Computing Minimum and Maximum Reachability Times in Probabilistic Systems[*]

Luca de Alfaro

Department of Electrical Engineering and Computer Sciences,
University of California at Berkeley.
Email: `dealfaro@eecs.berkeley.edu`

Abstract

A Markov decision process is a generalization of a Markov chain in which both probabilistic and nondeterministic choice coexist. Given a Markov decision process with costs associated with the transitions and a set of target states, the *stochastic shortest path* problem consists in computing the minimum expected cost of a control strategy that guarantees to reach the target. In this paper, we consider the classes of stochastic shortest path problems in which the costs are all non-negative, or all non-positive. Previously, these two classes of problems could be solved only under the assumption that the policies that minimize or maximize the expected cost also lead to the target with probability 1. This assumption does not necessarily hold for Markov decision processes that arise as model for distributed probabilistic systems. We present efficient methods for solving these two classes of problems without relying on additional assumptions. The methods are based on algorithms to transform the original problems into problems that satisfy the required assumptions. The methods lead to the efficient solution of two basic problems in the analysis of the reliability and performance of partially-specified systems: the computation of the minimum (or maximum) probability of reaching a target set, and the computation of the minimum (or maximum) expected time to reach the set.

1 Introduction

Markov decision processes are generalizations of Markov chains in which probabilistic choice coexists with nondeterministic choice [Bel57]. Several models of distributed probabilistic systems are based either on Markov decision processes [BdA95, KB98] or on closely related formalisms, such as the *concurrent Markov chains* of [Var85], the *probabilistic automata* of [SL94, WSS94], and the *timed probabilistic automata* of [Seg95]. Several models based on process algebras are also closely related to Markov decision processes

[*]An abbreviated version of this paper is to appear in *Proceedings of CONCUR 99: Concurrency Theory.* This research was supported in part by the NSF CAREER award CCR-9501708, by the DARPA (NASA Ames) grant NAG2-1214, by the DARPA (Wright-Patterson AFB) grant F33615-98-C-3614, by the ARO MURI grant DAAH-04-96-1-0341, and by the Gigascale Silicon Research Center.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **1999**		2. REPORT TYPE		3. DATES COVERED **00-00-1999 to 00-00-1999**	
4. TITLE AND SUBTITLE **Computing Minimum and Maximum Reachability Times in Probabilistic Systems**				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Berkeley,Department of Electrical Engineering and Computer Sciences,Berkeley,CA,94720-1770**				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES **17**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

[LS89, JL91, WSS94]. In these proposals, probability enables the modeling of phenomena related to reliability and performance, while nondeterminism has been used to model concurrency [Var85, PZ86, Seg95], inputs [Seg95], imprecise knowledge of the transition probabilities [dA97, dA98], and in general any behavior for which probabilistic information is not known.

A Markov decision process (MDP) consists of a set of states; with each state is associated a set of possible actions. At every state, the choice of the next action is nondeterministic; once chosen, the action determines the transition probability distribution for the successor state. In order to quantify the probabilistic properties of an MDP, the concept of *policy* is introduced [Der70], related to the *schedulers* of [Var85, PZ86] and to the *adversaries* of [SL94, Seg95]. A policy is a criterion for selecting the actions during a behavior of the system; once the policy is fixed, the MDP is reduced to a conventional stochastic process. A simple way to introduce time in these models is to associate with each pair consisting of state and of a related action the time (or the expected time) spent at the state when the action is selected [Han94, Seg95, dA98]. One of the basic questions we can ask about the timing behavior of such a system is the expected time needed to reach a given set of target states from a specified starting state. Being able to answer this question opens the way to the automated verification of systems properties such as expected time to failure, expected task completion time, and several others. Since the system model includes nondeterminism, the answer to this expected time question consists not in a single value, but rather in a range of values comprised between a minimum and a maximum, depending on whether the policy in use hastens or delays the reaching of the target. This paper is concerned with the question of how to compute these minimum and maximum values.

The problem of computing the maximum and minimum reachability times can be reduced to the *stochastic shortest path* (SSP) problem [EZ62, Der70]. In the statement of the SSP problem, with each state-action pair is associated a real-valued cost; the SSP problem consists in computing the minimum expected cost incurred to reach a set of target states. Hence, to compute the minimum (resp. maximum) reachability time, it suffices to equate the cost to the time (resp. to the time multiplied by -1) and to solve the resulting SSP problem. However, previous solutions to the SSP problem rely on assumptions that do not necessarily hold for the SSP problems obtained by the above reduction. In particular, previous solutions require that the target set can be reached with probability 1 from every state, and that either (a) every policy that does not lead to the target with probability 1 yields infinite expected total cost, or (b) the policies that minimize or maximize the expected total cost also lead to the target with probability 1 [BT91, Ber95]. Under either one of these assumptions, the goal of reaching the target can be disregarded in the solution of the optimization problem, and the SSP problem can be solved by determining the policy that minimizes the total cost. If the starting and target states are part of a formal specification, or if the time associated with state-action pairs can be 0, as in [Han94, Seg95, dA97, dA98], these assumptions do not hold in general, and new solution methods are required.

The aim of this paper is to present methods for solving the SSP problem that rely on the assumptions that the costs are all non-negative, or all non-positive. We call the SSP problems that satisfy these assumptions the *non-negative* and *non-positive* SSP problems.

Solving these SSP problems suffices for solving the original problem about the maximum and minimum reachability times. Furthermore, we show that the proposed solution methods can be applied to the efficient computation of the maximum and minimum probability of reaching a target set of states.

The minimum expected cost to reach a set of target states is well defined only if the target can be reached with probability 1. The first step in the solution of the SSP problem consists thus in computing the set of states from which the target set can be reached with probability 1. This problem can be solved in polynomial time by a reduction to linear programming [Der70]. In this paper we present a more efficient algorithm, that solves the problem in time quadratic in the size of the MDP, and that does not require numerical computation. The algorithm, originating from [dA97], is related to an algorithm for solving two-person reachability games presented in [dAHK98].

Once we have determined the states from which the target set cannot be reached with probability 1, we present two methods for solving the SSP problem on the remaining states. First, we show that non-negative and non-positive SSP problems can be solved using linear programming over the extended field $\mathbb{R} \cup \{\pm\infty\}$. Second, we present translation algorithms that transform non-negative and non-positive SSP problems into SSP problems that satisfy the assumptions previously considered in the literature [BT91, Ber95]. This enables the use of several well-known techniques for the solution of non-negative and non-positive SSP problems, such as value iteration methods, and methods based on learning and sample path analysis (see [BT91, Ber95] again). The translation algorithms have strongly-polynomial time complexity in the size of the MDP being translated. As the algorithms never increase and often reduce the size of the MDPs, they also perform a beneficial pre-conditioning prior to the application of numerical solution methods.

