Université de Mons

Faculty of Science
Computer Science

Multi-objective Synthesis in Markov Decision Processes

Master thesis submitted by
Florent Delgrange
in fulfilment of the requirements for the
Computer Science Master degree

Directors :
Véronique Bruyère
Mickael Randour

Academic year 2017-2018

Abstract

Markov decision processes (MDPs, for short) are systems widely used to model both stochastic and nondeterministic situations, requiring to optimise costs to achieve a goal. In this thesis, we begin by presenting algorithms that synthesise strategies allowing to reach a set of target states in an MDP while ensuring a good expected cost-to-target, or ensuring high probability to reach this set with a cost bounded. Then, we introduce probabilistic-rewards computation tree logic (PRCTL, for short), a probabilistic branching time logic, allowing to express reachability, expectation, cost-bounded reachability, persistence, and repeated reachability properties. We present the related algorithms to build strategies to verify such properties in MDPs. Afterwards, we introduce a way to build strategies ensuring a worst case guarantee (in terms of cost) to reach a set of target states, and we approach then three multi-objective decision problems. The first consists in deciding the existence of a strategy ensuring good expectation under an acceptable worst case. The second is about deciding the existence of a strategy ensuring simultaneously multiple reachability properties in an MDP. The third is finally about deciding the existence of a strategy ensuring simultaneously multiple cost-bounded reachability properties in a multi-dimensional MDP, i.e., an MDP with multiple cost dimensions. For these last two problems, an analysis of trade-offs between satisfying strategies is preferred, and we present the notion of Pareto curve to deal with this approach.

Finally, we present the tool Storm, a modern model checker able to deal with multi-objective problems.

Keywords

Markov decision processes, model checking, multi-objective, stochastic systems, shortest path

Acknowledgements

Firstly, I would especially like to thank my directors and guides, Véronique Bruyère and Mickael Randour for their valuable advices and support.
Then, I would like to thank everyone who accompanied me during my studies at the University of Mons, my Professors, my family, and my classmates and friends, especially Arman Davidyan, Semih Locqueneux, Jeremy Gheysen, Martin Lempereur, Dimitri Waelkens, Clément Tamines, and Aline Goeminne, with whom I shared and overcame all these trials and events leading to the graduation.
Finally, I particularly thank Carolane Gerbehaye for her immeasurable support.

Contents

Introduction

Markov decision processes are *stochastic systems*, allowing to model situations requiring both decision making and probabilistic behaviour. Such systems have many applications, including economy, communication, biology, security, machine learning, and many others. These systems consist of states, actions, and transition probabilities. On entering a state, an action has to be nondeterministically chosen. Once this action is chosen, the next state is determined following a probability distribution associated with the current state and the action chosen. *Strategies* allow to solve nondeterminism, prescribing each action to choose. In this thesis, we present many problems for Markov decision processes and describe how to build strategies satisfying these problems. Choosing an action may induce *multiple* costs, and a natural questioning is about how to minimise such costs to reach one or more set of target states starting from a given state in the system.

In Chapter 1, we formally define some concepts of probability theory, what are *Markov chains*, essential to understand the probabilistic behaviour of Markov decision processes, and we present fundamental concepts and reachability problems in these stochastic systems. Next, we formally define what are Markov decision processes, strategies, and we present two variants of the *stochastic shortest path problem*. Indeed, in a first step, we assume actions having a unique *weight dimension*. As Markov decision processes model stochastic situations, a strategy cannot ensure to reach some target states with a fixed cost. This problem cannot therefore be considered as a classical shortest path problem in a graph. The two problems we present consist in deciding the existence of a strategy offering good *expected cost-to-target*, or high *cost-bounded reachability probability*. Instances of these problems are, e.g., respectively deciding if there exists a strategy for a node in a wireless sensor network allowing to deliver a message to another node with an expected time of t_1 time units, or deliver this message within t_2 time units with high probability.

Then, in Chapter 2, we present some algorithms, allowing to build strategies in order to *model-check* Markov decision processes. *Model checking* is about verifying that properties hold in a system. In order to do that in Markov decision processes, we present a probabilistic branching time logic, called *probabilistic computation tree logic*, which is appropriate to express a large class of properties such as reachability, persistence, and repeated reachability. Instances of such properties are, e.g., reaching some target states avoiding some "bad" states in a limit number of step or with a cost bounded, with high probability, or always visiting "safe" states with high probability. We additionally extend this logic to support costs, allowing to describe expected cost-to-target and cost-bounded reachability properties.

Afterwards, in Chapter 3, considering that actions have multiple *weight dimensions*, we present some *multi-objectives* problems, consisting in deciding the existence of strategies satisfying *simultaneously* multiple sub-problems, possibly conflicting, requiring the analysis of *trade-offs*. For instance, it could be interesting for a node in a wireless sensor network to deliver a message within t time units with high probability, while having high probability of low energy consumption. We also present how to build strategies for other type of multi-objective problems, such as ensuring good expected cost-to-target to reach the targets under an acceptable worst case. For instance, it could be interesting for the node in the wireless sensor network to have a strategy that *guarantees* to deliver a message within t time units while offering good expected time to deliver this message.

Finally, in Chapter 4, we present a modern model-checker, called Storm [DJKV17]. In addition to be able to classically model-check any Markov decision process with properties expressed in a language based on probabilistic computation tree logic, this model checker has the particularity of supporting *multi-objective model checking*.

Chapter 1

Preliminaries

This chapter refers to (and summarises) my Master project [Del17]. Concepts, theorems and definitions are essentially inspired by the chapter ten of the book "Principles of model checking" [BK08], the chapter "Model checking probabilistic systems" of the course named "Formal verification of computer systems" of Mickael Randour [Ran18] as well as the article "Variations on the stochastic shortest path problem" [RRS15] by Mickael Randour et al.

Before studying different problems in *Markov decision processes* and defining optimal *strategies* referring to such problems, we introduce some fundamental concepts. We are interested in the cost of *paths* of Markov decision processes as well as the probability to reach some states in these models. To do that, we need to define a proper *probability measure* on *events* formed with these paths. So, first of all, we need to define what are *Markov chains*. Indeed, these stochastic models are essential to define such a probability measure.

1.1 Markov chains

Markov chains are models describing evolutions of stochastic systems. The particularity of such a system is that it evolves from any state to one of its successors following a probability distribution. Thus, going from a state to another in a Markov chain only depends on the current state of the system.

Definition 1.1 (**Discrete-time Markov chain**)**.** A *(weighted) discrete-time Markov chain* (MC, for short) is a stochastic model defined by a tuple $\mathcal{M} = (S, \Delta, w, AP, L)$ where

- S is a countable set of states,
- $\Delta : S \times S \to [0, 1] \cap \mathbb{Q}$ is a *transition function* such that

$$\forall s \in S, \sum_{s' \in S} \Delta(s, s') = 1$$

  where $\Delta(s, s')$ describes the probability that the system goes from state s to state s' in one transition,
- $w : S \times S \to \mathbb{N}_0$ is a weight function that associates a strictly positive weight to each transition,
- AP is a set of atomic propositions, and

- $L : S \to 2^{AP}$ is a labelling function.

We assume that all MCs considered in this document are *finite*, i.e., have finite state space.

Remark 1.1 (*Natural labelling*). AP and L are used for *model checking* and can sometimes be omitted. In that case, we consider that $AP = S$ and L is the natural labelling of each state, i.e., for all $s \in S$, $L(s) = \{s\}$.

Property 1.1. Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC and $s \in S$ be a state of $\mathcal{M}$. The transition function Δ defines a probability distribution $\Delta_s : S \to \mathbb{Q} \cap [0, 1]$, $s' \mapsto \Delta(s, s')$ on S.

Definition 1.2 (Underlying graph of a Markov chain). Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC. The *underlying graph of* $\mathcal{M}$ is a directed graph $G = (V, E)$ where the states of the MC act as vertices, i.e., $V = S$, and the edges $(s, s') \in E$ of this graph are pairs of states $(s, s') \in S^2$ such that $\Delta(s, s') > 0$.

We can represent an MC with its underlying graph, where each edge (s, s') of this graph is labelled with the probability to go from s to s' (i.e., $\Delta(s, s')$) and with the weight of this edge (i.e., $w(s, s')$). Additionally, labels of each state can be represented next to it.

Notation 1.1 (*Size of an MC*). An MC $\mathcal{M} = (S, \Delta, w, AP, L)$ is called *finite* if its state space S is finite. The size of $\mathcal{M}$ corresponds to the number of states of S, i.e., $|S|$, plus the number of edges in the underlying graph of $\mathcal{M}$, i.e., the size of the set $\{(s, s') \in S^2 \mid \Delta(s, s') > 0\}$.

Example 1.1 (*Production of solar panels according to weather*). Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be the MC of Figure 1.1. This system models the production of energy (in *kilo Joules*) of an installation of solar panels, according to weather. Here, the states are elements of $S = \{s_0, s_1, s_2, s_3\}$ and the atomic propositions are

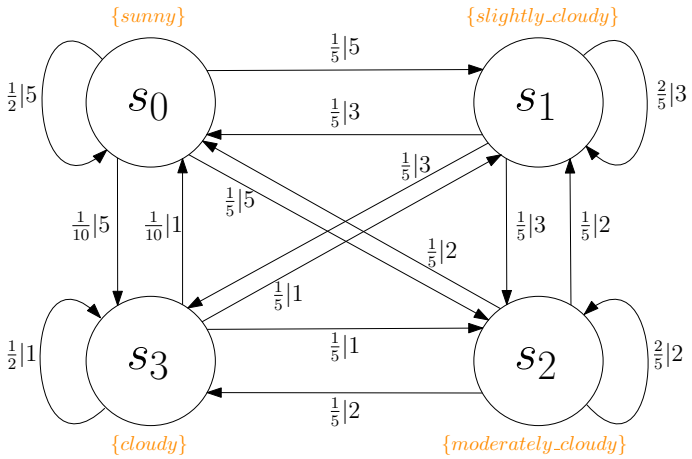


Figure 1.1. MC modelling a daily production of energy (in kJ) of solar panels according to weather

elements of $AP = \{sunny, slightly\_cloudy, moderately\_cloudy, cloudy\}$. The transition function is given by edges in this figure (e.g., $\Delta(s_0, s_1) = \frac{1}{5}$ is the probability

that the day after a sunny day is a slightly cloudy day) as well as the cost of each transition (e.g., $w(s_0, s_1) = 5$). Finally, labels of states are put next to them in orange in the figure (e.g., $L(s_0) = \{sunny\}$). The way w is defined for this MC yields that, when a day is sunny, the installation produces $5kJ$ during this day, $3kJ$ when a day is slightly cloudy, $2kJ$ when a day is moderately cloudy and, finally, $1kJ$ when the day is cloudy.

1.1.1 Paths of Markov chains

Before addressing how to compute probabilities in MCs, we must introduce the notion of paths in MCs. Actually, *probabilistic events* of MCs are sets of paths and prefixes of these paths are used to define these events.

Definition 1.3 (**Path of an MC**)**.** Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC. A *path* $\pi = s_0 s_1 s_2 \dots$ of $\mathcal{M}$ is a (infinite) sequence of states of the MC such that, for all $i \in \mathbb{N}$, $\Delta(s_i, s_{i+1}) > 0$. We denote by $Paths(s)$ the set of paths $\pi = s_0 s_1 s_2 \dots$ of $\mathcal{M}$ starting from the state $s \in S$, i.e., such that $s_0 = s$.

Definition 1.4 (**Finite path of an MC**)**.** Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC. A finite path $\hat{\pi} = s_0 \dots s_n$ of $\mathcal{M}$, with $n \in \mathbb{N}$, is a finite sequence of states of $\mathcal{M}$ such that $\Delta(s_i, s_{i+1}) > 0$ for all $i \in \{0, \dots, n-1\}$. We denote by $Paths_{fin}(s)$ the set of finite paths $\hat{\pi} = s_0 \dots s_n$ starting from the state $s \in S$, i.e., such that $s_0 = s$. Furthermore, each finite path $\hat{\pi} \in Paths_{fin}(s) = s_0 \dots s_n$ is actually a *prefix* of the path $\pi = s'_0 s'_1 s'_2 \dots \in Paths(s)$, where $s_i = s'_i$ for all $i \in \{0, \dots, n\}$. The set of prefixes of the path π is denoted by $Prefs(\pi)$.

1.1.2 Probabilities and events in Markov chains

We introduce now some relevant notions from probability and measure theory.

Definition 1.5 (σ**-algebra**)**.** A σ-algebra is a pair (Ω, σ) where Ω is a non-empty set of *outcomes* and $\sigma \subseteq 2^\Omega$ is a set of *events*, complying with the following rules:

- $\emptyset \in \sigma$,
- if $E \in \sigma$, then $\overline{E} \in \sigma$, where $\overline{E} = \Omega \setminus E$, and
- if $E_i \in \sigma$ for each $i \in \mathbb{N}$, then $\bigcup_{i \in \mathbb{N}} E_i \in \sigma$.

Definition 1.6 (**Probability measure and space**)**.** Let (Ω, σ) be a σ-algebra. A function $\mathbb{P} : \sigma \to [0, 1]$ is a *probability measure* on (Ω, σ) and $(\Omega, \sigma, \mathbb{P})$ is a *probability space* iff

- $\mathbb{P}(\Omega) = 1$, and
- if $(E_i)_{i \in \mathbb{N}}$ is a disjoint sequence of events of σ, then $\mathbb{P}(\bigcup_{i \in \mathbb{N}} E_i) = \sum_{i \in \mathbb{N}} \mathbb{P}(E_i)$.

In the context of probability measure, the events, i.e., the elements of σ, are said to be *measurable*. That is, the measurability of an event E means that E is an element of the σ-algebra (Ω, σ) equipped with the probability measure $\mathbb{P}$, i.e., $E \subseteq \Omega$ with $E \in \sigma$, where its probability is given by $\mathbb{P}(E)$.

We are now interested in probabilistic events of MCs and how measuring them. Formally, for a given MC $\mathcal{M}$ with state space S, and a state $s \in S$ of this MC, there is a probability space $(\Omega, \sigma, \mathbb{P}_s)$, where the paths of $\mathcal{M}$ starting from this state s play the role of the outcomes, i.e., $\Omega = Paths(s)$. The family of measurable events σ is generated by the *cylinder sets* spanned by the prefixes of paths starting from s.

Definition 1.7 (**Cylinder set**)**.** Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC, $s \in S$ be a state of $\mathcal{M}$ and $\hat{\pi} \in Paths_{fin}(s)$ be a finite path of $\mathcal{M}$, the cylinder set formed by the prefix $\hat{\pi}$ is defined as follows:

$$Cyl(\hat{\pi}) = \{\pi \in Paths(s) \mid \hat{\pi} \in Prefs(\pi)\}.$$

A cylinder set spanned with a finite path $\hat{\pi}$ consists thus in the set of paths starting with $\hat{\pi}$. We can define all events according to this formulation, with countable union or complementation of cylinder sets. For example, the event consisting of a singleton containing just one unique path $\pi = s_0 s_1 \dots$ is given by $\bigcap_{i \in \mathbb{N}} C_i$, with $C_i = Cyl(s_0 s_1 \dots s_i)$.

Theorem 1.1 (***Probability measure***)**.** *Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC and $s \in S$ be a state of $\mathcal{M}$. There exists a unique probability measure $\mathbb{P}_s$ on the σ-algebra over $Paths(s)$ where the probabilities of cylinder sets are given by*

$$\mathbb{P}_s(Cyl(s_0 \dots s_n)) = \prod_{i=0}^{n-1} \Delta(s_i, s_{i+1})$$

with $s_0 = s$ and $n \in \mathbb{N}$.

Corollary 1.1. *Any event defined using complementation or countable union of cylinder sets are measurable.*

Example 1.2 (*Measuring an event of an MC modelling a solar panel system*). Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be the MC of Example 1.1. The probability of sunny weather four days in a row starting on a sunny day is given by $\mathbb{P}_{s_0}(Cyl(s_0 s_0 s_0 s_0)) = (\frac{1}{2})^3 = \frac{1}{8}$

1.1.3 Reachability in Markov chains

With this background, we can now introduce some classical problems related to the reachability from a state to a subset of target states in a Markov chain. We will thus first tackle the reachability problem.

Definition 1.8 (**Reachability event**)**.** Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC, $s \in S$, and $T \subseteq S$ be a set of target states, the event of reaching T, denoted by $\Diamond T$, is defined as follows:

$$\Diamond T = \{\pi = s_0 s_1 s_2 \dots \in Paths(s) \mid \exists i \in \mathbb{N},\ s_i \in T\}.$$

The event $\Diamond T$ is measurable since it can be defined as a countable union of cylinder sets. Indeed, let $Paths^T_{fin}(s)$ be the set of finite paths $\hat{\pi} = s_0 \dots s_n \in Paths_{fin}(s)$, such that for all $i \in \{0, \dots, n-1\}$, $s_i \notin T$ and $s_n \in T$,

$$\Diamond T = \bigcup_{s_0 \dots s_n \in Paths^T_{fin}(s)} Cyl(s_0 \dots s_n).$$

By definition of $Paths^T_{fin}(s)$, all of these cylinder sets are disjoint, and we can thus measure $\Diamond T$ with:

$$\mathbb{P}_s(\Diamond T) = \sum_{s_0 \dots s_n \in Paths^T_{fin}(s)} \mathbb{P}_s(Cyl(s_0 \dots s_n))$$

It remains to compute this probability.

