], probabilistic bisimilarity distance zero for labelled Markov chains can be decided in $O(m \log n)$, where n and m are the number of states and transitions of the labelled Markov chain. In this dissertation, we have shown that distance one can also be decided in polynomial time. As a consequence, we can determine in polynomial time how many, if any, distances are non-trivial, that is, greater than zero and smaller than one. We have developed three new algorithms in which we compute the number of non-trivial distances first. As we have shown in Chapter 9, the algorithm by Bacci *et al.* [3] ($D_0 + \text{SPI}$), that does not decide distance one before computing the non-trivial distances using policy iteration, can compute distances for labelled Markov chains up to 150 states. For one such labelled Markov

chain, their algorithm takes more than 49 hours. The new algorithm that we have presented in Section 8.3.1 takes 13 milliseconds instead of 49 hours. Furthermore, the new algorithm can compute distances for labelled Markov chains with more than 10,000 states in less than 50 minutes.

12.2 Algorithms for Probabilistic Automata

Inspired by the algorithm which uses the first order theory over the reals for labelled Markov chains, we have developed an algorithm to compute the distances for probabilistic automata. We have proposed an alternative characterization of the probabilistic bisimilarity distances in terms of a simple stochastic game, which may form a basis of a policy iteration algorithm. Moreover, we have presented a polynomial-time algorithm to decide distance one.

12.3 Future Work

In this section, we briefly discuss some possible avenues for future work. There are two main directions. Firstly, the worst-case running time of general (partial) policy iteration and the expected running time of the randomized policy iteration to compute the distances for labelled Markov chains remain unknown. Secondly, no policy iteration algorithm to compute the distances for probabilistic automata has been developed yet.

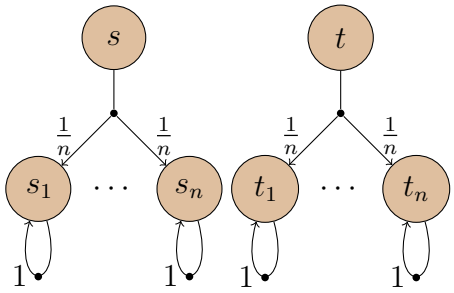
12.3.1 Time Complexity of Other Policy Iteration Algorithms for Labelled Markov Chains

We have proved that the simple policy iteration algorithm for labelled Markov chains runs in exponential time in the worst case. The general policy iteration algorithm for infinite-horizon Markov decision processes with total-reward optimality criteria is exponential time in the worst case [35]. However, it is unclear if the exponential lower bound holds for those Markov decision processes which are transformed from labelled Markov chains. We are also interested in the expected running time of the randomized policy iteration algorithm. It has been an open problem for more than fifty years whether closely related randomized algorithms run in expected polynomial time.

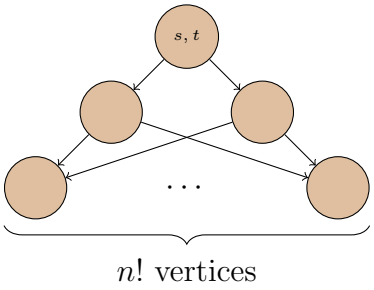
12.3.2 Policy Iteration Algorithms for Probabilistic Automata

In Section 10.3, we have presented an alternative characterization of the probabilistic bisimilarity distances for probabilistic automata. In future work, we plan to use this characterization as the basis for an algorithm to compute the probabilistic bisimilarity distances for probabilistic automata based on the policy iteration algorithm due to Hoffman and Karp [52].

Consider the following probabilistic automaton.



This probabilistic automaton induces the following game graph.



If μ and ν are both the uniform distribution on n elements, then the vertices of $\Omega(\mu, \nu)$ can be viewed as permutations (see, for example, [81, Theorem 8.4]). As a result, from the state pair (s, t) after one move by the max player and one move by the min player, $n!$ vertices can be reached. Hence, we may encounter an exponential blow-up when we transform a probabilistic automaton into a game. As a consequence, it is not immediately obvious which results from game theory can be transferred to our setting. We leave this for future research.