Finally, we apply the algorithms presented in this paper to the computation of the minimum and maximum probability of reaching a set of target states. The computation of the minimum reachability probability is useful for determining lower bounds for the probability of reaching desirable system configurations, or of accomplishing tasks from given starting points. The computation of the maximum reachability probability is one of the basic problems in probabilistic verification: aside from being of interest in its own right, it is at the basis of the algorithms for the determination of the maximum and minimum probability with which a linear-time temporal logic formula holds over an MDP [CY90, CY95, BdA95]. While the maximum reachability probability can be computed with the algorithms of [CY90], the proposed approach minimizes the size of the numerical problem to be solved.

2 Preliminaries

A *Markov decision process* (MDP) is a generalization of a Markov chain in which nondeterministic choice coexists with probabilistic one. Markov decision processes are closely related to the *probabilistic automata* of [Rab63], to the *concurrent Markov chains* of [Var85], and to the *simple probabilistic automata* of [SL94, Seg95]. To present their definition, given a countable set C we denote by $\mathcal{D}(C)$ the set of probability distributions over C, i.e. the set of functions $f : C \mapsto [0,1]$ such that $\sum_{x \in C} f(x) = 1$. Given a distribution $f \in \mathcal{D}(C)$, we indicate by $\mathit{Support}(f) = \{x \in C \mid f(x) > 0\}$.

An MDP $\mathcal{M} = (S, Acts, A, p)$ consists of the following components:

- a finite set S of states;
- a finite set $Acts$ of actions;
- a function $A : S \mapsto 2^{Acts}$ that associates with each $s \in S$ a finite set $A(s) \subseteq Acts$ of actions available at s;
- a function $p : S \times Acts \mapsto \mathcal{D}(S)$ that associates with each $s, t \in S$ and $a \in A(s)$ the probability $p(s, a)(t)$ of a transition from s to t when action a is selected.

A *path* of the MDP $\mathcal{M}$ is an infinite sequence $\omega : s_0, a_0, s_1, a_1, \ldots$ of alternating states and actions, such that $s_i \in S$, $a_i \in A(s_i)$ and $p(s_i, a_i)(s_{i+1}) > 0$ for all $i \geq 0$. For $i \geq 0$, the sequence is constructed by iterating a two-phase selection process. First, an action $a_i \in A(s_i)$ is selected nondeterministically; second, the successor state s_{i+1} is chosen according to the probability distribution $p(s_i, a)$. Given a path $\omega : s_0, a_0, s_1, a_1, \ldots$ and $k \geq 0$, we denote by $X_k(\omega)$, $Y_k(\omega)$ its k-th state s_k and its k-th action a_k, respectively. Given a state $s \in S$ and an action $a \in A(s)$ for s, we also denote by $dest(s, a) = \{t \in S \mid p(s, a)(t) > 0\}$ the set of possible successors of s when a is selected.

To be able to talk about the probability of system behaviors, we need to specify the criteria with which the actions are chosen. To this end, we use the concept of *policy* [Der70], closely related to the adversaries of [SL94, Seg95] and to the schedulers of [Var85, PZ86]. A policy η is a mapping $\eta : S^+ \mapsto \mathcal{D}(Acts)$, which associates with each finite sequence of states $s_0, s_1, \ldots, s_n \in S^+$ and each $a \in A(s_n)$ the probability $\eta(s_0, \ldots, s_n)(a)$ of choosing a after following the sequence of states $s_0, \ldots, s_n$. We require that $\eta(s_0, \ldots, s_n)(a) > 0$ implies $a \in A(s_n)$: a policy can choose only among the actions that are available at the state where the choice is made. We indicate with Pol the set of all policies. We say that a policy η is *memoryless* if $\eta(s_0, \ldots, s_n)(a) = \eta(s_n)(a)$ for all sequences of states $s_0, \ldots, s_n \in S^+$ and all $a \in A(s)$.

For every state $s \in S$, we denote by Ω_s the set of paths having s as initial state, and we let $\mathcal{B}_s \subseteq 2^{\Omega_s}$ be the σ-algebra of *measurable* subsets of Ω_s, following the classical definition of [KSK66]. Under policy η the probability of following a finite path prefix $s_0 a_0 s_1 a_1 \cdots s_n$ is $\prod_{i=0}^{n-1} p(s_i, a_i)(s_{i+1})\, \eta(s_0 \cdots s_i)(a_i)$. These probabilities for prefixes give rise to a unique probability measure on $\mathcal{B}_s$. We write $\mathrm{Pr}_s^\eta(\mathcal{A})$ to denote the probability of event $\mathcal{A}$ in Ω_s under policy η, and $\mathrm{E}_s^\eta\{f\}$ to denote the expectation of the random function f from state s under policy η.

2.1 The stochastic shortest path problem

An instance $\Pi = (S, Acts, A, p, R, c, g)$ of the *stochastic shortest path* problem consists of an MDP $(S, Acts, A, p)$, together with the additional components R, c and g:

- $R \subseteq S$ is the the set of *destination states;*
- $c : S \times Acts \mapsto \mathbb{R}$ is the *running cost function,* that associates with each state $s \in S \setminus R$ and each action $a \in A(s)$ the cost $c(s, a)$;

- $g : R \mapsto \mathrm{I\!R}$ is the *terminal cost function,* that associates to each $s \in R$ its terminal cost $g(s)$.

We say that an instance of the SSP problem is *non-negative* (resp. *non-positive*) if $c(s,a) \geq 0$ (resp. $c(s,a) \leq 0$) for all $s \in S$ and $a \in A(s)$; note that the sign of g is not relevant for this definition.

The SSP problem consists in determining the minimum cost of reaching R when following a policy that reaches R with probability 1, provided such a policy exists. Precisely, let $T_R(\omega) = \min\{k \mid X_k(\omega) \in R\}$ be the position of first visit of a path in R. For all $s \in S$ we denote by $\mathit{Prp}(s) = \{\eta \in \mathit{Pol} \mid \Pr_s^\eta(T_R < \infty) = 1\}$ the set of policies that lead from s to R with probability 1; these policies are the *proper policies* for s. Given a state $s \in S$, the *cost* v_s^η of a policy η is defined by

$$v_s^\eta = \mathrm{E}_s^\eta \left\{ g(X_{T_R}) + \sum_{k=0}^{T_R - 1} c(X_k, Y_k) \right\} . \tag{1}$$

A policy η is *optimal* if $v_s^\eta = v_s^*$ for all $s \in S \setminus R$. With this notation, the SSP problem consists in:

1. determining the set $Q = \{s \in S \setminus R \mid \mathit{Prp}(s) \neq \emptyset\}$ of states having at least one proper policy;

2. computing the minimum cost $v_s^* = \inf_{\eta \in \mathit{Prp}(s)} v_s^\eta$ of a proper policy at all $s \in Q$.