Theorem 1.2 (*Probability of reachability).* *Computing $\mathbb{P}_s(\Diamond T)$ for all $s \in S$ can be done in polynomial time through a system of linear equations (cf. Appendix A.1.1 for more details).*

We consider now the *cost of paths* of an MC. Furthermore, we are interested in the cost of paths to reach T. To compute this cost, we use the *truncated sum function.*

Definition 1.9 (**Truncated sum**)**.** Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC, $s \in S$ be a state of $\mathcal{M}$, $\pi = s_0 s_1 s_2 \cdots \in Paths(s)$ be a path of $\mathcal{M}$ and $T \subseteq S$ be a set of target states. The truncated sum of π is the cost to reach T through π for the first time. More formally, the function $\mathsf{TS}^T : Paths(s) \to \mathbb{N} \cup \{\infty\}$ is defined as follows:

$$\mathsf{TS}^T(\pi) = \begin{cases} \sum_{i=0}^{n-1} w(s_i, s_{i+1}) & \text{if } \forall i \in \{0, \dots, n-1\}, s_i \notin T \text{ and } s_n \in T, \\ \infty & \text{else.} \end{cases}$$

We define the truncated sum on finite paths starting from s in the same way.

With this function, it is now possible to introduce the concept of expected cost-to-target as well as the probability of reaching this target set with a bounded cost.

Definition 1.10 (**Expected cost-to-target**)**.** Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC, $s \in S$ be a state of $\mathcal{M}$ and $T \subseteq S$ be a set of target states. We define the expected cost to reach T, i.e., $\mathbb{E}_s(\mathsf{TS}^T)$, corresponding to *the expected truncated sum of paths from s to reach T* as follows:

- if $\mathbb{P}_s(\Diamond T) < 1$, then $\mathbb{E}_s(\mathsf{TS}^T) = \infty$,
- else,

$$\mathbb{E}_s(\mathsf{TS}^T) = \sum_{c=0}^{\infty} c \cdot \mathbb{P}_s(\{\pi \in Paths(s) \mid \mathsf{TS}^T(\pi) = c\}).$$

Remark 1.2 (*Infinite expected cost-to-target).* Assume that $\mathbb{P}_s(\Diamond T) < 1$. We enter in the first condition of the definition of $\mathbb{E}_s(\mathsf{TS}^T)$, i.e., the expected cost-to-target is infinite. Indeed, let

$$A = \{\pi \in Paths(s) \mid \exists \hat{\pi} \in Prefs(\pi),\ \hat{\pi} \in Paths^T_{fin}(s)\}, \text{ and}$$

$$B = \{\pi \in Paths(s) \mid \forall \hat{\pi} \in Prefs(\pi),\ \hat{\pi} \notin Paths^T_{fin}(s)\}.$$

Note that for all $\pi \in B$, $\mathsf{TS}^T(\pi) = \infty$, $A \cap B = \emptyset$, and $A \cup B = Paths(s)$. As we have assumed that $\mathbb{P}_s(A) = \mathbb{P}_s(\Diamond T) < 1$, we have $\mathbb{P}_s(B) > 0$, and then

$$\mathbb{E}_s(\mathsf{TS}^T) = \mathbb{P}_s(A) \cdot \mathbb{E}_s(\mathsf{TS}^T | A) + \mathbb{P}_s(B) \cdot \underbrace{\mathbb{E}_s(\mathsf{TS}^T | B)}_{=\infty} = \infty.$$

Remark 1.3 (*Equivalent characterisation of the expected cost-to-target*). An equivalent characterisation of the expected cost from $s \in S$ to T in case of $\mathbb{P}_s(\Diamond T) = 1$ is given by

$$\mathbb{E}_s(\mathsf{TS}^T) = \sum_{\hat{\pi} \in Paths^T_{fin}(s)} \mathbb{P}_s(Cyl(\hat{\pi})) \cdot \mathsf{TS}^T(\hat{\pi}).$$

Theorem 1.3 (*Solutions of the expected costs-to-target*). *Computing* $\mathbb{E}_s(\mathsf{TS}^T)$ *for all* $s \in S$ *can be done in polynomial time through a system of linear equations (cf. Appendix A.1.2 for more details).*

The last concept that we define in MCs is the cost-bounded reachability.

Definition 1.11 (**Cost-bounded reachability probability**). Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC, $s \in S$ be a state of $\mathcal{M}$, $T \subseteq S$ be a set of target states and $\ell \in \mathbb{N}$ be a cost threshold. The *probability to reach* T *from* s *with a cost bounded by the threshold* ℓ is defined as follows:

$$\mathbb{P}_s(\Diamond_{\leq \ell} T) = \mathbb{P}_s(\{\pi \in Paths(s) \mid \mathsf{TS}^T(\pi) \leq \ell\})$$

The event $\{\pi \in Paths(s) \mid \mathsf{TS}^T(\pi) \leq \ell\}$ is actually measurable: it corresponds to a countable union of cylinder sets formed by prefixes of paths for which the truncated sum is less than or equal to ℓ.

Theorem 1.4 (*Probability of the cost-bounded reachability*). *The probability of the cost bounded reachability to a set of target states* $T \subseteq S$ *from a state* $s \in S$ *can be computed in pseudo-polynomial time*[1] *in the size of* $\mathcal{M}$ *and in the length of* ℓ.

Intuitively, we unfold $\mathcal{M}$ by recording in each state the cost of the current path, yielding a new MC $\mathcal{M}'$. The new set of target states T' in $\mathcal{M}'$ is the set of target states in $\mathcal{M}$ that have a current cost less than or equal to ℓ (cf. Appendix A.2 for more details). Computing the probability of the cost bounded reachability in $\mathcal{M}$ is done by computing the probability of reaching T' in $\mathcal{M}'$. As computing this probability is done in polynomial time in the size of $\mathcal{M}'$, it is done in pseudo-polynomial time in the size of $\mathcal{M}$ and in the length of ℓ.

Now, we study similar problems, but in different systems subject to nondeterminism.

1.2 Markov decision processes

Markov decision processes are systems that model situations describing both nondeterministic and stochastic evolution. In comparison with Markov chains, a Markov decision process requires a decision making to go from a state s to its successors. Indeed, a nondeterministic choice, named here an *action*, must be done when the system is in a state s. When this action is chosen, the system evolves following the probability distribution formed by s and the action chosen.

[1] *pseudo-polynomial time in the length of the threshold value* means that the time complexity is exponential in its encoding

Definition 1.12 (**Markov decision process**). A *Markov decision process*, (MDP, for short) is a tuple $\mathcal{M} = (S, A, \Delta, w, AP, L)$ where

- S is a countable set of states,
- A is a countable set of actions; we denote by $A(s) \in 2^A$ the set of enabled actions when the system is in state s such that, for all $s \in S$, $A(s) \neq \emptyset$,
- $\Delta : S \times A \times S \to [0,1] \cap \mathbb{Q}$ is the probability transition function such that

$$\forall s \in S,\ \forall \alpha \in A(s),\ \sum_{s' \in S} \Delta(s, \alpha, s') = 1,\ \text{and}$$
$$\forall s, s' \in S,\ \forall \alpha \in A \setminus A(s),\ \Delta(s, \alpha, s') = 0,$$

  where $\Delta(s, \alpha, s')$ defines the probability of going from s to s' in one transition when the action $\alpha \in A(s)$ is chosen,
- $w : A \to \mathbb{N}_0$ is a weight function that links a strictly positive weight to each action,
- AP is a set of atomic propositions, and
- $L : S \to 2^{AP}$ is a labeling function.

We assume that all MDPs considered in this document are *finite*, i.e., have finite state space.

Remark 1.4 (*Natural labelling*). Again, AP and L can be omitted. In that case, we consider that $AP = S$ and L is the natural labelling of each state, i.e., for all states $s \in S$, $L(s) = \{s\}$.

Property 1.2. Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP, $s \in S$ be a state of $\mathcal{M}$ and $\alpha \in A(s)$ be an enabled action of the state s. The transition function Δ defines a probability distribution $\Delta_{s,\alpha} : S \to [0,1] \cap \mathbb{Q}$, $s' \mapsto \Delta(s, \alpha, s')$ on S.

Property 1.3. An MC is essentially an MDP where, for each state s, $|A(s)| = 1$.

We will now introduce some useful notations.

Notation 1.2 (*Successors and predecessors*). Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP,

- $Succ(s, \alpha) = \{s' \in S \mid \Delta(s, \alpha, s') > 0\}$ is the set of α-successors of the state $s \in S$ in the MDP, i.e., the set of possible successors of s when the action α is chosen,
- $Succ(s) = \{s' \in S \mid \exists \alpha \in A(s),\ \Delta(s, \alpha, s') > 0\}$ is the set of successors of the state $s \in S$ in the MDP, and
- $Pred(s) = \{s' \in S \mid \exists \alpha \in A(s'),\ \Delta(s', \alpha, s) > 0\}$ is the set of predecessors of the state $s \in S$ in the MDP.

Definition 1.13 (Underlying graph of a Markov decision process). Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP. The *underlying graph of* $\mathcal{M}$ is a directed graph $G = (V, E)$ where the states of the MDP act as vertices, i.e., $V = S$, and the edges $(s, s') \in E$ of this graph are pair of states $(s, s') \in S^2$ such that there exists an action $\alpha \in A(s)$ where $\Delta(s, \alpha, s') > 0$.

Notation 1.3 (*Size of an MDP*). An MDP $\mathcal{M} = (S, A, \Delta, w, AP, L)$ is called *finite* if its state space S is finite. The size of $\mathcal{M}$ corresponds to the number of states of S, i.e., $|S|$, plus the size of the set $\{(s, \alpha, s') \in S \times A \times S \mid \Delta(s, \alpha, s') > 0\}$.

Example 1.3. Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be the MDP of Figure 1.2. We have that $S = \{s_0, s_1, s_2\}$, $A = \{\alpha, \beta, \gamma\}$ and $AP = \{a, b\}$. If the system is currently in state s_0, labeled with $L(s_0) = \{a\}$, the only available action is β (because $A(s_0) = \{\beta\}$). The cost of choosing β is $w(\beta) = 3$. Thus, the system evolves following the probability distribution defined by $\Delta_{s_0,\beta}$: it has a probability of $\frac{1}{2}$ to go to its β-successor s_1 and the same probability to go to its β-successor s_2. Assume that the system evolves to the state s_2, with $L(s_2) = \emptyset$. So, in that case, the system has two possible decisions: α or γ (because $A(s_2) = \{\alpha, \gamma\}$). If α is chosen, the system returns to s_2 with a probability one and a cost of $w(\alpha) = 5$. Else, if γ is chosen, the system goes to s_0 with a probability one and a cost of $w(\gamma) = 2$.

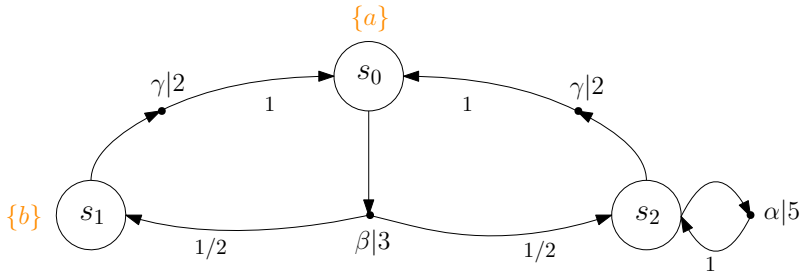


Figure 1.2. MDP with 3 states, 3 actions and 2 atomic propositions

1.2.1 Strategies in Markov decision processes

In order to resolve nondeterminism inside MDPs, we need the notions of paths in MDPs and strategies.

Definition 1.14 (Path in an MDP). Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP and the transition relation

$$\rightarrow = \{(s, \alpha, s') \in S \times A \times S \mid \Delta(s, \alpha, s') > 0\},$$

a path π of $\mathcal{M}$ is defined as a sequence of states and actions

$$\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \dots$$

Let $s \in S$ be a state of $\mathcal{M}$, we denote by $Paths(s)$ the set of paths of $\mathcal{M}$ starting from the state s, i.e., such that $s_0 = s$.

In opposition to MCs, there is no probabilistic space defined on paths of MDPs. This is related to nondeterminism linked to the choices of possible actions: when the system enters in a state s, an enabled action $\alpha \in A(s)$ must be chosen to allow the system to go to one of its successors according to the probability distribution $\Delta_{s,\alpha}$. While this action is not chosen, neither probabilistic distribution on the successors of s is defined. That makes this choice purely non-deterministic. We need strategies to resolve this nondeterminism.

Definition 1.15 (**Histories**)**.** Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP. A *history* of $\mathcal{M}$ is a finite sequence of states $s_0 \dots s_n \in S^+$ where, for all $i \in \{1, \dots, n\}$ there exists an enabled action $\alpha \in A(s_{i-1})$ such that $\Delta(s_{i-1}, \alpha, s_i) > 0$. Intuitively, a history is a sequence of states that brought the system from the state s_0 to the state s_n along a path of $\mathcal{M}$. The set of histories of $\mathcal{M}$ is denoted by $\mathcal{H}(\mathcal{M})$.

Definition 1.16 (**Strategy**)**.** Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP. A *strategy* (also named *policy* or *scheduler* in the literature) for $\mathcal{M}$ is a function $\sigma : \mathcal{H}(\mathcal{M}) \to A$ that selects, for a given history $h = (s_0 \dots s_n) \in \mathcal{H}(\mathcal{M})$, an enabled action of s_n, i.e., $\sigma(s_0 \dots s_n) = \alpha \in A(s_n)$. The path $\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \dots$ is a σ-path iff $\alpha_i = \sigma(s_0 \dots s_{i-1})$ for all $i \in \mathbb{N}_0$. The set of σ-paths starting from the state $s \in S$ is denoted by $Paths^\sigma(s)$.

Strategies that we study here are *pure*, i.e., each action is chosen by the strategy with probability one. We will see later that *randomised* strategies exist and are essential to resolve some types of problems in MDPs (cf. Subsection 3.3.2).

If a strategy controls the decisions of an MDP, then the nondeterminism is solved and the MDP acts like an MC. Actually, an MDP $\mathcal{M}$ controlled by a strategy σ can be formalised as an MC $\mathcal{M}^\sigma$.

Definition 1.17 (**Markov chain induced by a strategy**)**.** Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP and σ be a strategy for $\mathcal{M}$. The MC induced by σ is given by $\mathcal{M}^\sigma = (\mathcal{H}(\mathcal{M}), \Delta^\sigma, w^\sigma, AP, L^\sigma)$, where, for all history $h = s_0 s_1 \dots s_n$ of $\mathcal{M}$,

- $\Delta^\sigma(h, h.s_{n+1}) = \Delta(s_n, \sigma(h), s_{n+1})$
- $w^\sigma(h, h.s_{n+1}) = w(\sigma(h))$
- $L^\sigma(h.s_{n+1}) = L(s_{n+1})$

Property 1.4. Let $\mathcal{M}$ be an MDP with state space S, σ be a strategy on $\mathcal{M}$ and $s \in S$ be a state of $\mathcal{M}$. There exists a bijection between the set $Paths^\sigma(s)$ on $\mathcal{M}$ and the set $Paths(s)$ on the MC induced by the strategy σ, $\mathcal{M}^\sigma$.

Since events of an MC are measurable, events of an MDP controlled by a strategy are also measurable through the MC induced by this strategy.

Notation 1.4. We denote by $\mathbb{P}_s^\sigma$ the probability measure defined on paths starting from the state s of the Markov chain induced by the strategy σ.

Markov chains induced by such strategies can be seen as forests of trees, each of them representing an infinite unfolding of the MDP from one of its states where actions are controlled by the strategy. This is actually due to the fact that each state of these trees represent a history of $\mathcal{M}$, and the size of the set of histories of an MDP is infinite. So, such induced MCs have infinite size. In practice, such strategies are thus not used because they require to know the complete history of the system to decide which action to choose. It is why these strategies are said to have an *infinite-memory*. *Finite-memory strategies* allow to avoid this problem and induce finite-size MCs (in the case of the MDP controlled by the strategy has a finite size).

Definition 1.18 (**Finite-memory strategy**)**.** Let $\mathcal{M}$ be an MDP with state space S, and action space A. A *finite-memory strategy* $\sigma = (Q, \sigma_{act}, \delta, \delta_0)$ is a *Moore machine*, where

- Q is a finite set of *modes*,
- $\sigma_{act} : Q \times S \to A$ is the *next action function* that chooses, for any $s \in S$, an action $\alpha \in A(s)$ following a mode $q \in Q$ in which the machine is currently,
- $\delta : Q \times S \to Q$ is the *transition function*,
- $\delta_0 : S \to Q$ is the *initialisation function* that chooses the initial mode of the machine following a state $s \in S$ of the MDP from which the machine is initialised.

Definition 1.19 (**Markov chain induced by a finite-memory strategy**)**.** Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP and $\sigma = (Q, \sigma_{act}, \delta, \delta_0)$ be a finite-memory strategy for $\mathcal{M}$. The product of $\mathcal{M}$ by σ, given by

$$\mathcal{M} \otimes \sigma = \mathcal{M}^\sigma = (S \times Q, \Delta^\sigma, w^\sigma, AP, L^\sigma)$$

yields the MC $\mathcal{M}^\sigma$, being the MC induced by the finite-memory strategy σ, where, for all states $s, s' \in S$ and for all modes $q, q' \in Q$ of the strategy,

- $\Delta^\sigma((s,q),(s',q')) = \begin{cases} \Delta(s, \sigma_{act}(q,s), s') & \text{if } \delta(q,s) = q', \\ 0 & \text{else,} \end{cases}$
- $w^\sigma((s,q),(s',q')) = w(\sigma_{act}(q,s))$, and
- $L^\sigma(s,q) = L(s)$.