To prove Lemma 10.3.19, which provides the second part of the proof of the alternative characterization of the probabilistic bisimilarity distances, we rely on the discounted functions Δ_c and $\Gamma_c^{A_c^*, I}$ for $c \in (0, 1)$. In particular, in the proof of Proposition 10.3.18 we use the fact that $\Gamma_c^{A_c^*, I}$ has a unique fixed point. If we were

able to prove that $\Gamma_1^{A^*,I}$ has a unique fixed point, then we would be able to give a proof of Lemma 10.3.19 that does not rely on discounted functions. We also leave that for future research.

Bibliography

[1] Luca Aceto, Anna Ingolfsdottir, Kim Larsen, and Jiří Srba. *Reactive systems: Modelling, specification and verification*. Cambridge University Press, Cambridge, United Kingdom, 2003.

[2] Ravindra Ahuja, Thomas Magnanti, and James Orlin. *Network flows: Theory, algorithms, and applications*. Prentice-Hall, Upper Saddle River, NJ, USA, 1993.

[3] 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.

[4] Giorgio Bacci, Giovanni Bacci, Kim Larsen, and Radu Mardare. The behavior of probabilistic systems: From equivalences to behavioral distances. In Martin Abadi, Philippa Gardner, Andrew Gordon, and Radu Mardare, editors, *Essays for the Luca Cardelli Fest*, pages 15–26, Cambridge, United Kingdom, September 2014. Microsoft Research.

[5] Giorgio Bacci, Giovanni Bacci, Kim Larsen, and Radu Mardare. On-the-fly computation of bisimilarity distances. *Logical Methods in Computer Science*, 13(2), June 2017.

[6] Giovanni Bacci, Giorgio Bacci, Kim Larsen, and Radu Mardare. On the metric-based approximate minimization of Markov chains. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming*, volume 80 of *Leibniz International Proceedings in Informatics*, pages 104:1–104:14, Warsaw, Poland, July 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[7] 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.

[8] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, Cambridge, MA, USA, 2008.

[9] Jaco de Bakker and Erik de Vink. *Control flow semantics*. MIT Press, Cambridge, MA, USA, 1996.

[10] Jaco de Bakker and Jeffery Zucker. Denotational semantics of concurrency. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, pages 153–158, San Francisco, CA, USA, May 1982. ACM.

[11] Stefan Banach. Sur les opérations dans les ensembles abstraits et leurs applications aux equations intégrales. *Fundamenta Mathematicae*, 3:133–181, 1922.

[12] Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

[13] Patrick Billingsley. *Probability and measure*. Wiley Series in Probability and Statistics. Wiley, New York, NY, USA, 3rd edition, 1995.

[14] Franck van Breugel. On behavioural pseudometrics and closure ordinals. *Information Processing Letters*, 112(18):715–718, October 2012.

[15] Franck van Breugel. Probabilistic bisimilarity distances. *ACM SIGLOG News*, 4(4):33–51, November 2017.

[16] Franck van Breugel, Claudio Hermida, Michael Makkai, and James Worrell. Recursively defined metric spaces without contraction. *Theoretical Computer Science*, 380(1/2):143–163, June 2007.

[17] Franck van Breugel, Steven Shalit, and James Worrell. Testing labelled Markov processes. In Peter Widmayer, Stephan Eidenbenz, Francisco Triguero, Rafael Morales, Ricardo Conejo, and Matthew Hennessy, editors, *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 537–548, Malaga, Spain, July 2002. Springer-Verlag.

[18] Franck van Breugel, Babita Sharma, and James Worrell. Approximating a behavioural pseudometric without discount for probabilistic systems. In Helmut Seidl, editor, *Proceedings of 10th International Conference on Foundations of Software Science and Computational Structures*, volume 4423 of *Lecture Notes in Computer Science*, pages 123–137, Braga, Portugal, March 2007. Springer-Verlag.