Usually, the SSP problem is considered to consist only in the second question, and the existence of at least one proper policy for each state is stated as an assumption. However, when the SSP problem is used to compute the minimum or maximum reachability times between an initial state and a set of target states that are part of a reliability of performance specification, we cannot assume that the target set can be reached from the initial state with probability 1. Hence, in Section 2.3 we present an algorithm to solve also this first question. In addition, we will characterize the optimal policies for non-negative and non-positive SSP problems.

SSP problem and reachability time. In a *timed probabilistic system,* the timing behavior of an MDP (S, Acts, A, p) is specified by means of a function $\mathit{time} : S \times \mathit{Acts} \mapsto \mathrm{I\!R}^+$ that associates with each $s \in S$ and $a \in A(s)$ the expected amount of time $\mathit{time}(s,a)$ spent at state s when action a is selected [dA98]. Given a set R of target states, to compute the minimum (resp. maximum) expected time to reach R it suffices to solve an SSP problem having cost functions defined by $c(s,a) = \mathit{time}(s,a)$ (resp. $c(s,a) = -\mathit{time}(s,a)$) and $g(s) = 0$, for all $s \in S$ and $a \in A(a)$. The minimum (resp. maximum) expected time to reach R from $s \in S \setminus R$ is then given by v_s^* (resp. $-v_s^*$).

2.2 End components

The algorithms that we present to solve the classes of SSP problems rely on the notion of *end component* [dA97]. End components are the analogous concept in Markov decision processes of the closed recurrent classes of Markov chains [KSK66]: they represent the set of states and actions that can be repeated infinitely often along a path with non-zero probability. Related sets of states have been used for solving optimization problems on MDPs [CY95]. Given an MDP $\mathcal{M} = (S, \mathit{Acts}, A, p)$, a *sub-MDP* is a pair (C, D), where $C \subseteq S$ is a subset of states and $D : S \mapsto \mathit{Acts}$ is a function that associates to each $s \in S$ a subset $D(s) \subseteq A(s)$ of actions. A sub-MDP (C, D) is an *end component* if the following conditions hold:

- *Closure:* for all $s \in C$, $a \in D(s)$, and $t \in S$, if $p(s,a)(t) > 0$ then $t \in C$.
- *Connectivity:* Let $E = \{(s,t) \in C \times C \bigm| \exists a \in D(s) \,.\, p(s,a)(t) > 0\}$; then, the graph (C, E) is strongly connected.

We say that an end component (C, D) is contained in a sub-MDP (C', D') if

$$\{(s,a) \mid s \in C \wedge a \in D(s)\} \subseteq \{(s,a) \mid s \in C' \wedge a \in D'(s)\} \ .$$

We say that an end component (C, D) is *maximal* in a sub-MDP (C', D') if there is no other end component (C'', D'') contained in (C', D') that properly contains (C, D). We denote by $\mathit{Mec}(C', D')$ the set of maximal end components of (C', D'). It is not difficult to see that, given a sub-MDP (C, D), the set $\mathit{Mec}(C, D)$ can be computed in time polynomial in $|C| + \sum_{s \in C} |D(s)|$ using simple graph algorithms; an algorithm to do so is given in [dA97, §3]. Given a path ω, denote by $\mathit{InfS}(\omega) = \{s \in S \mid \overset{\infty}{\exists}\, k \,.\, X_k(\omega) = s\}$ the set of states visited infinitely often by ω, where $\overset{\infty}{\exists}$ is a shorthand for "there are infinitely many distinct". Also, define $\mathit{InfA}(\omega) : S \mapsto 2^{\mathit{Acts}}$ by $\{a \in A(s) \mid \overset{\infty}{\exists}\, k \,.\, X_k(\omega) = s \wedge Y_k(\omega) = a\}$ for all $s \in S$. The following theorem summarizes the basic property of end components [dA97].

Theorem 1 *For all $s \in S$ and all $\eta \in \mathit{Pol}$, we have*

$$\Pr_s^\eta\Big((\mathit{InfS}(\omega), \mathit{InfA}(\omega)) \textit{is an end component}\Big) = 1 \ .$$

2.3 Computing the set of states having proper policies

As a first step in the solution of the SSP problem, we must compute the set

$$\mathit{Reach}(R) = \Big\{s \in S \mid \exists \eta \in \mathit{Pol} \,.\, \Pr_s^\eta(T_R < \infty) = 1\Big\}$$

consisting of the states having at least one proper policy. This problem can be solved by reducing it to several well-known dynamic programming problems, such as the *maximum average reward* problem [Der70] or the *maximum reachability probability* problem [CY90]. However, these reductions yield algorithms that are based on linear programming, and their time complexity is only weakly polynomial, i.e. it depends on the size of the bit strings encoding the probability values in the input description of the problem. We present here an algorithm that solves the problem in time quadratic in the size of the MDP, and that does not require any numerical computation. The algorithm is originally from [dA97], and is related to an algorithm for solving reachability problems in two-person games presented in [dAHK98]. The algorithm is also reminiscent of an algorithm independently proposed in [Var95]. To present the algorithm, given two subsets $X, Y \subseteq S$ of states we define the predicate $APre(Y, X)$ so that for all $s \in S$,

$$s \models APre(X, Y) \quad \textit{iff} \quad \exists a \in A(s) \,.\, \Big(dest(s,a) \subseteq Y \wedge dest(s,a) \cap X \neq \emptyset \Big) \,.$$

Given a subset R of target states, we compute *Reach*(R) by the following μ-calculus expression:

$$\textit{Reach}(R) = \nu Y \,.\, \mu X \,.\, (APre(Y, X) \vee R) \,, \tag{2}$$

where we have used the slightly improper notation of denoting by R a predicate that holds exactly for the states in R. The algorithm (2) can be understood as follows. Denoting by Y_k the value of the set Y computed at iteration $k \geq 0$, we have initially $Y_0 = S$. At the end of the first iteration, we have $Y_1 = S \setminus C_0$, where C_0 is the subset of states of S that cannot reach R. At the end of the second iteration, we have $Y_2 = Y_1 \setminus C_1$, where C_1 is the set of states that cannot reach R without risking to enter C_0. In general, at the end of iteration $k > 0$, we have $S_k = S_{k-1} \setminus C_{k-1}$, where C_{k-1} consists of the states that cannot reach R without risking to enter $\bigcup_{i=0}^{k-2} C_i$. Given an MDP $\mathcal{M} = (S, A, p)$, define its graph size $|\mathcal{M}|$ by

$$|\mathcal{M}| = \textstyle\sum_{s \in S} \sum_{a \in A(s)} \Big| \textit{Support}(p(s,a)) \Big| \,.$$

The following theorem summarizes the results about this algorithm.