Example 1.4 (*MC induced by a strategy*). Let $\mathcal{M}$ be the MDP of Figure 1.2 and $\sigma = (Q, \sigma_{act}, \delta, \delta_0)$ be the finite-memory strategy of Figure 1.3. We assume that $\delta_0(s_0) = \delta_0(s_1) = m_0$ and $\delta_0(s_2) = m_1$. Based on Figure 1.3, we describe the behaviour of σ as follows. Assume we start in state s_0 in $\mathcal{M}$, the strategy is thus initialised in mode m_0. Once σ reads s_0, the next action function chooses the action β and σ goes to mode m_1 while $\mathcal{M}$ goes from s_0 to s_1 or s_2 with probability $\frac{1}{2}$. Assume that $\mathcal{M}$ makes a transition from s_0 to s_2. In that case, the strategy reads then s_2, the next action function chooses the action α, and σ comes back to mode m_0. Following the probability transition function of $\mathcal{M}$, as α is chosen, the system

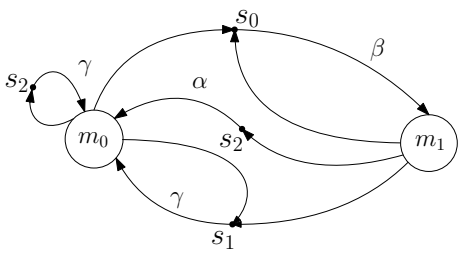


Figure 1.3. Finite-memory strategy for $\mathcal{M}$ with 2 modes

loops and stays in state s_2. Then, s_2 is read by σ, the next action function chooses γ, and σ stays in mode m_0 while the system comes back to state s_0.

So, the strategy simply consists in choosing once α when the system enters in the state s_2, and choosing γ just after that, to return to the state s_0. The MC induced by the product of $\mathcal{M}$ by σ is given in Figure 1.4.

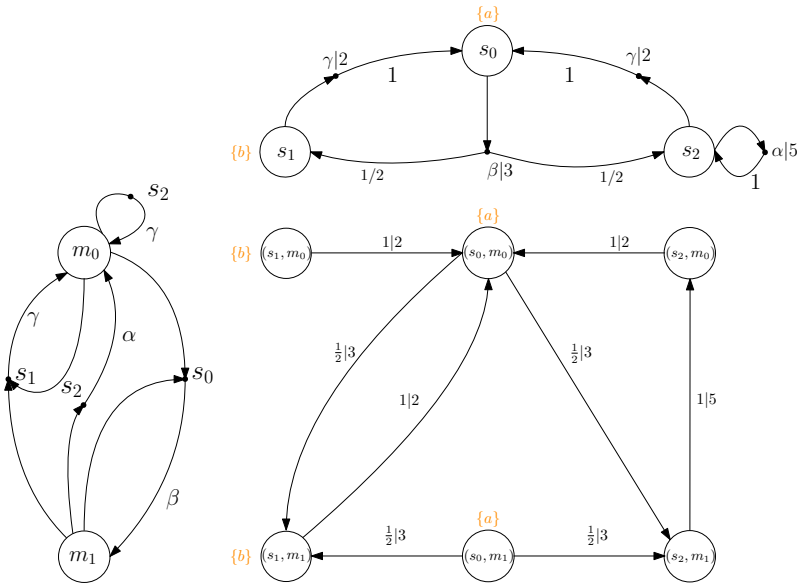


Figure 1.4. Markov chain induced by the strategy σ, defined for the MDP $\mathcal{M}$

Finally, the last type of strategy that we will see is a particular case of finite-memory strategies, where strategies have only one mode. In that case, the action chosen by such a strategy only depends on the current state of the system.

Definition 1.20 (Memoryless strategy). Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP. A *memoryless strategy* is a function $\sigma : S \to A$ that links each state s of $\mathcal{M}$ to an enabled action $\alpha \in A(s)$ of this state.

1.2.2 Stochastic shortest path in Markov decision processes

Problems that we address in this subsection consist in deciding if there exists an optimal strategy allowing to reach a subset of target states with a *"minimal cost"* (we will come back on this notion later). As for MCs, we first address the *stochastic reachability problem* in an MDP.

Definition 1.21 (**SR problem**)**.** Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP, $s \in S$ be a state of $\mathcal{M}$, T be a set of target states, and $\alpha \in [0, 1]$ be a probability threshold. The *stochastic reachability* problem (SR, for short) consists in deciding if there exists a strategy σ such that

$$\mathbb{P}_s^\sigma(\Diamond T) \geq \alpha.$$

Theorem 1.5 (***Solving the* SR *problem).*** *The* SR *problem can be decided in polynomial time in the size of $\mathcal{M}$, and an optimal pure memoryless strategy for this problem can be built in polynomial time.*

In order to solve the SR problem, we define a linear program (LP, for short) of polynomial size in the size of the model. The optimal solution of this LP corresponds to the maximal probability to reach T from each state of the MDP, and is used to build a pure memoryless strategy. This strategy is thus said to be *optimal* (cf. Appendix A.3.1 for more details).

Before introducing the rest, we first introduce the *truncated sum* of paths of MDPs in order to compute the cost of paths of an MDP.

Definition 1.22 (**Truncated sum for MDPs**)**.** Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP, $s \in S$ be a state of $\mathcal{M}$, $T \subseteq S$ be a set of target states and $\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \cdots \in Paths(s)$ be a path of $\mathcal{M}$ starting from s. The truncated sum $\mathsf{TS}^T : Paths(s) \to \mathbb{N} \cup \{\infty\}$ of the path π is defined as follows:

$$\mathsf{TS}^T(\pi) = \begin{cases} \sum_{i=1}^{n} w(\alpha_i) & \text{if } \forall i \in \{0, \ldots, n-1\}, s_i \notin T \text{ and } s_n \in T \\ \infty & \text{else.} \end{cases}$$

By considering the cost of paths of an MDP, we are now interested in building strategies inducing the shortest path to go from a state of the model to a set of target states. The notion of shortest path in an MDP is not as obvious as in a graph due to the uncertainty linked to the stochastic behaviour of the MDP. The stochastic shortest path problem in an MDP can be introduced in different ways. We will first tackle the *stochastic shortest path expectation* problem and then tackle the *stochastic shortest path percentile* problem.

Since events of any MDP controlled by a strategy are measurable and since the cost of paths of any MDP is computable via the truncated sum function, we can compute the expected cost of paths that reach a set of target states using a strategy, and more particularly build the strategy that minimises the expected cost of these paths.

Definition 1.23 (**SSP-E problem**)**.** Let $\mathcal{M}$ be an MDP with state space S, $s \in S$ be a state of $\mathcal{M}$, $T \subseteq S$ be a set of target states, and $\ell \in \mathbb{N}$ be a cost threshold. The *stochastic shortest path expectation* problem (SSP-E, for short) consists in deciding if there exists a strategy σ such that

$$\mathbb{E}_s^\sigma(\mathsf{TS}^T) \leq \ell,$$

where $\mathbb{E}_s^\sigma(\mathsf{TS}^T)$ corresponds to the expected cost of paths starting from s in the MC induced by σ.

Theorem 1.6 (*Solving the* SSP-E *problem).* *The* SSP-E *problem can be decided in polynomial time in the size of $\mathcal{M}$, and an optimal pure memoryless strategy for this problem can be built in polynomial time in the size of $\mathcal{M}$.*

As for the SR problem, we solve the problem by defining an LP of polynomial size of the model, for which the optimal solution corresponds to the minimal expected cost-to-target from every state of the model. This solution is used to build an optimal memoryless strategy for the problem (cf. Appendix A.3.2 for more details).

Finally, the last problem that we present in this chapter is the *stochastic shortest path percentile* problem. For this problem, we are interested in deciding if there exists a strategy allowing to reach a set of target states with a bounded cost and with high probability.

Definition 1.24 (SSP-P **problem**)**.** Let $\mathcal{M}$ be an MDP with state space S, $s \in S$ be a state of $\mathcal{M}$, $T \subseteq S$ be a set of target states, $\ell \in \mathbb{N}$ be a cost threshold, and $\alpha \in [0,1] \cap \mathbb{Q}$ be a probability threshold. The *stochastic shortest path percentile* problem (SSP-P, for short) consists in deciding if there exists a strategy σ such that

$$\mathbb{P}_s^\sigma(\Diamond_{\leq \ell} T) \geq \alpha.$$

Theorem 1.7 (*Solving the* SSP-P *problem*)**.*** *The* SSP-P *problem can be decided in pseudo-polynomial time in the size of $\mathcal{M}$ and in the length of ℓ. An optimal pure finite-memory strategy (more precisely pseudo-polynomial-memory) for this problem can be built in pseudo-polynomial time in the size of $\mathcal{M}$ and in the length of ℓ.*

In order to solve the SSP-P problem for a given MDP $\mathcal{M}$, a state s of $\mathcal{M}$, a cost threshold ℓ, and a probability threshold α, we define an new MDP $\mathcal{M}_\ell$, being the *unfolding of $\mathcal{M}$ from s up to ℓ*. The idea is building an optimal strategy for the SR problem for a set of new target states in $\mathcal{M}_\ell$, representing the target states in $\mathcal{M}$ for which the threshold ℓ has not been exceeded by paths starting from s, and for the probability threshold α (cf. Algorithm 8 in Appendix A.4). The strategy built to solve the SR problem for $\mathcal{M}_\ell$ is pure and memoryless in $\mathcal{M}_\ell$, but pseudo-polynomial-memory for $\mathcal{M}$.

Definition 1.25 (**Unfolding of an MDP up to a cost threshold**)**.** Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP, $s^* \in S$ be a state of $\mathcal{M}$, $T \subseteq S$ be a set of target states of $\mathcal{M}$ and $\ell \in \mathbb{N}$ be a cost threshold. Unfolding $\mathcal{M}$ from s^* up to the threshold ℓ can be done as follows: we build $\mathcal{M}_\ell = (S_\ell, A_\ell, \Delta_\ell, w, AP, L_\ell)$ for the subset $T_\ell \subseteq S_\ell$ where

- S_ℓ is composed of states (s, v) where $s \in S$ and $v \in \{0, \dots, \ell\} \cup \{\bot\}$. We consider that $\bot > \ell$, with $\bot + v = \bot$ for all $v \in \{0, \dots, \ell\}$. Intuitively, v records the cost of paths while unfolding $\mathcal{M}$.
- For each $\alpha \in A$, we have $\alpha \in A_\ell$ and for all $(s, v) \in S_\ell$, $A_\ell(s, v) = A(s)$.

- $\Delta_\ell : S_\ell \times S_\ell \to [0,1]$ is the probability transition function such that for all $(s,v),(s',v') \in S_\ell$ and for all $\alpha \in A(s)$,

$$\Delta_\ell((s,v),\alpha,(s',v')) = \begin{cases} \Delta(s,\alpha,s') & \text{if } v' = v + w(\alpha) \leq \ell \text{ or} \\ & \text{if } v' = \bot \text{ and } v + w(\alpha) > \ell, \\ 0 & \text{else.} \end{cases}$$

- $L_\ell : S_\ell \to AP,\ (s,v) \mapsto L(s)$.

Target states are states of $T_\ell = \{(s,v) \mid s \in T \wedge v \leq \ell\}$.

Remark 1.5 (*Optimised unfolding*). Firstly, as we unfold $\mathcal{M}$ from s^*, we limit the state space S_ℓ with the states reachable from $(s^*,0)$. Then, since all state $(s,v) \in S_\ell$ such that $v = \bot$ can never reach T_ℓ, it is not useful to keep these states in the unfolded MDP $\mathcal{M}_\ell$. So, we can replace all these states in S_l with a unique state $s_\bot$ such that, for all $(s,v) \in S_\ell$ where $v \neq \bot$ and for all $\alpha \in A(s)$ such that $v + w(\alpha) > \ell$, $\Delta_\ell((s,v),\alpha,s_\bot) = 1$. Then, we define a new action $\alpha_\bot \in A_\ell$ with an arbitrary weight that will allow a self-loop for $s_\bot$, i.e., $\Delta_\ell(s_\bot,\alpha_\bot,s_\bot) = 1$. Furthermore, we can apply the same rule for all states $(s,v) \in S_\ell$ such that s cannot reach T in the underlying graph of $\mathcal{M}$.

Example 1.5 (*Unfolding of an MDP*). Let $\mathcal{M}$ be the MDP of Example 1.3. This MDP can be unfolded from the state s_0 up to the threshold $\ell = 8$ for the target states set $\{s_1\}$ (cf. Figure 1.5). The highlighted strategy in Figure 1.5 is the optimal

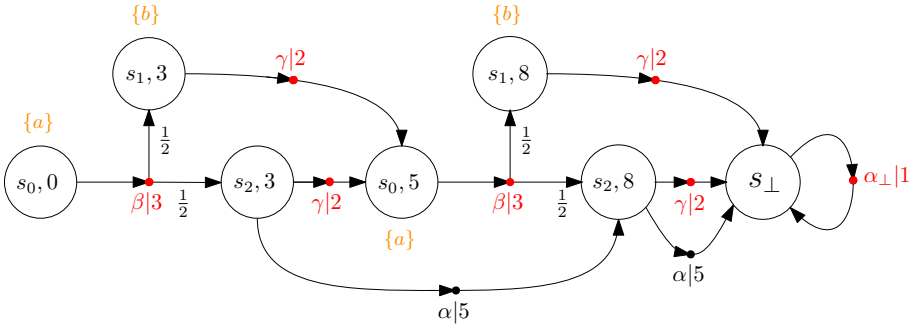


Figure 1.5. Unfolding of $\mathcal{M}$ from s_0 up to $\ell = 8$ for $\{s_1\}$

memoryless strategy satisfying the SR problem in $\mathcal{M}_\ell$. This strategy satisfy the SSP-P problem in $\mathcal{M}$ and is thus finite-memory in $\mathcal{M}$.

Chapter 2

Model checking for Markov decision processes

Concepts, theorems and definitions of this chapter are essentially inspired by chapters and sections of the book "Principles of model checking" [BK08] treating of linear temporal logic, computation tree logic, probabilistic computation tree logic and probabilistic rewards computation tree logic.

In this chapter, we present a way to *model-check* Markov decision processes, i.e., verify that properties hold in states of a given model. We focus on model checking via a *probabilistic branching-time logic*. We detail the syntax, the semantic and model checking algorithms of this logic. Furthermore, we extend it to support weights of Markov decision processes, and we then formulate requests to solve problems we encountered in the first chapter.

In order to introduce model checking for Markov decision processes, we first present the basic events used to form properties in this logic. In a Markov decision process, a strategy is required to resolve nondeterminism. The events are then measured with the probability measure of the probability space defined on paths starting from a given state of the Markov chain induced by such a strategy. We thus begin by formally introducing these events and presenting a way to compute them in Markov chains, before doing the same in Markov decision processes.

2.1 Temporal events

In Markov chains, *temporal events* allow to describe properties related to the reachability to a set of target states in the system. These properties are classified into two categories: *quantitative* and *qualitative* properties. Quantitative properties may assert the probability to reach a subset of target states, avoiding some bad states in an infinite or finite number of steps, and qualitative properties are special cases of quantitative properties where probabilities are trivial (i.e., zero or one). We present in this section a way to determine the probability of events describing such properties, which will be useful later to model-check Markov chains and Markov decision processes in a probabilistic branching time logic.

2.1.1 Constrained reachability

Definition 2.1 (**Constrained reachability events**)**.** Let $\mathcal{M}$ be an MC with state space S, $s \in S$, $T, C \subseteq S$, and $n \in \mathbb{N}$. The events $C \,\mathsf{U}^{\leq n}\, T$ and $C \,\mathsf{U}\, T$ are sets of

paths starting from s describing the constrained reachability by the set of states C to the set of target states T. These events are defined as follows:

$$C \,\mathsf{U}^{\leq n}\, T = \{s_0 s_1 s_2 \cdots \in Paths(s) \mid \exists j \leq n,\ s_j \in T, \text{ and } \forall i < j,\ s_j \in C\},$$
$$C \,\mathsf{U}\, T = \{s_0 s_1 s_2 \cdots \in Paths(s) \mid \exists j \in \mathbb{N},\ s_j \in T, \text{ and } \forall i < j,\ s_j \in C\}.$$

Their notation is inspired by *PCTL* (a probabilistic branching time logic, cf. Section 2.2).

Lemma 2.1 (Measurability of the constrained reachability events)**.** *Let $\mathcal{M}$ be an MC with state space S, $s \in S$, $C, T \subseteq S$, and $n \in \mathbb{N}$. The events $C \,\mathsf{U}^{\leq n}\, T$ and $C \,\mathsf{U}\, T$ are measurable.*

Proof: The events $C \,\mathsf{U}^{\leq n}\, T$ and $C \,\mathsf{U}\, T$ are measurable since they can be defined as countable union of cylinder sets:

$$C \,\mathsf{U}^{\leq n}\, T = \bigcup_{k=0}^{n} \bigcup_{s_0 \ldots s_k \in Paths_{fin}^{C,k,T}(s)} Cyl(s_0 \ldots s_k),$$
$$C \,\mathsf{U}\, T = \bigcup_{k \in \mathbb{N}} \bigcup_{s_0 \ldots s_k \in Paths_{fin}^{C,k,T}(s)} Cyl(s_0 \ldots s_k)$$

where $n \in \mathbb{N}$, and

$$Paths_{fin}^{C,n,T}(s) = \{s_0 \ldots s_n \in Paths_{fin}(s) \mid \forall i < n,\ s_i \in C \setminus T \ \wedge s_n \in T\}.$$

$\mathbb{P}_s(C \,\mathsf{U}^{\leq n}\, T)$ and $\mathbb{P}_s(C \,\mathsf{U}\, T)$ denote the probability of the union of cylinder sets of finite paths of $Paths_{fin}^{C,k,T}(s)$ starting from the state $s \in S$. □

The constrained reachability event allows to derivate some other classical events:

- $\Diamond T = S \,\mathsf{U}\, T$, i.e., *reach* T (cf. Subsection 1.1.3 in the first chapter),
- $\Diamond^{\leq n} T = S \,\mathsf{U}^{\leq n}\, T$, i.e., reach T *in at most* n *steps*,
- $\Box T = \overline{\Diamond \overline{T}}$, i.e., *always visit* T,
- $\Box^{\leq n} T = \overline{\Diamond^{\leq n} \overline{T}}$, i.e., *repeat* T *at least* n *times*.

In order to compute the probability of such events, it is sufficient to compute the probability of the constrained reachability.