[19] Franck van Breugel, Babita Sharma, and James Worrell. Approximating a behavioural pseudometric without discount. *Logical Methods in Computer Science*, 4(2), April 2008.

[20] Franck van Breugel and James Worrell. Towards quantitative verification of probabilistic systems. In Fernando Orejas, Paul Spirakis, and Jan van Leeuwen, editors, *Proceedings of 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 421–432, Crete, July 2001. Springer-Verlag.

[21] 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, Oxford, United Kingdom, May 2014.

[22] 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.

[23] 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.

[24] Taolue Chen, Tingting Han, and Jian Lu. On behavioral metric for probabilistic systems: definition and approximation algorithm. In *Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery*, pages 21–25, Haikou, China, August 2007. IEEE.

[25] Taolue Chen, Tingting Han, and Jian Lu. On metrics for probabilistic systems: definitions and algorithms. *Computers and Mathematics with Applications*, 57(6):991–999, March 2009.

[26] Edmund Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 1999.

[27] Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, February 1992.

[28] Brian Davey and Hilary Priestley. *Introduction to lattices and order*. Cambridge University Press, Cambridge, United Kingdom, 2002.

[29] Yuxin Deng, Tom Chothia, Catuscia Palamidessi, and Jun Pang. Metrics for action-labelled quantitative transition systems. In Antonio Cerone and Herbert Wiklicky, editors, *Proceedings of 3rd Workshop on Quantitative Aspects of Programming Languages*, volume 153(2) of *Electronic Notes in Theoretical Computer Science*, pages 79–96, Edinburgh, United Kingdom, April 2005. Elsevier.

[30] Salem Derisavi, Holger Hermanns, and William Sanders. Optimal state-space lumping in Markov chains. *Information Processing Letters*, 87(6):309–315, September 2003.

[31] 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.

[32] Josée Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. The metric analogue of weak bisimulation for probabilistic processes. In *Proceedings of 17th Annual IEEE Symposium on Logic in Computer Science*, pages 413–422, Copenhagen, Denmark, July 2002. IEEE.

[33] Josee Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Metrics for labelled Markov processes. *Theoretical Computer Science*, 318(3):323–354, June 2004.

[34] Allen Emerson and Edmund Clarke. Characterizing correctness properties of parallel programs using fixpoints. In Jaco de Bakker and Jan van Leeuwen, editors, *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, volume 85 of *Lecture Notes in Computer Science*, pages 169–181, Noordwijkerhout, The Netherlands, July 1980. Springer-Verlag.

[35] John Fearnley. Exponential lower bounds for policy iteration. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide,

and Paul Spirakis, editors, *Proceedings of the 37th Colloquium on Automata, Languages and Programming*, volume 6199 of *Lecture Notes in Computer Science*, pages 551–562, Bordeaux, France, July 2010. Springer-Verlag.

[36] Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous Markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011.

[37] Norman Ferns and Doina Precup. Bisimulation metrics are optimal value functions. In Nevin Zhang and Jin Tian, editors, *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pages 210–219, Quebec City, QC, Canada, July 2014. AUAI Press.

[38] Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer-Verlag, New York, NY, USA, 1997.

[39] Ivana Filipovic, Peter O'Hearn, Noam Rinetzky, and Hongseok Yang. Abstraction for concurrent objects. In *Proceedings of the 18th European Symposium on Programming*, volume 5502 of *Lecture Notes in Computer Science*, pages 252–266, York, UK, March 2009. Springer-Verlag.

[40] Oliver Friedmann. *Exponential lower bounds for solving infinitary payoff games and linear programs*. PhD thesis, Ludwig-Maximilians-University, Munich, Germany, 2011.

[41] Hongfei Fu. Computing game metrics on Markov decision processes. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming*, volume 7392 of *Lecture Notes in Computer Science*, pages 227–238, Warwick, UK, July 2012. Springer-Verlag.

[42] Hongfei Fu. Personal communication, January 2013.

[43] 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.