Theorem 2 *Given an MDP $\mathcal{M} = (S, A, p)$ and a set $R \subseteq S$ of target states, relation (2) correctly computes Reach(R) in time quadratic in $|\mathcal{M}|$.*

Once the set *Reach*(R) has been computed, we can replace the original SSP problem $(S, \textit{Acts}, A, p, R, c, g)$ with a new problem $(Q, \textit{Acts}, A', p', R, c', g')$, where $Q = \textit{Reach}(R)$, where p', c', g' are the restrictions of p, c, g to Q, and where for all $s \in Q$ we let $A'(s) = \{a \in A(s) \mid dest(s,a) \subseteq Q\}$. To avoid a change of notation, in the following we denote an instance of the SSP problem again by $(S, \textit{Acts}, A, p, R, c, g)$, but we assume that *Reach*$(R) = S$. This is equivalent to assuming that the above reduction has been made already.

3 Solving Non-Negative SSP Problems

The class of SSP problems that is most closely related to the non-negative class, and for which solution methods have been presented in the literature, is discussed in [BT91, Ber95]. There, it is shown that the SSP problem can be solved under the additional assumption that, for all $s \in S$, there is a *proper* policy that minimizes the total cost (1). An example of SSP problem in which this assumption does not hold is depicted in Figure 1. Clearly, the policy that minimizes (1) is the policy η_1 that always chooses action a at s_3; this policy leads to the expected cost $v_{s_1}^{\eta_1} = 1$. However, this policy is not proper, and it is easy to see that for every proper policy η it is $v_{s_1}^{\eta} = 3$.

To understand why the iterative approaches such as value iteration cannot be applied immediately to this problem, let $n = |S \setminus R|$, and denote with $\boldsymbol{v} = [v_s]_{s \in S \setminus R} \in \mathbb{R}^n$ a vector

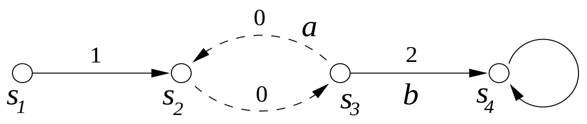

Figure 1: An instance of SSP problem. The target set is $R = \{s_4\}$, and the terminal cost is $g(s_4) = 0$. States are represented as nodes of a graph, and actions as edges. We have indicated only the actions a and b corresponding to state s_3, where $A(s_3) = \{a, b\}$. In this example, all actions (including a and b) are deterministic, i.e. they lead to only one destination state. The actions are labeled with their cost c. The two actions having cost 0 have been indicated with dashed lines. A larger instance of SSP problem is presented in Figure 3.

of real numbers indexed by the states of $S \setminus R$. Define the Bellman operator $L : \mathbb{R}^n \mapsto \mathbb{R}^n$ on the space of $\boldsymbol{v}$ by

$$[L(\boldsymbol{v})]_s = \min_{a \in A(s)} \left[c(s,a) + \sum_{t \in S \setminus U} p(s,a)(t)\, v_t + \sum_{t \in R} p(s,a)(t)\, g(t) \right] \qquad s \in S \setminus R \,, \tag{3}$$

where $[L(\boldsymbol{v})]_s$ denotes the s-components of vector $L(\boldsymbol{v})$. Given an initial vector $\boldsymbol{v}^0$, the value iteration method computes the sequence of vectors $\boldsymbol{v}^0, \boldsymbol{v}^1, \boldsymbol{v}^2, \ldots$ by $\boldsymbol{v}^{k+1} = L(\boldsymbol{v}^k)$, for $k \geq 0$, and returns as answer $\lim_{k \to \infty} \boldsymbol{v}^k$, provided the limit exists. The initial vector $\boldsymbol{v}^0$ represents an initial (often arbitrary) estimate for the minimum expected reachability cost; each iteration of the Bellman operator L is aimed at improving the estimate. Clearly, the answer returned by the value iteration procedure is a fixpoint of L. However, in non-negative SSP problems the Bellman operator L may admit more than one fixpoint: for example, in the SSP problem of Figure 1, for $x \geq 0$ all vectors

$$\boldsymbol{v}(x) = [v_1, v_2, v_3] = [3, 2, 2] - x[1, 1, 1] \tag{4}$$

satisfy $\boldsymbol{v} = L(\boldsymbol{v})$. If L admits more than one fixpoint, the sequence $\boldsymbol{v}^0, \boldsymbol{v}^1, \boldsymbol{v}^2, \ldots$ can converge to any one of them, depending on the value of the initial estimate $\boldsymbol{v}^0$. In the example of Figure 1, starting from the initial vector $[0, 0, 0]$, the value iteration method converges to the fixpoint $[1, 0, 0]$. However, we will prove that the solution of the SSP problem corresponds to the largest fixpoint, which in this case is $[3, 2, 2]$. The fact that the Bellman operator does not necessarily admit a unique fixpoint in non-negative SSP problems not only prevents a direct application of value iteration methods, but also blocks the line of analysis of [BT91] for the solution based on linear programming.

We present two approaches to the solution of non-negative SSP problems. The first approach is based on the observation that the difficulties in solving non-negative SSP problems stem from the presence in the SSP problem of end components consisting of state-action pairs having 0 cost. If we remove these components, we obtain an equivalent problem whose Bellman operator has a unique fixpoint; the problem can then be solved using any of several methods that have been developed for SSP problems, including linear programming and value iteration. This approach has two advantages. First, it enables to exploit in the solution of the SSP problem many numerical techniques that have been devised to handle large-sized problems. Second, the algorithm that removes the end components often achieves a reduction of the size of the problem.

The second approach consists in reducing the SSP problem directly to linear programming: since the solution of the linear programming problem corresponds to the greatest fixpoint of the Bellman operator, as we will show, it also corresponds to the solution of the SSP problem. The correctness proof of this second approach relies on an analysis of the first approach.

3.1 Eliminating 0-cost end components

A *0-cost* end component is an end component (C, D) such that $c(s,a) = 0$ for all $s \in C$ and all $a \in D(s)$. As we will show (see Theorem 3), the lack of uniqueness of the fixpoint is due to the presence of 0-cost end components in the MDP. In a 0-cost component (C, D), by selecting at each $s \in C$ the actions in $D(s)$ uniformly at random, we can go from any state of C to any other state of C with probability 1 while incurring cost 0. Hence, the states of a 0-cost end component are equivalent from the point of view of the minimum cost to the target. The following algorithm exploits this fact to eliminate the 0-cost end components of an MDP by replacing them with single states. The algorithm opens the way to the use of iterative methods based on the Bellman operator for the solution of the non-negative SSP problem.

Algorithm 1 (eliminating 0-cost end components)
Input: SSP problem $\Pi = (S, Acts, A, p, R, c, g)$.