Let $s \in S$, $C, T \subseteq S$, and $n \in \mathbb{N}$. In order to compute $\mathbb{P}_s(C \,\mathsf{U}^{\leq n}\, T)$ and $\mathbb{P}_s(C \,\mathsf{U}\, T)$, we introduce the following notations. Let $S_{=0}$, $S_{=1}$ and $S_{=?}$ be partitions of S such that

- $T \subseteq S_{=1} \subseteq \{s \in S \mid \mathbb{P}_s(C \,\mathsf{U}\, T) = 1\}$,
- $S \setminus (C \cup T) \subseteq S_{=0} \subseteq \{s \in S \mid \mathbb{P}_s(C \,\mathsf{U}\, T) = 0\}$,
- $S_{?} = S \setminus (S_{=1} \cup S_{=0})$.

Then, let A be a quadratic matrix with rows and columns referring the states of $S_?$. This matrix is obtained from the transition function Δ as follows:

$$A_{i,j} = \Delta(s_i, s_j) \quad \forall s_i, s_j \in S_?$$

Let $n_? = |S_?|$. Similarly, let b be a vector of probabilities in $[0,1]^{n_?}$, defined as $(b_s)_{s \in S_?}$ such that $b_s = \sum_{t \in S_{=1}} \Delta(s,t)$, i.e., such that b_s describes the probability of reaching a state of $S_{=1}$ in one outgoing transition of the state s.

Notation 2.1 (*Lifted probability transition*). Let $s \in S$ be a state of $\mathcal{M}$. We denote by $\Delta(s,T)$ the probability of reaching T in one outgoing transition of s, i.e., $\Delta(s,T) = \sum_{t \in T} \Delta(s,t)$.

So, we have $b_s = \Delta(s, S_{=1})$ for each $s \in S_?$.

Theorem 2.1 (*Least fixed point characterisation for constrained reachability).* *Let x be a vector of probabilities in $[0,1]^{n_?}$ such that $x = (\mathbb{P}_s(C \,U T))_{s \in S_?}$. This vector is the least fixed point of the operator $\Upsilon : [0,1]^{n_?} \to [0,1]^{n_?}$ which is given by*

$$\Upsilon(y) = Ay + b$$

Furthermore, let $x^{(0)} = \vec{0}$ be the vector consisting of zeros only and $x^{(n+1)} = \Upsilon(x^n)$ for $n \in \mathbb{N}$, then

- $x^{(n)} = (x_s^{(n)})_{s \in S_?}$, *where* $x_s^{(n)} = \mathbb{P}_s(C \,U^{\leq n} S_{=1})$ *for each state* $s \in S_?$,
- $x^{(0)} \leq x^{(1)} \leq x^{(2)} \leq \cdots \leq x$, *and*
- $x = \lim_{n \to \infty} x^{(n)}$.

where $\leq$ denotes here the partial order relation

$$\leq = \{(y, y') \in [0,1]^{n_?} \times [0,1]^{n_?} \mid y_s \leq y'_s \ \ \forall s \in S_?\}.$$

By definition of A and b, for any $s \in S_?$ and for $y = (y_s)_{s \in S_?} \in [0,1]^{n_?}$,

$$(\Upsilon(y))_s = y'_s = \sum_{s' \in S_?} \Delta(s,s') \cdot y_{s'} + \Delta(s, S_{=1}).$$

Since $0 \leq y_s \leq 1$ for all $s \in S?$, $\Delta(s,s') \geq 0$, and $\sum_{s' \in Succ(s)} \Delta(s,s') = 1$, this implies $0 \leq y_{s'} \leq 1$. We then have that Υ is a well-defined function from $[0,1]^{n_?}$ to $[0,1]^{n_?}$. Furthermore, let $x_s = \mathbb{P}_s(C \,\mathsf{U}\, T)_{s \in S_?}$ for each state $s \in S$, by definition of Υ, we intuitively have

$$\begin{aligned}
x^{(0)} &= \vec{0}, \\
x^{(1)} &= Ax^{(0)} + b \\
&= b, && \text{(states of } C \text{ reaching } S_{=1} \text{ in one transition)} \\
x^{(2)} &= Ax^{(1)} + b \\
&= Ab + b \\
&= \Big(\sum_{s' \in S_?} \Delta(s,s') \cdot \Delta(s', S_{=1}) + \Delta(s, S_{=1}) \Big)_{s \in S_?} \\
& && \text{(states of } C \text{ reaching } S_{=1} \text{ in max. two transition steps)}
\end{aligned}$$

$$
\begin{aligned}
&\cdots \\
x^{(n)} &= Ax^{(n-1)} + b \qquad \textit{(states of } C \textit{ reaching } S_{=1} \textit{ in max. } n \textit{ transition steps)} \\
&= A^{n-1}b + A^{n-2}b + \cdots + Ab + b \\
&\cdots \\
x &= Ax + b \\
x &= (\sum_{s' \in S} \Delta(s,s') \cdot x_{s'} + \Delta(s, S_{=1}))_{s \in S_?} \qquad \textit{(states of } C \textit{ eventually reaching } S_{=1})
\end{aligned}
$$

Remark 2.1 (*Choosing $S_{=0}$ and $S_{=1}$*). For efficiency reasons, it is a good idea to deal with the largest subsets $S_{=0}$ and $S_{=1}$. Indeed, this allows to reduce the size of the matrix A and allows faster computations to determine the probability of constrained reachability events. However, under the assumption that $S_{=0}$ is a proper subset of $\{s \in S \mid \mathbb{P}_s(C \,\mathsf{U}\, T) = 0\}$, the operator Υ may have more than one fixed point. Let $\mathcal{M}$ be the MC of Figure 2.1 and $T = \{t\}$. Consider the event $\Diamond T$. Let us take

Figure 2.1. Markov chain $\mathcal{M}$ with 2 states : s and t

the smallest subsets $S_{=1} = T$ and $S_{=0} = S \setminus (S \cup T) = \emptyset$, thus $S_? = S \setminus T = \{s\}$. Following Theorem 2.1, let $x = (\mathbb{P}_s(C \,\mathsf{U}\, T))_{s \in S_?}$, where $x = Ax + b$. As $x = x_s$ and $b = 0$ (because $\Delta(s,t) = 0$), we have $x = Ax$, with $A = 1$ (because $\Delta(s,s) = 1$). The related operator $\Upsilon : [0,1] \to [0,1]$ is given by $\Upsilon(y_s) = y_s$ for any $y_s \in [0,1]$ and has then infinitely many fixed points.

The unicity of the fixed point is actually guaranteed if $\mathcal{M}$ is finite and if we take the largest subset for $S_{=0}$, i.e., $S_{=0} = \{s \in S \mid \mathbb{P}_s(C \,\mathsf{U}\, T) = 0\}$.

Theorem 2.2 (*Unique solution for the constrained reachability).* *Let $\mathcal{M}$ be a Markov chain with state space S, the subsets $T, C \subseteq S$,*

$$S_{=0} = \{s \in S \mid \mathbb{P}_s(C \,\mathsf{U}\, T) = 0\}, \quad T \subseteq S_{=1} \subseteq \{s \in S \mid \mathbb{P}_s(C \,\mathsf{U}\, T) = 1\},$$

and $S_? = S \setminus (S_{=0} \cup S_{=1})$. Then , the vector $(\mathbb{P}_s(C \,\mathsf{U}\, T))_{s \in S_?}$ is the unique fixed point of Υ and the unique solution of the equation system $x = Ax + b$, where $A = (\Delta(s,s'))_{s,s' \in S_?}$ and $b = (\Delta(s, S_{=1}))_{s \in S_?}$.

Remark 2.2 (*Reachability problem*). Remind the reachability problem addressed in the first chapter (i.e., computing $\mathbb{P}_s(\Diamond T)$ for a subset of target states T and for any $s \in S$, cf. Subsection 1.1.3), the linear equations system resolving the problem (cf. Appendix A.1.1) is actually derived from Theorem 2.2.

Remark 2.3 (*Probability of the always event*). Let $\mathcal{M}$ be an MDP with state space S $s \in S$, and $T \subseteq S$, the event $\Box T$ is equal to $\overline{\Diamond \overline{T}}$. Since we can compute $\mathbb{P}_s(S \,\mathsf{U}\, \overline{T}) = \mathbb{P}_s(\Diamond \overline{T})$ (cf. Theorem 2.2), we have $\mathbb{P}_s(\Box T) = 1 - \mathbb{P}_s(\Diamond(S \setminus T))$.

We now provide a way to compute the largest subset $S_{=0}$. Note that we will also see later a way to compute the largest subset $S_{=1}$ in order to allow faster computations to solve the system of equations defined in Theorem 2.2 (cf. Corollary 2.2).

Definition 2.2 (**Paths satisfiability relation with constrained reachability events**)**.** Let $\mathcal{M}$ be an MC with state space S and $\pi \in Paths(s)$ be a path starting from any state $s \in S$. The path π *satisfies* $C \mathsf{U} T$ iff there exists a prefix $\hat{\pi} \in Pref(\pi)$ and a number of steps $k \in \mathbb{N}$ such that $\hat{\pi} \in Paths_{fin}^{C,k,T}(s)$. This relation is denoted by $\models$, with $\pi \models C \mathsf{U} T$.

Lemma 2.2 (Zero probability equivalence of constrained reachability events)**.** *Let $s \in S$ be a state in an MC with state space S and $C, T \subseteq S$, the two following propositions are equivalent:*

(a.) $\mathbb{P}_s(C \mathsf{U} T) = 0$.

(b.) $\forall \pi \in Paths(s),\ \pi \not\models (C \mathsf{U} T)$.

Proof:
$((b.) \implies (a.))$. Assume that $\forall \pi \in Paths(s),\ \pi \not\models C \mathsf{U} T$. So, by definition of the satisfaction relation $\models$, that means there does not exist any finite path $\hat{\pi} \in Paths_{fin}(s)$ such that $\hat{\pi} \in Paths_{fin}^{C,k,T}(s)$ for some $k \in \mathbb{N}$. By definition of $C \mathsf{U} T$, that implies $C \mathsf{U} T = \emptyset$ and then $\mathbb{P}_s(\emptyset) = 0$.
$(\neg(b.) \implies \neg(a.))$. Assume there exists a path $\pi \in Paths(s)$ such that $\pi \models C \mathsf{U} T$. Then, that means there exists a prefix $\hat{\pi} = s_0 \dots s_k \in Pref(\pi)$ and a number of steps $k \in \mathbb{N}$ such that $\hat{\pi} \in Paths_{fin}^{C,k,T}(s)$. So, we have at least one cylinder set in the union forming the event $C \mathsf{U} T$ (i.e., $Cyl(s_0 \dots s_k)$), and then $\mathbb{P}_s(C \mathsf{U} T) \geq \prod_{i=0}^{k-1} \Delta(s_i, s_{i+1}) > 0$. □

Lemma 2.3 (Computing $S_{=0}$ with graph theory)**.** *Let $s \in S$ be a state in an MC with state space S and $C, T \subseteq S$, the statement $\forall \pi \in Paths(s),\ \pi \not\models (C \mathsf{U} T)$ can be decided in polynomial time in the size of $\mathcal{M}$ with a graph theory based algorithm.*

Proof: The set $B = \{s \in S \mid \exists \pi \in Paths(s),\ \pi \models (C \mathsf{U} T)\}$ is the smallest subset of S such that

(a.) $T \subseteq B$, and

(b.) $\forall s \in S,\ s \in C \ \wedge\ Succ(s) \cap B \neq \emptyset \implies s \in B$.

Indeed,

1. Assume that $B = \{s \in S \mid \exists \pi \in Paths(s),\ \pi \models C \mathsf{U} T\}$ and let us show that B satisfies (a.) and (b.). We obviously have that $T \subseteq B$ because all paths starting from $t \in T$ satisfies $C \mathsf{U} T$: take the finite path consisting in the single state t, we have that $t \in Paths_{fin}^{C,0,T}(t)$. Then, let $s \in S$ and assume that $s \in C \setminus T$ and $Succ(s) \cap B \neq \emptyset$. That means there exists a successor s' of s such that $s' \in B$. Thus, there exists a path $\pi \in Paths(s')$ such that $\pi \models C \mathsf{U} T$,

by definition of B. Furthermore, we have by definition of the satisfaction relation $\models$ that there exists a prefix $\hat{\pi} \in Pref(\pi)$ and a steps number $k \in \mathbb{N}$ such that $\hat{\pi} \in Paths_{fin}^{C,k,T}(s')$. So, let take the finite path $s.\hat{\pi}$, we have that $s.\hat{\pi} \in Paths_{fin}^{C,k+1,T}(s)$ and thus that $s.\pi \models C \mathsf{U} T$. That yields $s \in B$.

2. Let $B \subseteq S$ be the smallet set satisfying (a.) and (b.). Let us show that $\{s \in S \mid \exists \pi \in Paths(s),\ \pi \models C \mathsf{U} T\} \subseteq B$. First, if $\{s \in S \mid \exists \pi \in Paths(s),\ \pi \models C \mathsf{U} T\} = \emptyset$, then we trivially have that our assumptions are verified. Then, assume that $\{s \in S \mid \exists \pi \in Paths(s),\ \pi \models C \mathsf{U} T\} \neq \emptyset$ and let $s \in \{s \in S \mid \exists \pi \in Paths(s),\ \pi \models C \mathsf{U} T\}$. If $s \in T$, then $s \in B$ by (a.). Else, we have that there exists a path $\pi = s_0 s_1 s_2 \cdots \in Paths(s)$ starting from the state $s_0 = s$ such that $\pi \models C \mathsf{U} T$. By definition of the satisfaction relation $\models$, that means there exists $\hat{\pi} \in Pref(\pi)$ and a number of steps $k \in \mathbb{N}$ such that $\hat{\pi} \in Paths_{fin}^{C,k,T}(s)$, i.e., such that $\forall i \in \mathbb{N},\ i < k,\ s_i \in C \setminus T$ and $s_k \in T$. Let $i \in \{0, \ldots, k-1\}$. Assume that $s_{i+1} \in B$. Then, since $s_i \in C$ and $s_{i+1} \in Succ(s_i)$, we have $s_i \in B$ by (b.). Since $s_k \in T$ and $T \subseteq B$, we have by induction (from k to 0) on i that $s_0 = s \in B$. Thus, we finally have that $\{s \in S \mid \exists \pi \in Paths(s),\ \pi \models C \mathsf{U} T\} \subseteq B$.

Then, we can compute the set B with the following algorithm:

Algorithm 1 Smallest fixed point computation

Input : a Markov chain $\mathcal{M}$ with state space S, and $C, T \subseteq S$
Output : the set $\{s \in S \mid \exists \pi \in Paths(s),\ \pi \models (C \mathsf{U} T)\}$

```
B ← T
while A ← {s ∈ C \ B | Succ(s) ∩ B ≠ ∅} ≠ ∅ do
    B ← B ∪ A
return B
```

Finally, the set

$$\{s \in S \mid \mathbb{P}_s(C \mathsf{U} T) = 0\} = \{s \in S \mid \forall \pi \in Paths(s),\ \pi \not\models (C \mathsf{U} T)\}$$

is equal to $S \setminus B$. □

Example 2.1 (*Constrained reachability of an MC modelling the production of solar panels according to weather*). Let $\mathcal{M}_{sp} = (S, \Delta, w, AP, L)$ be the MC of Figure 2.2 (cf. Example 1.1 in the first chapter for more details about this MC). We are interested to know the probability that the weather goes from sunny to cloudy in at most three days (starting on a sunny day), i.e., $\mathbb{P}_{s_0}(\{s_0\} \mathsf{U}^{\leq 2} \{s_3\})$. Let $S_{=1} = \{s_3\}$ and $S_{=0} = \{s_1, s_2\}$, we have $S_? = \{s_0\}$. The least fixed point characterisation suggests the following iterative scheme:

- $x^{(0)} = 0$,
- $x^{(1)} = \Delta(s_0, s_0) \cdot x^{(0)} + \Delta(s_0, s_3) = \frac{1}{10}$, and
- $x^{(2)} = \Delta(s_0, s_0) \cdot x^{(1)} + \Delta(s_0, s_3) = \frac{1}{2} \cdot \frac{1}{10} + \frac{1}{10} = \frac{3}{20}$,

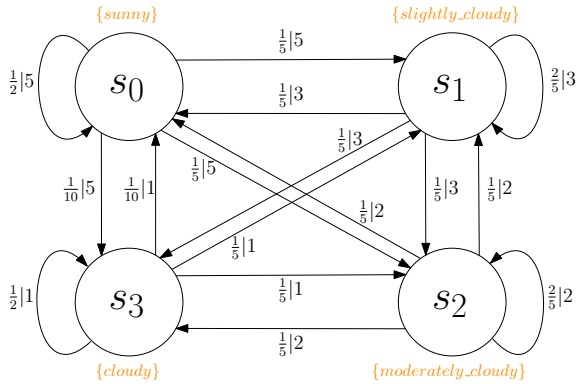


Figure 2.2. MC modelling a daily production of energy (in kJ) of solar panels according to weather

with $x^{(2)} = \mathbb{P}_{s_0}(\{s_0\} \mathsf{U}^{\leq 2} \{s_3\})$.

Example 2.2 (*Measuring constrained reachability event in an MC*). Let $\mathcal{M} = (S, \Delta)$ be the MC of Figure 2.3, $C = \{s_0, s_2, s_3\}$ and $T = \{s_1\}$. We are interested in the probability of the constrained reachability $C \mathsf{U} T$.