[44] Hugo Gimbert. Pure stationary optimal strategies in Markov decision processes. In Wolfgang Thomas and Pascal Weil, editors, *Proceedings of the 24th Annual Symposium on Theoretical Aspects of Computer Science*, volume 4393 of *Lecture Notes in Computer Science*, pages 200–211, Aachen, Germany, February 2007. Springer-Verlag.

[45] Patrice Godefroid and Pierre Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. In Kim Larsen and Arne Skou, editors, *Proceedings of the 3rd International Workshop on Computer Aided Verification*, volume 575 of *Lecture Notes in Computer Science*, pages 332–342, Aalborg, Denmark, 1991. Springer-Verlag.

[46] Thomas Dueholm Hansen. *Worst-case analysis of strategy iteration and the simplex method*. PhD thesis, Aarhus University, Aarhus, Denmark, July 2012.

[47] Felix Hausdorff. *Grundzüge der Mengenlehre*. Von Veit & Comp., Leipzig, 1914.

[48] Leen Helmink, Alex Sellink, and Frits Vaandrager. Proof-checking a data link protocol. In Henk Barendregt and Tobias Nipkow, editors, *Proceedings of the International Workshop on Types for Proofs and Programs*, volume 806 of *Lecture Notes in Computer Science*, pages 127–165, Nijmegen, The Netherlands, May 1993. Springer-Verlag.

[49] Maurice P. Herlihy and Jeannette M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems*, 12(3):463–492, July 1990.

[50] Ted Herman. Probabilistic self-stabilization. *Information Processing Letters*, 35(2):63–67, June 1990.

[51] Frank Hitchcock. The distribution of a product from several sources to numerous localities. *Studies in Applied Mathematics*, 20(1/4):224–230, April 1941.

[52] Alan Hoffman and Richard Karp. On nonterminating stochastic games. *Management Science*, 12(5):359–370, January 1966.

[53] Ronald Howard. *Dynamic programming and Markov processes*. MIT Press, Cambridge, MA, USA, 1960.

[54] Aron Itai and Michael Rodeh. Symmetry breaking in distributed networks. *Information and Computation*, 88(1):60–87, September 1990.

[55] 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.

[56] Leonid Kantorovich. On the transfer of masses (in Russian). *Doklady Akademii Nauk*, 5(1):1–4, 1942. Translated in *Management Science*, 5(1):1–4, October 1958.

[57] Joost-Pieter Katoen, Tim Kemna, Ivan Zapreev, and David Jansen. Bisimulation minimisation mostly speeds up probabilistic model checking. In Orna Grumberg and Michael Huth, editors, *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 4424 of *Lecture Notes in Computer Science*, pages 87–101, Braga, Portugal, March/April 2007. Springer-Verlag.

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

[59] James King. Symbolic execution and program testing. *Communications of the ACM*, 19(7):385–394, July 1976.

[60] Viktor Klee and Christoph Witzgall. Facets and vertices of transportation polytopes. In George Dantzig and Arthur 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.

[61] Bronisław Knaster. Un théorème sur les fonctions d'ensembles. *Annales de la Société Polonaise de Mathématique*, 6:133–134, 1928.

[62] Donald Knuth and Andrew Yao. The complexity of nonuniform random number generation. In Joseph Traub, editor, *Proceedings of a Symposium on New Directions and Recent Results in Algorithms and Complexity*, pages 375–428, Pittsburgh, PA, USA, April 1976. Academic Press.

[63] 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.

[64] 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.

[65] Kim Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, September 1991.

[66] Radu Mardare, Prakash Panangaden, and Gordon Plotkin. Quantitative algebraic reasoning. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 700–709, New York, NY, USA, July 2016. IEEE Computer Society Press.

[67] Mary Melekopoglou and Anne Condon. On the complexity of the policy iteration algorithm. Computer Science Technical Report 941, University of Wisconsin, Madison, WI, USA, June 1990.

[68] Robin Milner. *A calculus of communicating systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Germany, 1980.