Output: SSP problem $\widehat{\Pi} = ElimEC(\Pi) = (\widehat{S}, Acts, \widehat{A}, \widehat{p}, R, \widehat{c}, \widehat{g})$.

Method: For each $s \in S \setminus R$, let $D(s) = \{a \in A(s) \mid c(s,a) = 0\}$, and let $\{(B_1, D_1), \ldots, (B_n, D_n)\} = Mec(S \setminus R, D)$ be the set of 0-cost maximal end components that lie outside R. Define $\widehat{S} = S \cup \{\widehat{s}_1, \ldots, \widehat{s}_n\} \setminus \bigcup_{i=1}^n B_i$, where $\widehat{s}_1, \ldots, \widehat{s}_n$ are new states. The action sets associated with the states are defined by:

$$\begin{aligned} s \in S \setminus \textstyle\bigcup_{i=1}^n B_i : \qquad & \widehat{A}(s) = \{\langle s, a\rangle \mid a \in A(s)\} \\ 1 \leq i \leq n : \qquad & \widehat{A}(\widehat{s}_i) = \left\{ \langle s, a\rangle \;\middle|\; s \in B_i \wedge a \in A(s) \setminus D_i(s) \right\} . \end{aligned}$$

For $s \in \widehat{S}$, $t \in S \setminus \bigcup_{i=1}^n B_i$ and $\langle u, a\rangle \in \widehat{A}(s)$, the transition probabilities are defined by $\widehat{p}(s, \langle u, a\rangle)(t) = p(u,a)(t)$ and $\widehat{p}(s, \langle u, a\rangle)(\widehat{s}_i) = \sum_{t \in B_i} p(u,a)(t)$. For $s \in \widehat{S}$ and $\langle u, a\rangle \in \widehat{A}(s)$ we let $\widehat{c}(s, \langle u, a\rangle) = c(u,a)$; for $s \in R$ we let $\widehat{g}(s) = g(s)$. ∎

The algorithm replaces each 0-cost end component (B_i, D_i) with a single new state $\widehat{s}_i$, for $1 \leq i \leq n$. The actions associated with $\widehat{s}_i$ consist in all the pairs $\langle t, a\rangle$ such that $s \in C_i$ and $a \in A(s)$ is an action not belonging to the end component. Intuitively, taking action $\langle s, a\rangle$ at $\widehat{s}_i$ corresponds to taking action a from s, possibly leaving C_i. The transition probabilities and costs of the corresponding actions are unchanged, except that the probability of a transition to $\widehat{s}_i$ is equal to the probability of a transition into C_i in the original system, for $1 \leq i \leq n$. The result of applying Algorithm 1 to the instance of SSP depicted in Figure 1 is illustrated in Figure 2. The (maximal) end component

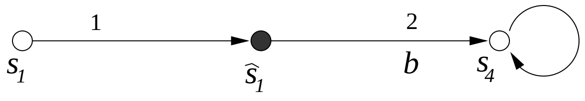

Figure 2: Result of applying Algorithm 1 to the instance of SSP problem depicted in Figure 2. The new state $\widehat{s}_1$ introduced by the algorithm is drawn as a filled circle.

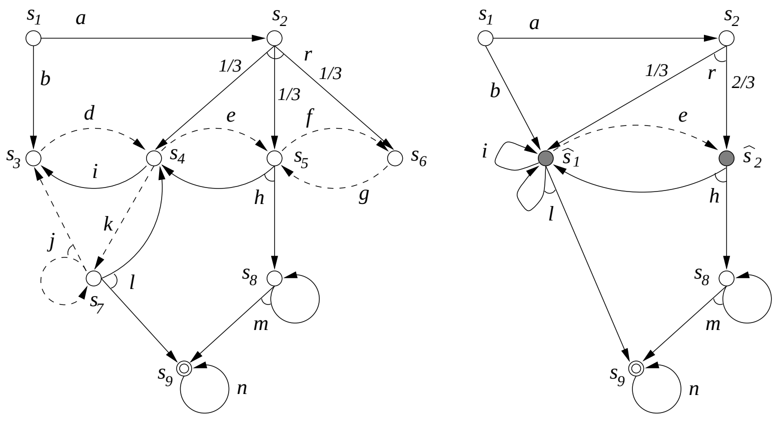

Figure 3: An instance of SSP problem (left), and the result of applying Algorithm 1 to it (right). Here, not all actions are deterministic, and we depict actions that can lead to more than one destination by "bundles" of edges. To simplify the diagrams, we have indicated only the transition probabilities corresponding to action r, and we have omitted all costs. The actions that have cost 0 have been represented by dashed edges. The target set is $R = \{s_9\}$. The new states $\widehat{s}_1$ and $\widehat{s}_2$ that have been introduced to replace the zero-cost end components are indicated by filled circles.

formed by states s_2, s_3 together with the 0-cost actions has been replaced by the single state $\hat{s}_1$. Figure 3 depicts another example of application of Algorithm 1. The algorithm computes the 0-cost end components $(B_1, D_1), (B_2, D_2)$, where the first end component is given by $B_1 = \{s_3, s_4, s_7\}$ and $D_1(s_3) = \{d\}$, $D_1(s_4) = \{k\}$, $D_1(s_7) = \{j\}$, and the second one by $B_2 = \{s_5, s_6\}$ and $D_2(s_5) = \{f\}$, $D_2(s_6) = \{g\}$. The algorithm replaces these end components with the two new states $\widehat{s}_1$ and $\widehat{s}_2$. This example illustrates the potential reduction of the state-space of the system.

Once the 0-cost end components have been eliminated, the next lemma shows that the reduced problem satisfies the following two assumptions:

SSP-1 For all $s \in S$, we have $Prp(s) \neq \emptyset$.

SSP-2 For all $s \in S$ and $\eta \notin Prp(s)$, we have $v_s^\eta = \infty$.

Lemma 1 *Consider an instance* Π *of non-negative SSP problem such that there is at least one proper policy for each state, and let* $\widehat{\Pi} = ElimEC(\Pi)$*. Then,* $\widehat{\Pi}$ *satisfies assumptions SSP-1 and SSP-2.*

Proof. By hypothesis (or more accurately, by the algorithm presented in Section 2.3), $\widehat{\Pi}$ satisfies SSP-1. By Theorem 1, the set of states and actions that are repeated infinitely often along a path is an end component. Hence, if all 0-cost end components have been eliminated, with probability 1 a path that does not reach R has infinite cost, showing that $\widehat{\Pi}$ satisfies SSP-2. ∎

The class of SSP problems that satisfies assumptions SSP-1 and SSP-2 has been studied in depth in the literature. In particular, it is known that the Bellman operator admits a unique fixpoint for this class of problems, and that there exist optimal policies that are memoryless [BT91]. Moreover, such problems can be solved using value-iteration and policy-iteration methods, which converge to the solution [BT91]. Other refined iterative methods for the solutions of this class of problems are presented in [BT96]. As hinted by Lemma 1, the uniqueness of the fixpoint of the Bellman operator is related to the presence of 0-cost end components.