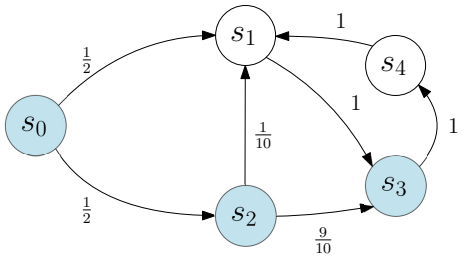


Figure 2.3. MC $\mathcal{M}$ with state space $S = \{s_0, s_1, s_2, s_3, s_4\}$

Let $S_{=1} = \{s_1\}$ and $S_{=0} = \{s \in S \mid \mathbb{P}_s(C \mathsf{U} T) = 0\} = \{s_3, s_4\}$, we have $S_? = \{s_0, s_2\}$. The vector $(\mathbb{P}_s(C \mathsf{U} T))_{s \in S_?}$ is the unique solution of the following system:

$$\begin{pmatrix} x_{s_0} \\ x_{s_2} \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_{s_0} \\ x_{s_2} \end{pmatrix} + \begin{pmatrix} \frac{1}{2} \\ \frac{1}{10} \end{pmatrix}$$
$$\begin{pmatrix} x_{s_0} \\ x_{s_2} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} x_{s_2} \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} \\ \frac{1}{10} \end{pmatrix}$$

So, $\mathbb{P}_{s_2}(C \mathsf{U} T) = x_{s_2} = \frac{1}{10}$ and $\mathbb{P}_{s_0}(C \mathsf{U} T) = x_{s_0} = \frac{11}{20}$.

In order to optimise the resolution of systems of linear equations defined to compute the probabilities of constrained reachability events, we need to compute the largest subset $S_{=1}$, and this can be done by looking at the limit behaviour of Markov chains.

2.1.2 Limit behaviour of Markov chains

Now, we will focus on events characterising the *long-run* behaviour of Markov chains. We will address here two events : $\Box\Diamond T$ (T is reached *infinitely often*) and $\Diamond\Box T$ (T is *persistent*), for any subset of states $T \subseteq S$. We begin by defining these events and show that they are measurable. After that, we introduce some graph theory notions, allowing to study the limit behaviour of Markov chains. Studying the limit behaviour of Markov chains allows to optimise computations during the resolution of problems requiring linear equations systems or linear programs. Moreover, it allows fast computations to determine probabilities of *infinitely often* and *persistent* events.

Definition 2.3 (**Infinitely often event**)**.** Let $\mathcal{M}$ be an MC with state space S, $s \in S$, and $T \subseteq S$ be a subset of states of $\mathcal{M}$. For any path $\pi = s_0 s_1 s_2 \cdots \in Paths(s)$, encountering *infinitely often* T actually means that, for all step $n \in \mathbb{N}$ (and thus, for any state s_n of π), there exists a state such that $s_m \in T$, with $m \geq n$. Then, the infinitely often event for T is defined as follows:

$$\Box\Diamond T = \{s_0 s_1 s_2 \cdots \in Paths(s) \mid \forall n \in \mathbb{N},\, \exists m \geq n,\, s_m \in T\}.$$

Definition 2.4 (**Persistence event**)**.** Let $\mathcal{M}$ be an MC with state space S, $s \in S$, and $T \subseteq S$ be a subset of states of $\mathcal{M}$. For any path $\pi = s_0 s_1 s_2 \cdots \in Paths(s)$, T is *persistent* actually means there exists a step $n \in \mathbb{N}$ such that, for all $i \geq n$, $s_i \in T$. The persistence event for T is defined as follows:

$$\Diamond\Box T = \overline{\Box\Diamond(S \setminus T)}.$$

Lemma 2.4 (Measurability of infinitely often events and persistence events)**.** *Let $\mathcal{M}$ be an MC with state space S, $s \in S$, and $T \subseteq S$ be a subset of states of $\mathcal{M}$. The events $\Box\Diamond T$ and $\Diamond\Box T$ are measurable.*

Proof: Let

$$Paths_{fin}^{m,T}(s) = \{s_0 \dots s_m \in Paths_{fin}(s) \mid s_m \in T\}, \text{ and}$$
$$Cyl^{m,T}(s) = \bigcup_{s_0 \dots s_m \in Paths_{fin}^{m,T}(s)} Cyl(s_0 \dots s_m)$$

for any $m \in \mathbb{N}$. The infinitely often event for T can be defined as a countable intersections and unions of cylinder sets:

$$\Box\Diamond T = \bigcap_{n \in \mathbb{N}} \bigcup_{m \geq n} Cyl^{m,T}(s).$$

This event is thus measurable, and $\mathbb{P}_s(\Box\Diamond T)$ denotes the probability measure of this event. Since $\Box\Diamond(S \setminus T)$ is measurable, $\Diamond\Box T$ is measurable, with

$$\mathbb{P}_s(\Diamond\Box T) = 1 - \mathbb{P}_s(\Box\Diamond(S \setminus T)).$$

□

We now provide a way to compute the probability of these events with graph theory algorithms. To do that, we need to introduce some graph concepts.

Definition 2.5 (Bottom strongly connected components). Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC and $T \subseteq S$ be a subset of states of $\mathcal{M}$.

- T is *strongly connected* if for any $s, s' \in T$, s is connected to s' in the underlying graph of $\mathcal{M}$, i.e., if there exists a finite path $s_0 \dots s_k \in Paths_{fin}(s)$ from s to s' in $\mathcal{M}$ such that $s_k = s'$.

- T is a *strongly connected component* of $\mathcal{M}$ (SCC, for short) iff T is strongly connected and no proper superset of T is strongly connected, i.e.,

$$T \text{ is strongly connected } \wedge \neg(\exists T',\ T' \text{ is strongly connected } \wedge\ T \subset T').$$

- T is a *bottom strongly connected component* of $\mathcal{M}$ (BSCC, for short) iff T is a SCC and no state outside T can be reached, i.e.,

$$T \text{ is an SCC } \wedge\ \forall s \in T,\ \Delta(s, T) = 1.$$

We denote by $\mathsf{BSCC}(\mathcal{M})$ the set of BSCCs of $\mathcal{M}$.

SCCs and BSCCs of any Markov chain can be computed with purely graph theory algorithms (e.g., with depth-first search based algorithms).

Example 2.3 (*Difference between SCCs and BSCCs*). Let $\mathcal{M} = (S, \Delta)$ be the MC of Figure 2.4.

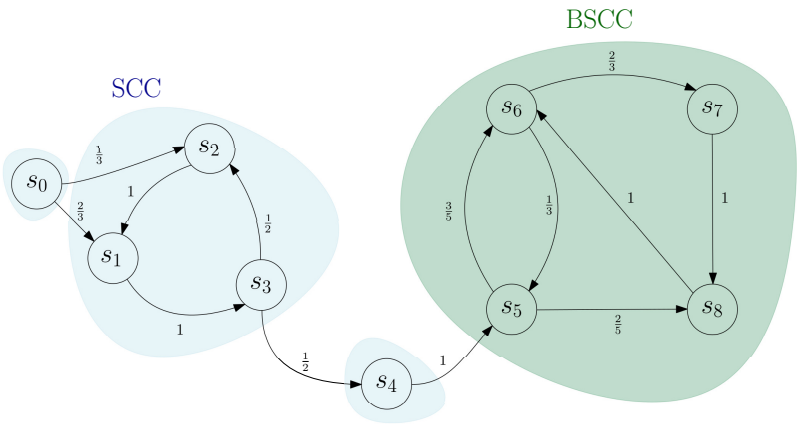


Figure 2.4. Markov chain $\mathcal{M}$ with state space composed by nine states, containing four SCCs and one BSCC

We have that the subset $T = \{s_1, s_2, s_3\} \subseteq S$ is an SCC because T is the largest subset containing the state s_1 such that all states are connected to each other. However, T is not a BSCC because the state $s_3 \in T$ is connected to s_4 by an edge in the underlying graph of $\mathcal{M}$ (we have $\Delta(s_3, s_4) > 0$), and s_4 is not connected to T. Furthermore, the subset $B = \{s_5, s_6, s_7, s_8\} \subseteq S$ is a BSCC. Indeed, B is the largest subset of S containing s_5 such that all states are connected to each other. Moreover, each state $s \in B$ verify the following assertion: $\Delta(s, B) = 1$.

Theorem 2.3 (*Limit behaviour of Markov chains*). *Let $\mathcal{M}$ be an MC with state space S and $s \in S$ be a state of $\mathcal{M}$,*

$$\mathbb{P}_s(\{\pi \in Paths(s) \mid inf(\pi) \in \mathsf{BSCC}(\mathcal{M})\}) = 1,$$

where $inf(\pi)$ denotes the set of states encountered infinitely often along $\pi = s_0 s_1 s_2 \cdots \in Paths(s)$, i.e., $inf(\pi) = \{s \in S \mid \forall n \in \mathbb{N}, \exists m \geq n, s_m = s\}$.

Theorem 2.3 means that, starting from any state of an MC, the system ends up in a BSCC with probability one, and all states of this BSCC are visited infinitely often.

Corollary 2.1 (*Quantitative repeated reachability*). *Let $\mathcal{M}$ be a finite MC with state space S, $T \subseteq S$ be a subset of states of $\mathcal{M}$, and $s \in S$ be a state of $\mathcal{M}$. Then, we can compute the probability to encounter infinitely often T as follows:*

$$\mathbb{P}_s(\square\lozenge T) = \mathbb{P}_s(\lozenge U),$$

where U is the union of all BSCCs B of $\mathcal{M}$ such that $B \cap T \neq \emptyset$. Furthermore, we can compute the probability that T is persistent as follows:

$$\mathbb{P}_s(\lozenge\square T) = \mathbb{P}_s(\lozenge U),$$

where U is the union of all BSCCs B of $\mathcal{M}$ such that $B \subseteq T$.

Proof: Let $t \in T$ be a target state.

1. Let $B \in \mathsf{BSCC}(\mathcal{M})$ be a BSCC of $\mathcal{M}$. In particular, assume there exists a state $t \in T$ such that $t \in B$, then $\mathbb{P}_s(\lozenge B) = \mathbb{P}_s(\{\pi \in Paths(s) \mid t \in inf(\pi) \wedge inf(\pi) = B\})$. Following the definition of BSCCs, we have that B is the largest subset containing t such that all states of B are connected to each other and such that $\forall s' \in B$, $\Delta(s', B) = 1$. Then, B is the only BSCC containing t. So, $\mathbb{P}_s(\lozenge B) = \mathbb{P}_s(\{\pi \in Paths(s) \mid t \in inf(\pi)\}) = \mathbb{P}_s(\square\lozenge\{t\})$, by definition of $inf(\pi)$ and $\square\lozenge\{t\}$.

2. Furthermore, assume that for all BSCC B, $t \notin B$. As we have

   $$\mathbb{P}_s(\{\pi \in Paths(s) \mid t \notin inf(\pi), inf(\pi) \in \mathsf{BSCC}(\mathcal{M})\}) = 1,$$

   we obviously have $\mathbb{P}_s(\{\pi \in Paths(s) \mid t \in inf(\pi)\}) = 0$, i.e., t is never visited infinitely often with a positive probability, and $\mathbb{P}_s(\square\lozenge\{t\}) = 0$.

By 1, we have that if t is in a BSCC B, then the BSCC B is the only one containing t and the probability to reach B is equals to the probability to encounter infinitely often t. Furthermore, by 2, we have that if t is encountered infinitely often, then t is necessarily in a BSCC. Thus, we can generalise 1 and 2 for all states in the set T:

$$\mathbb{P}_s(\square\lozenge T) = \mathbb{P}_s(\lozenge \bigcup_{B \in \mathsf{BSCC}(\mathcal{M}) \mid \exists t \in T, t \in B} B)$$

The second assertion can be shown in a similar way. □

Now, we come back on a problem addressed previously in Remark 2.1. Indeed, following an MC $\mathcal{M}$ with state space S and two subsets $C, T \subseteq S$, we have presented a graph theory based solution to compute the largest subset $S_{=0}$, i.e., $\{s \in S \mid \mathbb{P}_s(C \,\mathsf{U}\, T) = 0\}$, but we have not yet presented a solution to compute the largest subset $S_{=1}$, i.e., $\{s \in S \mid \mathbb{P}_s(C \,\mathsf{U}\, T) = 1\}$. We will see that this qualitative property can be verified thanks to the notion of BSCC.

Notation 2.2 (*Absorbing state*). Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC and $s \in S$ be a state of $\mathcal{M}$. We say that s is absorbing iff $\Delta(s, s') = 1$.

Theorem 2.4 (*Almost sure reachability*). *Let $\mathcal{M}$ be a finite MC with state space S, $s \in S$, $T \subseteq S$ be a set of absorbing states, and the two following successors and predecessors sets:*

$$Succ^*(s) = \{s' \in S \mid \exists \pi = s_0 s_1 s_2 \cdots \in Paths(s) \;\; \exists k \in \mathbb{N}_0,\, s_k = s'\},$$
$$Pred^*(T) = \{s' \in S \mid \exists \pi = s_0 s_1 s_2 \cdots \in Paths(s') \; \exists k \in \mathbb{N}_0,\, s_k \in T\}.$$

Then , the following statements are equivalent:

(a.) $\mathbb{P}_s(\Diamond T) = 1$.

(b.) $Succ^*(s') \cap T \neq \emptyset,\ \forall s' \in Succ^*(s)$.

(c.) $s \in S \setminus Pred^*(S \setminus Pred^*(T))$.

In particular,

$$\{s \in S \mid \mathbb{P}_s(\Diamond T) = 1\} = S \setminus Pred^*(S \setminus Pred^*(T))$$

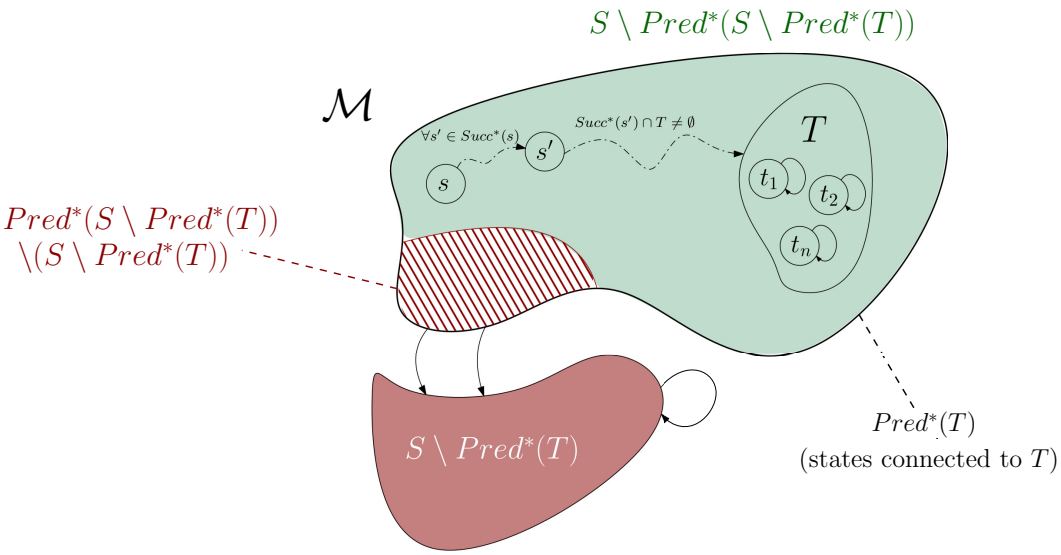


Figure 2.5. Intuitive representation of the subset $S \setminus Pred^*(S \setminus Pred^*(T))$

Proof: [BK08] We can easily derivate from the definitions of the sets $Succ^*$ and $Pred^*$ that (b.) and (c.) are equivalent.
$(\neg(b.) \implies \neg(a.))$. Let $s' \in Succ^*(s)$ such that $Succ^*(s') \cap T = \emptyset$. In Figure 2.5, we have that s is either in the subset represented by a red tiling pattern or in the red subset, and that s' is in the red subset. The sets $Succ^*(s)$ and $Succ^*(s')$ associated with these two possible situations are depicted in Figure 2.6. Thus, s' is a successor of s from which T cannot be reached. Then, we have $\mathbb{P}_s(\Diamond T) \leq 1 - \mathbb{P}_s(\Diamond\{s'\}) < 1$.

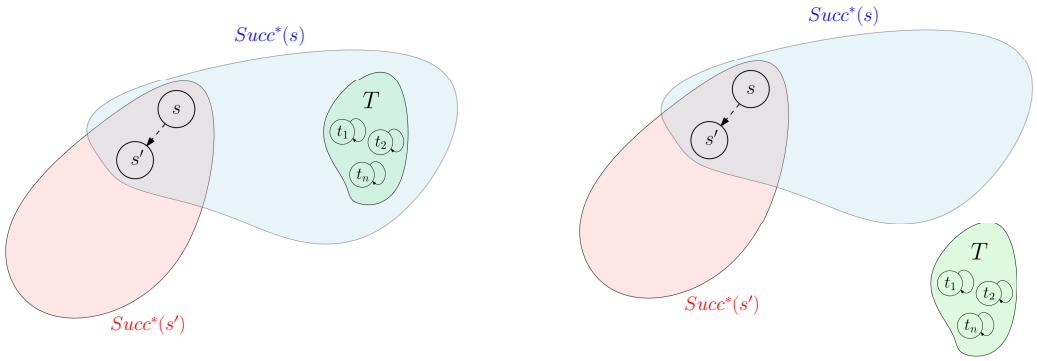


Figure 2.6. Intuitive representation of the sets $Succ^*(s)$ and $Succ^*(s')$ in the two possible situations

$((b.) \implies (a.))$. Assume that $Succ^*(s') \cap T \neq \emptyset$, for any successor s' of s. By Theorem 2.3, we reach a BSCC from s with probability one. Since each state in $t \in T$ is absorbing, each BSCC B of $\mathcal{M}$ consists in $B = \{t\}$ or it satisfies $T \cap B = \emptyset$. We will show that the latter case can not occur from s. Consider $T \cap B = \emptyset$ for some BSCC B. As B is a BSCC, $Succ^*(u) \cap T = \emptyset$, for each $u \in B$. However, as T is reachable for any $s' \in Succ^*(s)$, there is no BSCC with $T \cap B = \emptyset$ that is reachable from s. So, we almost surely have a state in T that will be reached from the state s. □

Theorem 2.4 allows to compute the largest subset $S_{=1} = \{s \in S \mid \mathbb{P}_s(\Diamond T) = 1\}$ for any finite MC with state space S and for any $T \subseteq S$ as follows:

Algorithm 2 Almost sure reachability

Input : a finite Markov chain $\mathcal{M}$ with state space S, a transition function Δ, and a set of target states $T \subseteq S$.
Output : the set $\{s \in S \mid \mathbb{P}_s(\Diamond T) = 1\}$.