[69] Abhishek Murthy, Md. Ariful Islam, Ezio Bartocci, Elizabeth Cherry, Flavio Fenton, James Glimm, Scott Smolka, and Radu Grosu. Approximate bisimulations for sodium channel dynamics. In David Gilbert and Monika Heiner, editors, *Proceedings of 10th International Conference on Computational Methods in Systems Biology*, volume 7605 of *Lecture Notes in Computer Science*, pages 267–287, London, United Kingdom, 2012. Springer-Verlag.

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

[71] Prakash Panangaden. *Labelled Markov processes*. Imperial College Press, London, United Kingdom, 2009.

[72] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., Englewood Cliffs, NJ, USA, 1982.

[73] 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.

[74] Stefan Ratschan. Efficient solving of quantified inequality constraints over the real numbers. *ACM Transactions on Computational Logic*, 7(4):723–748, October 2006.

[75] Steffen Rebennack. Ellipsoid method. In Christodoulos Floudas and Panos Pardalos, editors, *Encyclopedia of Optimization*, pages 890–899. Springer-Verlag, New York, NY, USA, 2009.

[76] Davide Sangiorgi. *Introduction to bisimulation and coinduction*. Cambridge University Press, Cambridge, United Kingdom, 2012.

[77] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Chichester, United Kingdom, 1986.

[78] Roberto Segala. *Modeling and verification of randomized distributed real-time systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, June 1995.

[79] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. In Bengt Jonsson and Joachim Parrow, editors, *Proceedings of the 5th International Conference on Concurrency Theory*, volume 836 of *Lecture Notes in Computer Science*, pages 481–496, Uppsala, Sweden, August 1994. Springer-Verlag.

[80] Prithviraj Sen, Amol Deshpande, and Lise Getoor. Bisimulation-based approximate lifted inference. In Jeff Bilmes and Andrew Ng, editors, *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 496–505, Montreal, QC, Canada, 2009. AUAI Press.

[81] Denis Serre. *Matrices: theory and applications*. Springer-Verlag, New York, NY, USA, 2010.

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

[83] James Strayer. *Linear programming and its applications*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, NY, USA, 1989.

[84] Richard Sutton and Andrew Barto. *Reinforcement learning: an introduction*. MIT Press, Cambridge, MA, USA, 1998.

[85] 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.

[86] Qiyi Tang and Franck van Breugel. Algorithms to compute probabilistic bisimilarity distances for labelled Markov chains. In Roland Meyer and Uwe Nestmann, editors, *Proceedings of the 28th International Conference on Concurrency Theory*, volume 85 of *Leibniz International Proceedings in Informatics*, pages 27:1–27:16, Berlin, Germany, September 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[87] Qiyi Tang and Franck van Breugel. Deciding probabilistic bisimilarity distance one for labelled Markov chains. In Hana Chockler and Georg Weissenbacher,

editors, *Proceedings of the 30th International Conference on Computer Aided Verification*, volume 10981 of *Lecture Notes in Computer Science*, pages 681–699, Oxford, UK, July 2018. Springer-Verlag.

[88] Qiyi Tang and Franck van Breugel. Deciding probabilistic bisimilarity distance one for probabilistic automata. To appear in Proceedings of the 29th International Conference on Concurrency Theory, September 2018.

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

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

[91] Mathieu Tracol, Josée Desharnais, and Abir Zhioua. Computing distances between probabilistic automata. In Mieke Massink and Gethin Norman, editors, *Proceedings 9th Workshop on Quantitative Aspects of Programming Languages*, volume 57 of *Electronic Proceedings in Theoretical Computer Science*, pages 148–162, Saarbrücken, Germany, April 2011.

[92] Antti Valmari and Giuliana Franceschinis. Simple $O(m \log n)$ time Markov chain lumping. In Javier Esparza and Rupak Majumdar, editors, *Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 6015 of *Lecture Notes in Computer Science*, pages 38–52, Paphos, Cyprus, March 2010. Springer-Verlag.

[93] 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.

[94] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1/2):343–359, May 1996.