Theorem 3 *Given a non-negative instance $(S, Acts, A, p, R, c, g)$ of SSP problem, the Bellman operator L admits a unique fixpoint iff there is no 0-cost end component (C, D) with $C \subseteq S \setminus R$.*

Proof. In one direction, assume that a non-negative instance of SSP problem does not contain any 0-cost end component. Reasoning as for Lemma 1, we have that assumption SSP-2 holds. If assumption SSP-1 also holds, then the uniqueness of the fixpoint follows from [BT91]. If assumption SSP-1 does not hold, then assumption SSP-2 ensures that the fixpoint of the Bellman operator diverges to $+\infty$ on the states where there is no proper policy. This, together with the analysis of [BT91] for the states where there are proper policies, ensures again the uniqueness of the fixpoint. Conversely, if there is a 0-cost end component in a non-negative SSP problem, then we can obtain multiple fixpoints of the Bellman operator by selecting one such end component, and by setting the value of the fixpoint there to any negative value, as done in (4). ∎

The following theorem relates the solutions of the SSP problems Π and $\widehat{\Pi}$, and it enables the (trivial) derivation of a solution for Π from a solution for $\widehat{\Pi}$.

Theorem 4 *Consider an instance Π of non-negative SSP such that there is at least one proper policy for each state, and let $\widehat{\Pi} = ElimEC(\Pi)$. Let also $B_1, \ldots, B_n$ be the 0-cost end components that are replaced by states $\widehat{s}_1, \ldots, \widehat{s}_n$. Denoting by $\boldsymbol{v}^*$ (resp. $\widehat{\boldsymbol{v}}^*$) the solution of the SSP problems on Π (resp. $\widehat{\Pi}$), we have $v^*_s = \widehat{v}^*_s$ for $s \in S \setminus \bigcup_{i=1}^n B_i$, and $v^*_s = \widehat{v}^*_{\widehat{s}_i}$ for $s \in B_i$, $1 \leq i \leq n$.*

Even though it might appear intuitively plausible that eliminating the 0-cost end components should not modify the solution of the SSP problem, the proof of the above theorem is somewhat involved; it can be found in [dA97]. The same analysis also leads to the following result.

Corollary 1 *Non-negative SSP problems admit memoryless optimal policies.*

3.2 Linear programming

The second approach is given by the following theorem.

Theorem 5 *Consider an instance* Π *of non-negative SSP such that there is at least one proper policy for each state. Then, the solution* $\boldsymbol{v}^*$ *of the SSP problem is the largest fixpoint of operator* L *defined in (3). Moreover, the following linear programming problem has* $\boldsymbol{v}^*$ *as unique solution:*

$$\textit{Maximize} \sum_{s \in S \setminus R} v_s \qquad \textit{subject to} \qquad v_s \leq c(s,a) + \sum_{t \in S \setminus R} p(s,a)(t)\, v_t + \sum_{t \in R} p(s,a)(t)\, g(t)$$

for all $s \in S \setminus R$ *and* $a \in A(s)$.

Proof. The theorem is proved by showing first that every fixpoint of the Bellman operator (3) is no greater (componentwise) than the solution of the SSP problem. Next, we use the relationship between Π and $\widehat{\Pi} = \mathit{ElimEC}(\Pi)$ to show that one of the fixpoints is equal to the solution of the SSP problem; this implies that the solution of the SSP problem is the largest fixpoint. Finally, it can be shown that the linear programming problem converges to the largest fixpoint, and thus to the solution of the SSP problem. The details can be found in [dA97]. ∎

4 Solving Non-Positive SSP Problems

Consider an instance $\Pi = (S, \mathit{Acts}, A, p, R, c, g)$ of non-positive SSP problem, and assume that $S = \mathit{Reach}(R)$, i.e. that for every state there is a proper policy. Unlike in the non-negative case, it is possible that $v_s^* = -\infty$ for some $s \in S \setminus R$, and the first step towards the solution of non-negative SSP problems consists in determining the set of states from which the minimum cost diverges to $-\infty$. This can be done with the following algorithm.

Algorithm 2

Input: A non-positive SSP problem $\Pi = (S, \mathit{Acts}, A, p, Rc, g)$, with $\mathit{Reach}(R) = S$.

Output: The subset $\mathit{Diverge}(\Pi) = \{s \mid v_s^* = -\infty\}$.

Method: Let $\mathcal{L} := \Big\{(C,D) \in \mathit{Mec}(S \setminus R, A) \;\Big|\; \exists s \in C \,.\, \exists a \in D(s) \,.\, c(s,a) < 0\Big\}$ be the set of end components outside R that have at least one strictly negative state-action pair, and let $C = \bigcup_{(C,D)\in\mathcal{L}} C$ be the union of their states.

Let $C_\infty = \mu X \,.\, \Big(\neg R \wedge (\mathit{APre}(S,X) \vee C)\Big)$ be the set of states that can reach C without entering R.

Return: C_∞. ∎

Theorem 6 *For an instance* Π *of non-positive SSP such that* $S = \mathit{Reach}(R)$, *we have that* $v_s^* = -\infty$ *iff* $s \in \mathit{Diverge}(\Pi)$.

Proof. From a state $s \in \mathit{Diverge}(\Pi)$, we can reach with positive probability an end component in $\mathcal{L}$. Once there, we can stay in the end component arbitrarily long, accumulating an arbitrarily large amount of negative cost, before proceeding to the target. Hence, we have $v^*_s = -\infty$. The details can be found in [dA97]. The proof of the converse, i.e., that if $s \notin \mathit{Diverge}(\Pi)$ then $v^*_s > -\infty$, will be given in Section 4.1. ∎

Once the set $\mathit{Diverge}(\Pi)$ has been computed, it remains to compute v^*_s for $s \in S \setminus (R \cup \mathit{Diverge}(\Pi))$. To this end, we first reduce the SSP problem by eliminating the states in $\mathit{Diverge}(\Pi)$. We define a new instance of SSP $\widetilde{\Pi} = \mathit{Converge}(\Pi) = (\widetilde{S}, \mathit{Acts}, \widetilde{A}, \widetilde{p}, R, \widetilde{c}, \widetilde{g})$, where $\widetilde{S} = S \setminus \mathit{Diverge}(\Pi)$, where $\widetilde{p}$, $\widetilde{c}$, $\widetilde{g}$ are the restrictions of p, c, g to $\widetilde{S}$, and where for all $s \in \widetilde{S}$ we let $\widetilde{A}(s) = A(s)$. The reduced non-positive SSP problem can then be solved in three ways: by eliminating the 0-cost end components, by linear programming, and by value iteration.