1: $\Delta_T \leftarrow \lambda(s, s') : \begin{cases} 1 & \text{if } s = s' \text{ and } s \in T \\ \Delta(s, s') & \text{else} \end{cases}$
2: $\mathcal{M}_T \leftarrow (S, \Delta_T)$ *(* make all states of T absorbing, yielding a new MC $\mathcal{M}_T$ *)*
3: $Pre^*(T) \leftarrow$ backward_search$(G^{\mathcal{M}_T}, T)$ *(* compute the set of states connected to T *)*
4: $Pre^*(S \setminus Pre^*(T)) \leftarrow$ backward_search$(G^{\mathcal{M}_T}, S \setminus Pre^*(T))$
5: **return** $S \setminus Pre^*(S \setminus Pre^*(T))$

where the notation $\lambda x : y$ is a lambda calculus based notation that gives the lambda calculus expression $\lambda x.y \equiv x \mapsto y$, $G^{\mathcal{M}}$ denotes the underlying graph of $\mathcal{M}$ and

the backward_search algorithm , for inputs $G^{\mathcal{M}_T}$ and T, explores $G^{\mathcal{M}_T}$ from the set T: it starts by marking all states of T, and then iteratively marks all unmarked predecessors of marked states (*backward breadth-first search*). All marked states are thus connected to T. An alternative recursive version of this algorithm exists (*backward depth-first search*). The time complexity of this algorithm is in $\mathcal{O}(|\mathcal{M}_T|)$.

Corollary 2.2 (*Qualitative constrained reachability*). *Let $\mathcal{M}$ be an MC with state space S and $C, T \subseteq S$, the sets*

$$S_{=0} = \{s \in S \mid \mathbb{P}_s(C \,\mathsf{U}\, T) = 0\} \text{ and } S_{=1} = \{s \in S \mid \mathbb{P}_s(C \,\mathsf{U}\, T) = 1\}$$

can be computed in time $\mathcal{O}(|\mathcal{M}|)$.

Proof: [BK08]

1. We can determine $S_{=0} = \{s \in S \mid \mathbb{P}_s(C \,\mathsf{U}\, T) = 0\}$ by computing the complement of the set $\{s \in S \mid \exists \pi \in Paths(\pi),\, \pi \models C \,\mathsf{U}\, T\}$ (cf. Lemma 2.3).

2. A linear-time algorithm for the computation of $S_{=1} = \{s \in S \mid \mathbb{P}_s(C \,\mathsf{U}\, T) = 1\}$ can be obtained by a reduction to the reachability problem to T in a slightly modified MC. The idea is the following: we make all states of T absorbing (to take advantage of Theorem 2.4) and do the same for all states in $S \setminus (C \cup T)$. The probability transition function of this modified MC $\mathcal{M}'$ is defined as follows:

$$\Delta'(s, s') = \begin{cases} 1 & \text{if } s = s' \text{ and } s' \in T \cup (S \setminus (C \cup T)), \\ 0 & \text{if } s \neq s' \text{ and } s \in T \cup (S \setminus (C \cup T)), \text{ and} \\ \Delta(s, s') & \text{otherwise.} \end{cases}$$

   Thus, we have:

   - $\mathbb{P}_s^{\mathcal{M}}(C \,\mathsf{U}\, T) = \mathbb{P}_s^{\mathcal{M}'}(\Diamond T)$ for all states $s \in C \setminus T$,
   - $\mathbb{P}_s^{\mathcal{M}}(C \,\mathsf{U}\, T) = \mathbb{P}_s^{\mathcal{M}'}(\Diamond T) = 1$ for all states $s \in T$, and
   - $\mathbb{P}_s^{\mathcal{M}}(C \,\mathsf{U}\, T) = \mathbb{P}_s^{\mathcal{M}'}(\Diamond T) = 0$ for all states $s \in S \setminus (C \cup T)$.

   Doing this reduction, we can apply Theorem 2.4 to compute the largest set $S_{=1}$ with Algorithm 2.

□

Thus, check if each temporal event presented in this section holds with probability one or zero can be done through simple graph theory algorithms in $\mathcal{O}(|\mathcal{M}|)$, without referring to probabilities of transitions.

2.1.3 Temporal events in Markov decision processes

We present in this subsection a way to deal with temporal events in MDPs and furthermore compute their probability. In the first chapter, we have approached a method allowing to compute the optimal strategy σ for the SR problem, i.e., compute the strategy σ following an MDP $\mathcal{M}$ with state space S, a state $s \in S$, and a subset of target states T, such that $\mathbb{P}_s^{\sigma}(\Diamond T) = \max_{\sigma} \mathbb{P}_s^{\sigma}(\Diamond T) = \mathbb{P}_s^{\max}(\Diamond T)$. We have seen that this can be done in polynomial time in the size of $\mathcal{M}$, through a linear program (cf. Theorem 1.5 and Appendix A.3.1). We can generalise this computation of optimal strategy for the constrained reachability events. To this purpose, we need to address the equation system from which the linear program used to compute this strategy is derived (cf. Appendix A.3.1).

Definition 2.6 (**Connectivity to a subset of target states**). Let $\mathcal{M}$ be an MDP with state space S , $s \in S$, and $T \subseteq S$. The state s is said to be *connected to* T if and only if there exists a path $\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \cdots \in Paths(s)$ such that there exists a state $s_k \in T$ for $k \in \mathbb{N}$.

Theorem 2.5 (***Bellman's equation system for optimal reachability probabilities***). *Let $\mathcal{M}$ be an MDP with state space S and with a probability transition function Δ, $s \in S$ be a state of $\mathcal{M}$ and $T \subseteq S$ be a subset of target states. The vector $(x_s)_{s \in S}$ with $x_s = \mathbb{P}_s^{\max}(\Diamond T)$ yields the unique solution of the following equation system: let the subset $S_{=1} \subseteq S$ be a subset of states such that $T \subseteq S_{=1} \subseteq \{s \in S \mid \mathbb{P}_s^{\max}(\Diamond T) = 1\}$,*

(1.) if $s \in S_{=1}$, then $x_s = 1$,

(2.) else if s is not connected to T, then $x_s = 0$,

(3.) else,

$$x_s = \max_{\alpha \in A(s)} \sum_{s' \in Succ(s,\alpha)} \Delta(s, \alpha, s') \cdot x_{s'}.$$

This theorem suggests an iterative approximation technique, called *value iteration*, to compute the values $x_s = \mathbb{P}_s^{\max}(\Diamond T)$ for $s \in S$. First, we assume $x_s^{(n)}$ being equal to x_s for all $n \in \mathbb{N}$, for each state $s \in S$ verifying conditions (1.) and (2.) determined by a backward reachability analysis of the underlying graph of $\mathcal{M}$. For other states s, we have

$$x_s = \lim_{n \to \infty} x_s^{(n)}$$

where

$$x_s^{(0)} = 0 \quad \text{and} \quad x_s^{(n+1)} = \max_{\alpha \in A(s)} \sum_{s' \in Succ(s,\alpha)} \Delta(s, \alpha, s') \cdot x_{s'}^{(n)}.$$

This yields $x_s^{(n)} \leq x_s^{(n+1)}$ for any $n \in \mathbb{N}$, and the values of $\mathbb{P}_s^{\max}(\Diamond T)$ can be approximated by iteratively computing vectors $x_s^{(n+1)}$ with vectors $x_s^{(n)}$ for all states $s \in S$ until $\max_{s \in S} |x_s^{(n+1)} - x_s^{(n)}| < \varepsilon$ for a small threshold value $\varepsilon > 0$.

Theorem 2.6 (*Value iteration for step-bounded reachability*). *Let $\mathcal{M}$ be an MDP with state space S and with a probability transition function Δ, and $T \subseteq S$ be a subset of target states. The value iteration approach can be used to compute the maximal probabilities for the event $\Diamond^{\leq n} T$. Indeed, the vector $(x_s)_{s \in S}^{(n)}$ with $x_s^{(n)} = \mathbb{P}_s^{\max}(\Diamond^{\leq n} T)$ yields the unique solution of the following equation system:*

- *if $s \in T$, then $x_s^{(n)} = 1$ for all $n \in \mathbb{N}$,*
- *else if s is not connected to T, then $x_s^{(n)} = 0$ for all $n \in \mathbb{N}$,*
- *else, $x_s^{(0)} = 0$ and*

$$x_s^{(n+1)} = \max_{\alpha \in A(s)} \sum_{s' \in Succ(s,\alpha)} \Delta(s, \alpha, s') \cdot x_{s'}^{(n)}.$$

According to the equation system defined in Theorem 2.6, we can build a finite memory strategy $\sigma = (Q, \sigma_{act}, \delta, \delta_0)$ such that $\mathbb{P}_s^{\sigma}(\Diamond^{\leq n} T) = \mathbb{P}_s^{\max}(\Diamond^{\leq n} T)$ for $s \in S$ and $n \in \mathbb{N}$: assume that modes of Q range from 0 to n such that $\delta_0(s) = m_0$ and $\delta(m_i, s) = m_{\min\{i+1,n\}}$ for all $s \in S$ and $i \in \{0, \ldots, n\}$. Then, we define σ_{act} as follows:

$$\sigma_{act}(m_i, s) = \begin{cases} \alpha \text{ such that } \alpha \in A(s) & \text{if } i = n, \\ \arg\max_{\alpha \in A(s)} \sum_{s' \in Succ(s,\alpha)} \Delta(s, \alpha, s') \cdot x_{s'}^{(i)} & \text{else.} \end{cases}$$

Intuitively, σ is initialised in mode m_0, and when the strategy is in mode m_i for $i \in \{0, \ldots, n-1\}$, it chooses the action that maximises $\sum_{s' \in Succ(s,\alpha)} \Delta(s, \alpha, s') \cdot x_{s'}^{(i)}$ and then go to the next mode m_{i+1}. When the strategy enters in mode m_n, it stays inside it forever and always chooses an arbitrary enabled action.

Theorem 2.7 (*Strategy for constrained reachability*). *Let $\mathcal{M}$ be an MDP with state space S, $s \in S$, $C, T \subseteq S$, and $n \in \mathbb{N}$. We can build a strategy σ and a strategy σ' respectively satisfying $\mathbb{P}_s^{\sigma}(C \,\mathsf{U}\, T) = \max_{\sigma} \mathbb{P}_s^{\sigma}(C \,\mathsf{U}\, T)$ and $\mathbb{P}_s^{\sigma'}(C \,\mathsf{U}^{\leq n}\, T) = \max_{\sigma} \mathbb{P}_s^{\sigma}(C \,\mathsf{U}^{\leq n}\, T)$ in polynomial time in the size of $\mathcal{M}$.*

Proof: We can compute $\mathbb{P}_s^{\max}(C \,\mathsf{U}\, T)$ (or $\mathbb{P}_s^{\max}(C \,\mathsf{U}^{\leq n}\, T)$) for any state $s \in S$ of the system by reduction to the reachability to T as follows: we make all states $s^* \in S \setminus (C \cup T)$ absorbing, i.e., we replace all enabled actions in $A(s^*)$ by a unique action α_{s^*} such that $\Delta(s^*, \alpha_{s^*}, s^*) = 1$. Then, we build a strategy σ such that $\mathbb{P}_s^{\sigma}(\Diamond T) = \mathbb{P}_s^{\max}(\Diamond T)$ (or $\mathbb{P}_s^{\sigma}(\Diamond T) = \mathbb{P}_s^{\max}(\Diamond^{\leq n} T)$) in this modified MDP. For $\mathbb{P}_s^{\max}(\Diamond T)$, this strategy corresponds to the optimal strategy built for the SR problem for the state s, the subset of target states T, and an arbitrary probability threshold (cf. Appendix A.3.1 for more details on the computation of this maximum probability). □

In order to optimise computations and the size of the LP defined to determine $\mathbb{P}_s^{\max}(\Diamond T)$ for all $s \in S$, it is possible to efficiently compute the largest set $S_{=1} = \{s \in S \mid \mathbb{P}_s^{\max}(\Diamond T) = 1\}$.

Theorem 2.8 (*Qualitative reachability in MDPs*). *Let $\mathcal{M}$ be an MDP with state space S, and $T \subseteq S$. The set*

$$S_{=1} = \{s \in S \mid \mathbb{P}_s^{\max}(\Diamond T) = 1\}$$

can be computed in time $\mathcal{O}(|\mathcal{M}|^2)$.

The set of states almost surely reaching T under the control of an optimal strategy can be computed with Algorithm 3.

Algorithm 3 Build the largest subset $S_{=1}$

Input : $\mathcal{M} = (S, A, \Delta, w, AP, L)$, an MDP, and $T \subseteq S$, a subset of target states.
Output : The set of states s of $\mathcal{M}$ for which $\mathbb{P}_s^{\max}(\Diamond T) = 1$.

1: $U \leftarrow \{s \in S \mid s$ is not connected to $T\}$ *(* U is the set of "bad states" *)*
2: **while** $U \neq \emptyset$ **do**
3: $R \leftarrow U$
4: **while** $R \neq \emptyset$ **do**
5: Let $u \in R$ *(* let u be a bad state *)*
6: $R \leftarrow R \setminus \{u\}$ *(* mark u as visited *)*
7: **for all** $(s, \alpha) \in Pred(u)$ such that $s \notin U \cup T$ **do**
8: remove α from $A(s)$ *(* remove all ingoing edges of u *)*
9: **if** $A(s) = \emptyset$ **then**
10: $R \leftarrow R \cup \{s\}$ *(* for any enabled action, s is always connected to a bad state *)*
11: $U \leftarrow U \cup \{s\}$ *(* thus, s is marked as bad state *)*
12: remove u and its outgoing edges in $\mathcal{M}$
13: $U \leftarrow \{s \in S \setminus U \mid s$ is not connected to $T\}$ *(* update the set of bad states *)*
14: **return** the remaining states

In order to compute this set of states, we consider $\mathcal{M}$ as being a directed graph. This directed graph is a refinement of its underlying graph, where the successors of any vertex $s \in S$ are new dummy vertices being the pairs (s, α), with $\alpha \in A(s)$. The outgoing edges of these dummy vertices (s, α) are α-successors of s. The key idea is looking at all "bad states", unconnected to the set of target states T (cf. line 1), and iteratively remove all dummy vertices of this modified graph having an outgoing edge to these bad states (cf. lines 7 and 8). That is, we remove these dummy vertices as well as their ingoing and outgoing edges. If a vertex representing a state s has no more outgoing edge to a dummy state, then whatever the enabled action chosen in the MDP by any strategy at state s, the system could in the worst case never reach T (i.e., the system do not almost surely reach T from s). In that case, such a state is tagged as being a bad state (cf. lines 9 and 11), and we repeat the operation until all the bad states have been considered. After that, we compute the set of states unconnected to T again and mark them as bad states (cf. line 13). Indeed, deleting dummy states could leave some bad states in the graph (e.g., an SCC unconnected to T could remain). We repeat all the operations since the beginning for these new bad states until no more bad states are found in the last step of the algorithm. The states unmarked as bad states are finally returned.

This algorithm is exact (cf. [BK08] for the correctness proof) and is quadratic in the size of $\mathcal{M}$. Indeed, this can be seen easily: the maximal number of iterations of the outermost loop (cf. line 2) is $|S|$ (a state can be marked as bad at most once). Then, in each iteration of this outermost loop, the set of states unconnected to T is computed in time $\mathcal{O}(|\mathcal{M}|)$ by backward search from T in the modified directed graph. Finally, the rest of operations cause in worst case time in $\mathcal{O}(|\mathcal{M}|)$, since states can be marked and dummy states can be removed at most once.

2.2 Probabilistic computation tree logic

Probabilistic computation tree logic (PCTL, for short) is a *branching-time temporal logic*. This logic allows to verify probabilistic systems via an execution tree, actually named *computation tree*. In this section, we describe this logic for Markov chains.

For a given system, the computation tree consists of an infinite unfolding of this system starting from a given state (i.e., the root of the tree), considering all branching possibilities. The key ideas behind this tree are that each node of the tree corresponds to a state of the system, and each prefix of the tree corresponds to a finite path of the system starting from the root of the tree. Moreover, each branch passing by a node of the tree corresponds to a path of the system having the prefix leading the root to the node in the tree in its set of prefixes. So, in Markov chains context, this tree is highly linked to cylinder sets: all branches of the tree passing by a node corresponds to the cylinder set spanned with the prefix leading the root to the node in the tree.

Example 2.4 (*Computation tree of an MC*). Let $\mathcal{M}$ be the MC with state space S and labelling function L of Figure 2.7. The computation tree of $\mathcal{M}$ starting from the state s_0 is given in Figure 2.8. We clearly see that any branch passing by a node s^* in this tree actually corresponds to a path $\pi = s_0 \dots s_k \dots \in Cyl(s_0 \dots s_k)$ such that $s_k = s^*$. For example, the path $\pi = s_0 s_0 (s_1 s_2)^\omega \in Cyl(s_0 s_0)$ is a branch of the tree passing by the node s_0 having $s_0 s_0$ as prefix. We will see that it is possible to express formulae in PCTL describing states properties of the system like the following one:

"*Do all executions starting from the state s_0 always eventually reach b with a nonzero probability ?*"

Model checking algorithms of PCTL will answer *yes* to this question. Intuitively, if we refer to the computation tree, we see that a node labelled with b appears in the *future* of any node s^* in the tree: if there exists an index n for each path $\pi = s'_0 s'_1 s'_2 \dots s'_n \dots \in Cyl(s'_0 \dots s'_k)$, where $s'_0 \dots s'_k$ is the prefix that leads the root to the node $s^* = s'_k$ in the tree, such that $k < n$ and s'_n is labelled with b, then the property is verified. This is true for all paths, except for the path s_0^ω (corresponding to the left path in the computation tree). Since this path has a zero probability, we have that this property is verified with a probability one from the state s_0. We will see in this chapter how to verify formally each property of this type by referring to Section 2.1. Indeed, for this example, the question can be reformulated as follows: $\mathbb{P}_{s_0}(\Box\Diamond\{s \in S \mid b \in L(s)\}) > 0$.

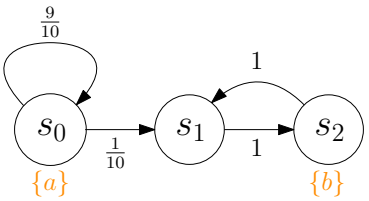


Figure 2.7. MC $\mathcal{M}$ with 3 states, s_0, s_1 and s_2, and 2 atomic propositions, a and b

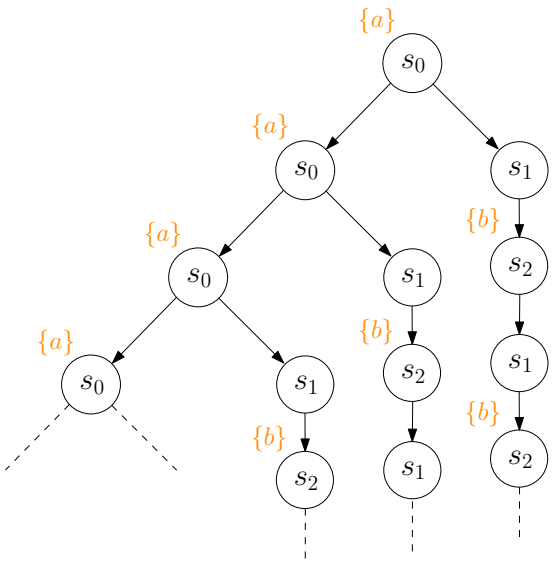


Figure 2.8. Computation tree of $\mathcal{M}$ starting from s_0

Syntax and semantic

PCTL has a two stages syntax where PCTL formulae are classified into state and path formulae. Intuitively, *state formulae* are assertions about atomic propositions in a state s and about probabilities over their branching structure, i.e., probabilities of *path formulae* starting from s. Actually, a path formula imposes conditions on a set of paths and this path formula is quantified by probability bounds.

Definition 2.7 (**Syntax of PCTL**). Let AP be a set of atomic propositions,

- PCTL *state formulae* over AP are formed according to the following grammar:

$$\Phi ::= true \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \mathcal{P}_J(\phi)$$

  where $a \in AP$ is an atomic proposition, $J \subseteq [0,1] \cap \mathbb{Q}$ is an interval with rational bounds and ϕ is a path formula.

- PCTL *path formulae* over AP are formed according to the following grammar:

$$\phi ::= \bigcirc\Phi \mid \Phi_1 \,\mathsf{U}\, \Phi_2 \mid \Phi_1 \,\mathsf{U}^{\leq n} \Phi_2$$

  where Φ, Φ_1 and Φ_2 are state formulae and $n \in \mathbb{N}$.

Intuitively, $\mathcal{P}_J(\phi)$ specifies that the probability of paths satisfying the path formula ϕ must be in the interval J. A path formula is formed by temporal operators, like $\bigcirc$ and U, with $\mathsf{U}^{\leq n}$ being U bounded by a maximum number of steps. There exist some other linear temporal operators, dealing with paths of the system (cf. Figure 2.9). These operators can be derived from the PCTL grammar: let $J \in [0,1]$ be an interval, Φ be a state formula and $n \in \mathbb{N}$ be a number of steps,

- $\mathcal{P}_J(\Diamond\Phi) \equiv \mathcal{P}_J(true \,\mathsf{U}\, \Phi)$ (*eventually* probability)

- $\mathcal{P}_J(\Diamond^{\leq n}\Phi) \equiv \mathcal{P}_J(true\, \mathsf{U}^{\leq n}\Phi)$ (*bounded enventually* probability)
- $\mathcal{P}_J(\Box\Phi) \equiv \neg\mathcal{P}_J(\Diamond\neg\Phi)$ (*always* probability)
- $\mathcal{P}_J(\Box^{\leq n}\Phi) \equiv \neg\mathcal{P}_J(\Diamond^{\leq n}\neg\Phi)$ (*bounded always* probability)

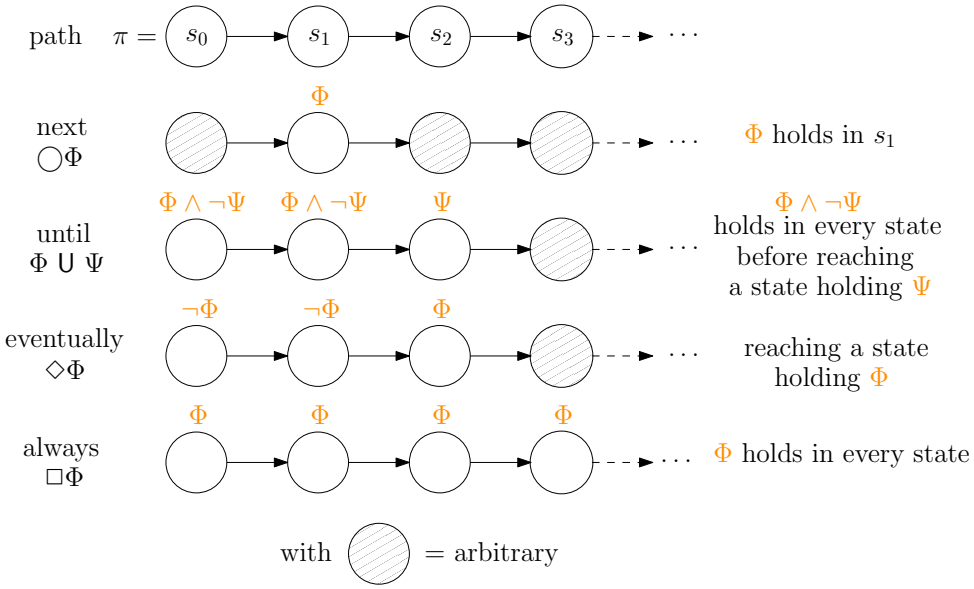


Figure 2.9. Intuitive semantic of linear temporal operators

Definition 2.8 (Semantic of PCTL). Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC, $s \in S$, Φ, Ψ be PCTL state formulae over AP, and ϕ be a PCTL path formula over AP,

$s \models \Phi$ iff the state formula Φ holds in the state s, i.e.,

- $s \models true$,
- $s \models a$ iff a is a label of s, i.e., $a \in L(s)$,
- $s \models \Phi \wedge \Psi$ iff $s \models \Phi$ and $s \models \Psi$,
- $s \models \neg\Phi$ iff $s \not\models \Phi$,
- $s \models \mathcal{P}_J(\phi)$ iff $\mathbb{P}_s(\{\pi \in Paths(s) \mid \pi \models \phi\}) \in J$.

Following a path $\pi = s_0 s_1 s_2 \cdots \in Paths(s)$, $\pi \models \phi$ iff π satisfies the path formula ϕ, i.e.,

- $\pi \models \bigcirc\Phi$ iff $s_1 \models \Phi$,
- $\pi \models \Phi\ \mathsf{U}\ \Psi$ iff $\exists j \in \mathbb{N}$, $s_j \models \Psi$ and $\forall i \in \mathbb{N}$, $i < j$, $s_i \models \Phi$,
- $\pi \models \Phi\ \mathsf{U}^{\leq n}\Psi$ iff $\exists j \in \mathbb{N}$, $j \leq n$, $s_j \models \Psi$ and $\forall i \in \mathbb{N}$, $i < j$, $s_i \models \Phi$.

Additionally,

- $\pi \models \Diamond\Phi$ iff $\exists j \in \mathbb{N}$, $s_j \models \Phi$,
- $\pi \models \Box\Phi$ iff $\forall j \in \mathbb{N}$, $s_j \models \Phi$.

The event $\{\pi \in Paths(s) \mid \pi \models \phi\}$ must be measurable to verify that $\mathcal{P}_J(\phi)$ holds in any state. In fact, these events can be formed through countable unions of cylinder sets, ensuring their measurability.

Theorem 2.9 (*Measurability of PCTL path formula events*). *Let $\mathcal{M}$ be an MDP with state space S, $s \in S$ be a state of $\mathcal{M}$ and ϕ be a PCTL path formula, the set $\{\pi \in Paths(s) \mid \pi \models \phi\}$ is a measurable event.*

Proof: Let $Paths(s, \phi) = \{\pi \in Paths(s) \mid \pi \models \phi\}$.

1. If $\phi = \bigcirc\Phi$, then $Paths(s, \phi)$ agrees with the union of the cylinders sets $Cyl(ss')$, where $s' \in Succ(s)$ and $s' \models \Phi$.

2. Else, if $\phi = \Phi_1 \mathsf{U} \Phi_2$, then the definition of $\pi \models \phi$ for any $\pi \in Paths(s)$ agrees with the definition of $\pi \models Sat(\Phi_1) \mathsf{U} Sat(\Phi_2)$. Thus, $Paths(s, \phi) = Sat(\Phi_1) \mathsf{U} Sat(\Phi_2)$.

3. Else, if $\phi = \Phi_1 \mathsf{U}^{\leq n} \Phi_2$ for any $n \in \mathbb{N}$, then the definition of $\pi \models \phi$ for any $\pi \in Paths(s)$ agrees with the definition of $\pi \models Sat(\Phi_1) \mathsf{U}^{\leq n} Sat(\Phi_2)$. Thus, $Paths(s, \phi) = Sat(\Phi_1) \mathsf{U}^{\leq n} Sat(\Phi_2)$.

As countable union of cylinder sets and constrained reachability events are measurable (cf. Lemma 2.1), $Paths(s, \phi)$ is measurable, for any path formula ϕ.

□

Remark 2.4 (*Probabilistic satisfiability of path formulae*).

- Assume there exists a state s in an MC such that $s \models \mathcal{P}_{=1}(\phi)$ for a given path formula ϕ. That does not mean that all paths $\pi \in Paths(s)$ satisfy ϕ, i.e.,

$$s \models \mathcal{P}_{=1}(\phi) \implies \forall \pi \in Paths(s),\ \pi \models \phi.$$

  Indeed, consider the MC $\mathcal{M}$ of Figure 2.7 and take the state s_0 of $\mathcal{M}$, the path $s_0^\omega \in Paths(s_0)$ and the path formula $\Diamond b$. We have $s_0 \models \mathcal{P}_{=1}(\Diamond b)$, because $\mathbb{P}_{s_0}(\Diamond\{s_2\}) = 1$, but we have $s_0^\omega \not\models (\Diamond b)$.

- Assume there exists a state s in an MC such that there exists a path $\pi \in Paths(s)$ such that $\pi \models \phi$ for a given path formula ϕ. That does not mean that $s \models \mathcal{P}_{>0}(\phi)$, i.e.,

$$\exists \pi \in Paths(s),\ \pi \models \phi \implies s \models \mathcal{P}_{>0}(\phi).$$

  Indeed, consider the MC $\mathcal{M}$ of Figure 2.7 and take the state s_0, the path $s_0^\omega \in Paths(s_0)$ and the path formula $\Box a$. We have that the path s_0^ω is the only path of $\mathcal{M}$ that verifies $\Box a$ (and thus, $s_0^\omega \models \Box a$). However, we have $\mathbb{P}_s(\{s_0^\omega\}) = 0$. Then, $s_0 \models \mathcal{P}_{=0}(\Box a)$.

Definition 2.9 (Satisfiability set for PCTL). Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC and Φ be a PCTL state formula on AP. The *satisfiability set* for the MC $\mathcal{M}$ is defined as follows:

$$Sat(\Phi) = \{s \in S \mid s \models \Phi\}.$$

Model checking

Definition 2.10 (PCTL model checking). Let $\mathcal{M}$ be an MC with state space S, $s \in S$ be a state of $\mathcal{M}$, and Φ be a PCTL state formula. The model checking problem for $\mathcal{M}$, s and Φ consists in deciding if $s \models \Phi$, i.e., if $s \in Sat(\Phi)$.

Following an MC $\mathcal{M}$, a state s of this MC and a state formula Φ, determine if $s \models \Phi$ can be done through a bottom-up traversal of the *parse tree* of Φ. Indeed, according to the PCTL syntax, we recursively decompose Φ following its sub-formulae. Moreover, each sub-formula of Φ is actually represented by a node of this tree. For each PCTL state formula Φ, we compute its satisfaction set from the node representing the formula, according to each satisfaction set computed from each of its childs and the characterisation of $Sat(\Phi)$.

Property 2.1 (Characterisation of Sat)**.** Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC, Φ, Ψ be PCTL state formulae over AP, and ϕ be a PCTL path formula over AP. The satisfaction set Sat is characterised as follows:

$$\begin{aligned} Sat(true) &= S, \\ Sat(a) &= \{s \in S \mid a \in L(s)\} \text{ for } a \in AP, \\ Sat(\Phi) \wedge Sat(\Psi) &= Sat(\Phi) \cap Sat(\Psi), \\ Sat(\neg\Phi) &= S \setminus Sat(\Phi), \text{ and} \\ Sat(\mathcal{P}_J(\phi)) &= \{s \in S \mid \mathbb{P}_s(Paths(s, \phi)) \in J\} \text{ for } J \subseteq [0,1] \cap \mathbb{Q}, \end{aligned}$$

where $Paths(s, \phi) = \{\pi \in Paths(s) \mid \pi \models \phi\}$. $\mathcal{P}_J(\phi)$ is computed according to the formula ϕ as follows: let $s \in S$ be a state of $\mathcal{M}$,

$$\begin{aligned} \mathbb{P}_s(Paths(s, \bigcirc\Phi)) &= \sum_{s' \in Sat(\Phi)} \Delta(s, s'), \\ \mathbb{P}_s(Paths(s, \Phi \,\mathsf{U}\, \Psi)) &= \mathbb{P}_s(Sat(\Phi) \,\mathsf{U}\, Sat(\Psi)), \text{ and} && \text{(cf. Theorem 2.2)} \\ \mathbb{P}_s(Paths(s, \Phi \,\mathsf{U}^{\leq n} \Psi)) &= \mathbb{P}_s(Sat(\Phi) \,\mathsf{U}^{\leq n} Sat(\Psi)). && \text{(cf. Theorem 2.1)} \end{aligned}$$

Additionally, we can optimise the computation of the satisfiability sets of the following formulae:

$$\begin{aligned} Sat(\mathcal{P}_J(\Diamond\mathcal{P}_{=1}(\Box\mathcal{P}_{=1}(\Diamond a)))) &= \{s \in S \mid \mathbb{P}_s(\Box\Diamond Sat(a)) \in J\}, && (\textit{infinitely often}) \\ Sat(\mathcal{P}_J(\Diamond\mathcal{P}_{=1}(\Box a))) &= \{s \in S \mid \mathbb{P}_s(\Diamond\Box Sat(a)) \in J\}, && (\textit{persistence}) \end{aligned}$$

for $J \subseteq [0,1]$ and $a \in AP$ (these two equalities are shown in [BK08]).

Example 2.5 (*PCTL model checking of an MC via its parse tree*). Let $\mathcal{M} = (S, \Delta, AP, L)$ be the MC of Figure 2.10 and the state formula $\Phi = c \vee \mathcal{P}_{\geq \frac{1}{2}}(a \,\mathsf{U}\, (b \wedge c))$. We are interested to know if $s_0 \models \Phi$. To do that, we build the parse tree of Φ (cf. Figure 2.11). We can compute $Sat(\Phi)$ through a bottom up traversal of this tree: we begin at the leaves of the tree, i.e., by computing $Sat(a)$, $Sat(b)$, and $Sat(c)$. Then, we compute $Sat(b \wedge c)$, that equals $Sat(b) \cap Sat(c)$, according to the characterisation of the satisfaction set. After that, we compute $Sat(\mathcal{P}_{\geq \frac{1}{2}}(a \,\mathsf{U}\, (b \wedge c)))$ by computing $\{s \in S \mid \mathbb{P}_s(Sat(a) \,\mathsf{U}\, Sat(b \wedge c)) \geq \frac{1}{2}\} = \{s \in S \mid \mathbb{P}_s(\{s_0, s_2, s_3\} \,\mathsf{U}\, \{s_1\}) \geq \frac{1}{2}\}$. We

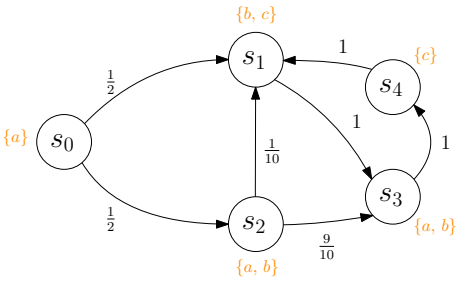


Figure 2.10. MC $\mathcal{M}$ with state space $S = \{s_0, s_1, s_2, s_3, s_4\}$ and atomic propositions of the set $AP = \{a, b, c\}$

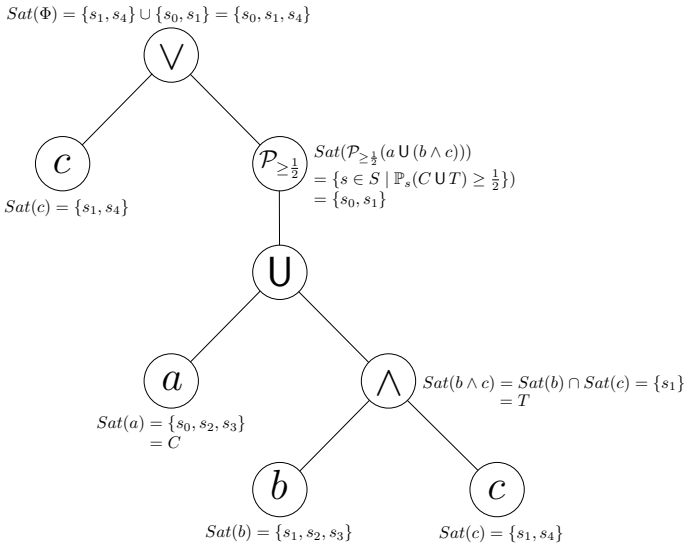


Figure 2.11. Parse tree of $\mathcal{M}$ for Φ

actually have already computed these probabilities (cf. Example 2.2). So, this set equals $\{s_0, s_1\}$. Finally, from the computation of $Sat(c)$ and $Sat(\mathcal{P}_{\geq \frac{1}{2}}(a \,\mathsf{U}\, (b \wedge c)))$, we can compute $Sat(\Phi) = \{s_1, s_4\} \cup \{s_0, s_1\} = \{s_0, s_1, s_4\}$. As $s_0 \in Sat(\Phi)$, then $s_0 \models \Phi$.