4.1 Eliminating 0-cost components

The first method for solving the reduced problem consists in eliminating the 0-cost end components using Algorithm 1 to compute $\widehat{\Pi} = \mathit{ElimEC}(\widetilde{\Pi})$. The following theorem asserts that $\widehat{\Pi}$ satisfies conditions SSP-1 and SSP-2: hence, the SSP instance $\widehat{\Pi}$ can be solved with the methods presented in [BT91, BT96, Ber95].

Theorem 7 *The non-positive SSP instance $\widehat{\Pi} = \mathit{ElimEC}(\widetilde{\Pi})$ satisfies conditions SSP-1 and SSP-2. Moreover, let $B_1, \ldots, B_n$ be the 0-cost end components that are replaced by states $\widehat{s}_1, \ldots, \widehat{s}_n$. Denoting by $\widetilde{\boldsymbol{v}}^*$ (resp. $\widehat{\boldsymbol{v}}^*$) the solution of the SSP problems on $\widetilde{\Pi}$ (resp. $\widehat{\Pi}$), we have $\widetilde{v}^*_s = \widehat{v}^*_s$ for $s \in \widetilde{S} \setminus \bigcup_{i=1}^n B_i$, and $\widetilde{v}^*_s = \widehat{v}^*_{\widehat{s}_i}$ for $s \in B_i$, $1 \le i \le n$.*

Proof. Since the costs are non-positive, the cost from a state never diverges to $+\infty$. Hence, by Theorem 1, a non-positive instance satisfies condition SSP-2 iff there are no end components entirely outside of the target R. To see that this condition holds for $\widehat{\Pi}$, note that the end components containing some negative cost have been eliminated by Algorithm 2, while those consisting entirely of 0-cost state-action pairs have been eliminated by Algorithm 1. The second part of the result is proved in an analogous way to Theorem 4, and the proof can be found in [dA97]. ∎

Theorem 7 also leads to the second part of Theorem 6. If $s \notin \mathit{Diverge}(\Pi)$, then $s \in \widetilde{S}$. The fact that assumptions SSP 1 and SSP 2 hold for $\widehat{\Pi}$, together with the results of [BT91], ensures then that $v^*_s > -\infty$.

Theorem 8 *An instance of non-negative SSP problem Π admits memoryless optimal (proper) policies iff $\mathit{Diverge}(\Pi) = \emptyset$. In any case, there is always a (possibly non memoryless) optimal proper policy.*

Proof. To see that if $\mathit{Diverge}(\Pi) \neq \emptyset$, then there are no memoryless optimal policies, refer to Algorithm 2. Since under a memoryless policy the MDP behaves like a Markov chain, under a memoryless proper policy each path stays for a finite expected amount of time in the end components in $\mathcal{L}$ before reaching R, so that $v^\eta_s > -\infty$ for all $s \in S \setminus R$. On the other hand, there is a (non-memoryless) policy such that, once we reach an end component in $\mathcal{L}$, we stay for infinite expected time in the end component (accumulating an infinite expected cost) before reaching the target with probability 1. The proof that if *Diverge*$(\Pi) = \emptyset$ there are memoryless optimal policies can be found in [dA97]. ∎

4.2 Linear programming

Reasoning as in the proof of Theorem 5, it is possible to show that the solution of the SSP problem corresponds to the largest fixpoint of the Bellman operator. The solution can thus be computed by linear programming.

Theorem 9 *Consider an instance Π of non-positive SSP problem such that $\Pi = Converge(\Pi)$, and such that there is at least one proper policy for each state. Then, the solution $\boldsymbol{v}^*$ of the SSP problem is the largest fixpoint of operator L of (3). Moreover, the following linear programming problem has $\boldsymbol{v}^*$ as unique solution:*

$$\textit{Maximize} \sum_{s \in S \setminus R} v_s \qquad \textit{subject to} \qquad v_s \leq c(s,a) + \sum_{t \in S \setminus R} p(s,a)(t)\, v_t + \sum_{t \in R} p(s,a)(t)\, g(t)$$

for all $s \in S \setminus R$ and all $a \in A(s)$.

4.3 Value iteration

The third way to solve the reduced problem is by value iteration. Convergence to the solution of the SSP problem can be ensured simply by using an initial estimate $\boldsymbol{v}^0$ that is identically 0.

Theorem 10 *Consider an instance Π of non-positive SSP such that $\Pi = Converge(\Pi)$, and such that there is at least one proper policy for each state. Then, the solution of the SSP problem is given by $\lim_{k \to \infty} L^k(0)$, where 0 is the vector all whose entries are 0.*

Proof. The theorem follows from the fact that, in a non-positive SSP problem, all fixpoints of the Bellman operator are componentwise smaller or equal to 0. Since the solution computed by Theorem 10 is the largest such fixpoint, by Theorem 9 it is also the solution of the SSP problem. ∎

5 Maximum and Minimum Reachability Probabilities

An instance $\Lambda = (S, Acts, A, p, T)$ of the *maximum* or *minimum reachability* problems consists of an MDP $\Pi = (S, Acts, A, p)$ together with a destination set T. The maximum and minimum reachability probability problems consists in determining, for all $s \in S$, the values

$$u_s^+ = \sup_{\eta \in Pol} \Pr_s^\eta(\exists k \,.\, X_k \in T) \qquad u_s^- = \inf_{\eta \in Pol} \Pr_s^\eta(\exists k \,.\, X_k \in T)\ .$$

Let $Z \subseteq S$ be the subset of states that cannot reach T (so that $u_s^+ = 0$ for $s \in Z$). From [CY90], we know that the maximum reachability probability can be solved using a linear programming problem on the set of variables $\{u_s \mid s \in S \setminus (T \cup Z)\}$. Here, we show how our results on the SSP problem can be used to improve the efficiency of that solution, as well as to solve the minimum reachability probability problem.

Maximum reachability probability. To reduce the maximum reachability probability problem to the SSP problem, we construct from the instance Λ an SSP instance $\Pi = \mathit{Ssp}^+(\Lambda) = (S, \mathit{Acts}, A, p, R, c, g)$, where $R := \mathit{Reach}(T) \cup Z$, the cost c is identically 0, and the terminal cost is defined by $g(s) = -1$ for $s \in \mathit{Reach}(T)$, and $g(s) = 0$ for $s \in Z$. Note that Λ is both a non-negative and a non-positive instance of SSP problem. The following theorem relates the two problems.

Theorem 11 *If $\Pi = \mathit{Ssp}^+(\Lambda)$, then $u_s^+ = -v_s^*$ for all $s \in S \setminus R$, where u_s^+ is computed on Λ and v_s^* on Π.*

Proof. Since every state of $S \setminus R$ can reach R, we have that $\mathit{Reach}(R) = S$, so that every state of S has a proper policy. From a memoryless optimal policy η_s for the SSP problem, we can construct a policy η_r that coincides with η_s on $S \setminus R$ such that $-v_s^{\eta_s} = u_s^{\eta_r}$, yielding $-v_s^* \leq u_s^+$ for all $s \in S \setminus T$. In the other direction, consider a memoryless policy η_r optimal for reachability (we know from [dA97] that such a policy exists). We have $u_s^{\eta_r} = -v_s^{\eta_r}$ for all $s \in S \setminus T$. Moreover, η_r is proper, since every state of $S \setminus R$ can reach T with positive probability. This yields the reverse inequality $-v_s^* \geq u_s^+$ for all $s \in S \setminus R$, and hence the result. ∎

Note that we have used algorithm (2) to reduce the size of the set of states on which the maximum reachability probability must be determined, from $S \setminus (T \cup Z)$ to $S \setminus (\mathit{Reach}(T) \cup Z)$. Theorem 11 opens the way to the application of Algorithm 1 for the solution of maximum reachability probability problems. Since the running cost c is identically 0, the algorithm eliminates all end components of the MDP that lie completely outside of $\mathit{Reach}(T) \cup Z$, achieving a further potential reduction in the size of the problem.

Minimum Reachability Probability. Let $\{(C_1, D_1), \ldots, (C_n, D_n)\} = \mathit{Mec}(S \setminus T, A)$ be the set of maximal end components lying outside T, and let $C = \bigcup_{i=1}^n C_i$ be the union of their states. Clearly, from $Z \cup C$ the minimum probability of reaching T is 0. Moreover, the MDP does not have any end component completely contained in $S \setminus (T \cup Z \cup C)$. From the instance $\Lambda = (S, \mathit{Acts}, A, p, T)$ we construct an SSP instance $\Pi = \mathit{Ssp}^-(\Lambda) = (S, \mathit{Acts}, A, p, R, c, g)$, where $R := T \cup Z \cup C$, the cost c is identically 0, and the terminal cost is defined by $g(s) = 0$ for $s \in Z \cup C$, and $g(s) = 1$ for $s \in T$. The following theorem relates the two problems, and it enables the computation of the minimum probability of reaching the target.

Theorem 12 *If $\Pi = \mathit{Ssp}^-(\Lambda)$, then $u_s^- = v_s^*$ for all $s \in S$, where u_s^- is computed on Λ and v_s^* is computed on Π.*

Proof. The proof of the theorem follows from the fact that all policies of Π are proper, and from the observation that from a policy η_s of Π, we can easily obtain a policy η_r for Λ such that $u_s^{\eta_r} = v_s^{\eta_s}$ for all $s \in S$, and vice versa. ∎

In this case, Algorithm 1 cannot be used to reduce the size of the problem, since there are no end components in $S \setminus R$. The reduction has been effected in a more direct way by adding the set C to the set of target states of the SSP problem.

Optimal policies. The maximum and minimum reachability problems admit memoryless optimal policies. This result is proved in [dA97].

References

[BdA95] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Found. of Software Tech. and Theor. Comp. Sci.*, volume 1026 of *Lect. Notes in Comp. Sci.*, pages 499–513. Springer-Verlag, 1995.

[Bel57] R.E. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[Ber95] D.P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 1995. Volumes I and II.

[BT91] D.P. Bertsekas and J.N. Tsitsiklis. An analysis of stochastic shortest path problems. *Math. of Op. Res.*, 16(3):580–595, 1991.

[BT96] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

[CY90] C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. In *Proc. 17th Int. Colloq. Aut. Lang. Prog.*, volume 443 of *Lect. Notes in Comp. Sci.*, pages 336–349. Springer-Verlag, 1990.

[CY95] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.

[dA97] L. de Alfaro. *Formal Verification of Probabilistic Systems.* PhD thesis, Stanford University, 1997. Technical Report STAN-CS-TR-98-1601.

[dA98] L. de Alfaro. Stochastic transition systems. In *CONCUR'98: Concurrency Theory. 9th Int. Conf.*, Lect. Notes in Comp. Sci. Springer-Verlag, 1998.

[dAHK98] L. de Alfaro, T.A. Henzinger, and O. Kupferman. Concurrent reachability games. In *Proc. 39th IEEE Symp. Found. of Comp. Sci.*, 1998.

[Der70] C. Derman. *Finite State Markovian Decision Processes.* Academic Press, 1970. cut?

[EZ62] J.H. Eaton and L.A. Zadeh. Optimal pursuit strategies in discrete-state probabilistic systems. *J. of Basic Engineering*, pages 23–29, 1962.

[Han94] H. Hansson. *Time and Probabilities in Formal Design of Distributed Systems.* Real-Time Safety Critical Systems Series. Elsevier, 1994.

[JL91] B. Jonsson and K.G. Larsen. Specification and refinement of probabilistic processes. In *Proc. 6th IEEE Symp. Logic in Comp. Sci.*, pages 266–277, 1991.

[KB98] C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing,* vol. 11, May 1998.

[KNPS99] M.Z. Kwiatkowska, G. Norman, D. Parker, and R. Segala. Symbolic model checking of concurrent probabilistic systems using MTBDDS and simplex. Technical Report CSR-99-1, University of Birmingham, 1999.

[KSK66] J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains.* D. Van Nostrand Company, 1966.

[LS89] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing (preliminary report). In *Proc. 16th ACM Symp. Princ. of Prog. Lang.*, pages 344–352, 1989.

[PZ86] A. Pnueli and L. Zuck. Probabilistic verification by tableaux. In *Proc. First IEEE Symp. Logic in Comp. Sci.*, pages 322–331, 1986.

[Rab63] M.O. Rabin. Probabilistic automata. *Information and Computation*, 6:230–245, 1963.

[Seg95] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, MIT, 1995. Technical Report MIT/LCS/TR-676.

[SL94] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. In *CONCUR'94: Concurrency Theory. 5th Int. Conf.*, volume 836 of *Lect. Notes in Comp. Sci.*, pages 481–496. Springer-Verlag, 1994.

[Var85] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state systems. In *Proc. 26th IEEE Symp. Found. of Comp. Sci.*, pages 327–338, 1985.

[Var95] M. Vardi. Infinite games against nature. Unpublished manuscript, 1995.

[WSS94] S.-H. Wu, S.A. Smolka, and E.W. Stark. Composition and behaviors of probabilistic I/O automata. In Springer-Verlag, editor, *CONCUR'94: Concurrency Theory. 5th Int. Conf.*, Lect. Notes in Comp. Sci., pages 511–528, 1994.