2.3 PCTL for Markov decision processes

The PCTL syntax for MDPs is almost the same as for MCs with the exception of the probabilistic operator $\mathcal{P}_J(.)$ due to the nondeterminism of MDPs: we can not verify probabilistic properties without referring to the notion of strategy. Then, we replace this operator with $\mathcal{P}_J^{\max}(.)$, referring to the probability measure defined on paths of the MC induced by the strategy offering the maximum probability of satisfying a given property. Thus, the formula $\mathcal{P}_J^{\max}(\phi)$ asserts

$$\mathbb{P}_s^{\max}(\{\pi \in Paths(s) \mid \pi \models \phi\}) \in J.$$

Note: $\mathbb{P}_s^{\max}(E)$ *refers to the probability measure defined on paths starting from the state* s *in the MC induced by the strategy maximising the probability of the event* E *in this MC, i.e.,* $\max_\sigma \mathbb{P}_s^\sigma(E)$.

Syntax and semantic

Definition 2.11 (**Syntax of PCTL for MDPs**). Let $\mathcal{M}$ be an MDP with space of atomic propositions AP,

- PCTL *state formulae* over AP are formed according to the following grammar:

$$\Phi ::= true \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \mathcal{P}_J^{\max}(\phi)$$

where $a \in AP$ is an atomic proposition, $J \subseteq [0,1] \cap \mathbb{Q}$ is an interval of probability bounds, and ϕ is a path formula.

- PCTL *path formulae* over AP are formed according to the same grammar as for path formulae in Markov chains context.

Definition 2.12 (**Semantic of PCTL for MDPs**)**.** Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP, $s \in S$, Φ, Ψ be PCTL state formulae over AP, and ϕ be a PCTL path formula over AP,

$s \models \Phi$ iff the state formula Φ holds in the state s, i.e.,

- $s \models true$,
- $s \models a$ iff a is a label of s, i.e., $a \in L(s)$,
- $s \models \Phi \wedge \Psi$ iff $s \models \Phi$ and $s \models \Psi$,
- $s \models \neg\Phi$ iff $s \not\models \Phi$,
- $s \models \mathcal{P}_J^{\max}(\phi)$ iff $\mathbb{P}_s^{\max}(\{\pi \in Paths(s) \mid \pi \models \phi\}) \in J$.

$\pi \models \phi$ iff the path π satisfies ϕ in the induced MC.

Model checking

PCTL model checking for MDPs is also done almost the same way as for MCs. Indeed, just the characterisation of the satisfiability set Sat is slightly different, due to the semantic of the probabilistic operator $\mathcal{P}_J^{\max}(.)$.

Property 2.2 (Characterisation of Sat for MDPs)**.** Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP, Φ, Ψ be PCTL state formulae over AP, and ϕ be a PCTL path formula over AP. The satisfaction set Sat is characterised as follows:

$$\begin{aligned} Sat(true) &= S, \\ Sat(a) &= \{s \in S \mid a \in L(s)\} \text{ for } a \in AP, \\ Sat(\Phi) \wedge Sat(\Psi) &= Sat(\Phi) \cap Sat(\Psi), \\ Sat(\neg\Phi) &= S \setminus Sat(\Phi), \text{ and} \\ Sat(\mathcal{P}_J^{\max}(\phi)) &= \{s \in S \mid \mathbb{P}_s^{\max}(Paths(s,\phi)) \in J\} \text{ for } J \subseteq [0,1] \cap \mathbb{Q}, \end{aligned}$$

where $Paths(s,\phi) = \{\pi \in Paths(s) \mid \pi \models \phi\}$. $\mathcal{P}_J^{\max}(\phi)$ is computed according to the formula ϕ as follows: let $s \in S$ be a state of $\mathcal{M}$,

$$\begin{aligned} \mathbb{P}_s^{\max}(Paths(s, \bigcirc\Phi)) &= \max_{\alpha \in A(s)} \sum_{s' \in Sat(\Phi)} \Delta(s,\alpha,s'), \\ \mathbb{P}_s^{\max}(Paths(s, \Phi \,\mathsf{U}\, \Psi)) &= \mathbb{P}_s^{\max}(Sat(\Phi) \,\mathsf{U}\, Sat(\Psi)), \text{ and} \qquad \text{(cf. Theorem 2.7)} \\ \mathbb{P}_s^{\max}(Paths(s, \Phi \,\mathsf{U}^{\leq n} \Psi)) &= \mathbb{P}_s^{\max}(Sat(\Phi) \,\mathsf{U}^{\leq n} Sat(\Psi)). \end{aligned}$$

2.4 Probabilistic computation tree logic with rewards extension

To handle properties involving weights of systems, there is a variant of PCTL incorporating the truncated sum function. This logic is named *probabilistic-reward computation tree logic* (PRCTL, for short). This logic introduces the expectation operator $\mathcal{E}_C(.)$, and additionally includes the *cost-bounded until* operator $\mathsf{U}_{\leq \ell}$.

As for PCTL, we define PRCTL in Markov chains and then describe the slight changes of the syntax and semantic induced by the nondeterminism of Markov decision processes in the following section.

Syntax and semantic

Definition 2.13 (**Syntax of PRCTL**)**.** Let AP be a set of atomic propositions,

- PRCTL *state formulae* are formed according to the following grammar:

$$\Phi ::= true \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \mathcal{P}_J(\phi) \mid \mathcal{E}_C(\Phi)$$

  where $a \in AP$ is an atomic proposition, $J \subseteq [0,1] \cap \mathbb{Q}$ is an interval of rational bounds, $C \subseteq \mathbb{N} \cup \{+\infty\}$ is an interval of cost bounds, and ϕ is a path formula.

- PRCTL *path formulae* are formed according to the following grammar:

$$\phi ::= \bigcirc\Phi \mid \Phi_1 \,\mathsf{U}\, \Phi_2 \mid \Phi_1 \,\mathsf{U}^{\leq n} \Phi_2 \mid \Phi_1 \,\mathsf{U}_{\leq \ell}\, \Phi_2$$

  where Φ, Φ_1 and Φ_2 are state formulae, and $n, \ell \in \mathbb{N}$.

Definition 2.14 (**Semantic of PRCTL**)**.** Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC, $s \in S$, $\pi \in Paths(s)$, and Φ, Ψ be PRCTL state formulae, the semantic of PRCTL is exactly the same as the semantic of PCTL, with the additional expectation operator $\mathcal{E}_C(.)$ and the additional cost bounded operator $\mathsf{U}_{\leq \ell}$, for which the semantic is defined as follows:

$$\begin{aligned} s &\models \mathcal{E}_C(\Phi) && \text{iff} \quad \mathbb{E}_s(\mathsf{TS}^{Sat(\Phi)}) \in C, \\ \pi &\models \Phi \,\mathsf{U}_{\leq \ell}\, \Psi && \text{iff} \quad \exists j \in \mathbb{N},\ s_j \models \Psi,\ \forall i \in \mathbb{N},\ i < j,\ s_i \models \Phi, \text{ and } \mathsf{TS}^{Sat(\Psi)}(\pi) \leq \ell. \end{aligned}$$

In order to ensure the measurability of all events formed by any path formula $\Phi \,\mathsf{U}_{\leq \ell}\, \Psi$ for state formulae Ψ and Φ, and a given cost threshold ℓ, we define the cost-bounded constrained reachability event as follows.

Definition 2.15 (**Cost-bounded constrained reachability event**)**.** Let $\mathcal{M}$ be an MC with state space S, $s \in S$, $C, T \subseteq S$, and $\ell \in \mathbb{N}$, the constrained cost-bounded reachability is defined as follows:

$$C \,\mathsf{U}_{\leq \ell}\, T = \{s_0 s_1 s_2 \cdots \in Paths(s) \mid \exists n \in \mathbb{N},\ s_n \in T \wedge \forall i < n,\ s_i \in C \wedge \mathsf{TS}^T \leq \ell\}.$$

Lemma 2.5 (Measurability of cost-bounded constrained reachability events)**.** *Let $\mathcal{M}$ be an MC with state space S, $s \in S$, $C, T \subseteq S$, $\ell \in \mathbb{N}$, the event $C \,\mathsf{U}_{\leq \ell}\, T$ is measurable.*

Proof: The event $C \,\mathsf{U}_{\leq \ell}\, T$ is measurable since it can be defined as a countable union of cylinder sets:

$$C \,\mathsf{U}_{\leq \ell}\, T = \bigcup_{s_0 \ldots s_n \in Paths^{C,T}_{\leq \ell}(s)} Cyl(s_0 \ldots s_n),$$

where $Paths^{C,T}_{\leq \ell}(s) = \{s_0 \ldots s_n \in Paths_{fin}(s) \mid \forall i < n,\, s_i \in C \setminus T \;\wedge\; s_n \in T \;\wedge\; \mathsf{TS}^T(s_0 \ldots s_n) \leq \ell\}$.

$\mathbb{P}_s(C \,\mathsf{U}_{\leq \ell}\, T)$ denotes the probability of the union of cylinder sets spanned with finite paths of the set $Paths^{C,T}_{\leq \ell}(s)$ starting from the state $s \in S$.

□

Model checking

The PRCTL model checking is also an extension of the PCTL model checking. We just need to describe the changes in the characterisation of Sat including the new notions introduced in PRCTL.

Property 2.3 (PRCTL characterisation of Sat)**.** Let $\mathcal{M}$ be an MC with state space S and atomic proposition space AP, Φ, Ψ be PRCTL state formulae over AP and ϕ be a PRCTL path formula over AP. The satisfaction set Sat is characterised the same way as in PCTL, with the following additional statements:

$$Sat(\mathcal{E}_C(\Phi)) = \{s \in S \mid \mathbb{E}_s(\mathsf{TS}^{Sat(\Phi)}) \in C\} \text{ for } C \subseteq \mathbb{N} \cup \{+\infty\}, \text{ and}$$
$$Sat(\mathcal{P}_J(\phi)) = \{s \in S \mid \mathbb{P}_s(Paths(s, \phi)) \in J\} \text{ for } J \subseteq [0,1] \cap \mathbb{Q},$$

where $Paths(s, \phi) = \{\pi \in Paths(s) \mid \pi \models \phi\}$. $\mathcal{P}_J(\phi)$ is computed according to the formula ϕ as in PCTL with the following additional statement:

$$\mathbb{P}_s(Paths(s, \Phi \,\mathsf{U}_{\leq \ell}\, \Psi)) = \mathbb{P}_s(Sat(\Phi) \,\mathsf{U}_{\leq \ell}\, Sat(\Psi)) \text{ for } s \in S \text{ and } \ell \in \mathbb{N}.$$

We already have presented a way to compute $\mathbb{P}_s(\Diamond_{\leq \ell}\, T) = \mathbb{P}_s(S \,\mathsf{U}_{\leq \ell}\, T)$ for a given state $s \in S$, a set of target states $T \subseteq S$, and a cost threshold $\ell \in \mathbb{N}$ (cf. Theorem 1.4). We slightly adapt this method to handle the computation of $\mathbb{P}_s(C \,\mathsf{U}_{\leq \ell}\, T)$ for $s \in S$, $C, T \subseteq S$, and $\ell \in \mathbb{N}$. We make all states of T and $S \setminus (C \,\mathsf{U}\, T)$ absorbing (cf. proof of Corollary 2.2), yiedling a new MC $\mathcal{M}'$. Then, we build the unfolding of $\mathcal{M}'$ up to ℓ (cf. Appendix A.2), i.e., $\mathcal{M}'_\ell$. We finally resolve a classical cost-bounded reachability (i.e., $C \,\mathsf{U}\, T$) in $\mathcal{M}'_\ell$.

Example 2.6 (*PRCTL model checking: expectation operator*). Let $\mathcal{M}$ be the MC with state space S and atomic proposition space AP of Figure 2.12, and the state formula $\Phi = \mathcal{E}_{\leq 5}(c \vee \mathcal{P}_{\geq \frac{1}{2}}(a \,\mathsf{U}\, (b \wedge c)))$. We are interested to know if $s_2 \models \Phi$. Let $\Psi = c \vee \mathcal{P}_{\geq \frac{1}{2}}(a \,\mathsf{U}\, (b \wedge c)))$. So, we compute $Sat(\mathcal{E}_{\leq 5}(\Psi)) = \{s \in S \mid \mathbb{E}_s(\mathsf{TS}^{Sat(\Psi)}) \leq 5\}$. The set $Sat(\Psi)$ has already been computed in Example 2.5. Then, the set $Sat(\mathcal{E}_{\leq 5}(\Psi)) = \{s \in S \mid \mathbb{E}_s(\mathsf{TS}^{\{s_0, s_1, s_4\}}) \leq 5\}$. We can compute $\mathbb{E}_s(\mathsf{TS}^{\{s_0, s_1, s_4\}})$ for all $s \in S$, according to the equation system defined in Appendix A.1.2. Let $S_{=?} = \{s \in S \mid \mathbb{P}_s(\Diamond\{s_0, s_1, s_4\}) = 1\} \setminus \{s_0, s_1, s_4\} = \{s_2, s_3\}$ and $(x_s)_{s \in S} \in [0,1]^{|S|}$

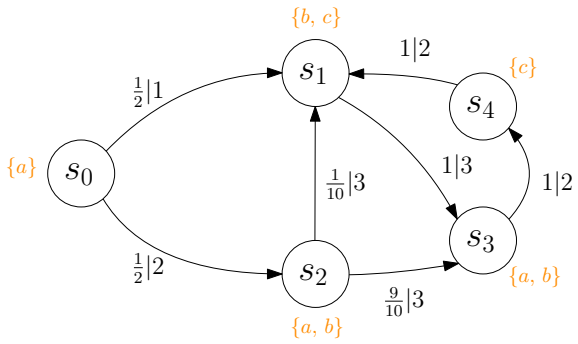


Figure 2.12. MC $\mathcal{M}$ with state space $S = \{s_0, s_1, s_2, s_3, s_4\}$ and atomic propositions of the set $AP = \{a, b, c\}$

such that $x_s = \mathbb{E}_s(\mathsf{TS}^{\{s_0,s_1,s_4\}})$ for all $s \in S$, we firstly have $x_{s_0} = x_{s_1} = x_{s_4} = 0$. We additionally have

$$\begin{pmatrix} x_{s_2} \\ x_{s_3} \end{pmatrix} = \begin{pmatrix} 0 & \frac{9}{10} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_{s_2} \\ x_{s_3} \end{pmatrix} + \begin{pmatrix} \frac{3}{10} \\ 2 \end{pmatrix},$$

and we thus have $x_{s_3} = \mathbb{E}_{s_3}(\mathsf{TS}^{\{s_0,s_1,s_4\}}) = 2 \leq 5$ and $x_{s_2} = \mathbb{E}_{s_2}(\mathsf{TS}^{\{s_0,s_1,s_4\}}) = \frac{9}{10}(x_{s_3} + 3) + \frac{3}{10} = \frac{9}{10} \cdot 5 + \frac{3}{10} = \frac{48}{10} \leq 5$. Then, $Sat(\mathcal{E}_{\leq 5}(\Psi)) = S$ what implies $s_2 \models \Phi$.

2.5 PRCTL for Markov decision processes

Again, PRCTL for Markov decision processes is slightly different than PRCTL for Markov Chains. We need to adapt the syntax and the semantic of PRCTL for Markov chains in order to support nondeterminism induced by Markov Decision processes. Indeed, in addition to the probabilistic operator $\mathcal{P}_J(.)$ that changes to $\mathcal{P}_J^{\max}(.)$ for the same reason as in PCTL, the expectation operator $\mathcal{E}_C(.)$ becomes ambiguous for Markov decision processes due to nondeterminism. In Markov chains, this operator directly refers to the expected cost to reach a set of target states. We cannot address the expect cost-to-target in Markov decision process without referring to the notion of strategies. Then, we replace this operator with $\mathcal{E}_C^{\min}(.)$, referring to the expected cost-to-target in the Markov chain induced by the strategy minimising this cost in the MDP.

Syntax and semantic

Definition 2.16 (**Syntax of PRCTL for MDPs**). Let $\mathcal{M}$ be an MDP with space of atomic propositions AP,

- PRCTL *state formulae* over AP are formed according to the following grammar:

$$\Phi ::= true \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \mathcal{P}_J^{\max}(\phi) \mid \mathcal{E}_C^{\min}(\Phi)$$

  where $a \in AP$ is an atomic proposition, $J \subseteq [0,1] \cap \mathbb{Q}$ is an interval of probability bounds, $C \subseteq \mathbb{N} \cup \{+\infty\}$ gives cost bounds, and ϕ is a path formula.

- PRCTL *path formulae* over AP are formed according to the same grammar as for PRCTL path formulae in Markov chains context.

Definition 2.17 (**Semantic of PRCTL for MDPs**)**.** Let $\mathcal{M}$ be an MDP with state space S, $s \in S$ and atomic proposition space AP, and Φ be a PRCTL state formula over AP, the semantic of PRCTL state formulae for MDPs is exactly the same as the PCTL state formulae semantic for MDPs, with the additional expectation operator $\mathcal{E}_C^{\min}(.)$ for which the semantic is defined as follows:

$$s \models \mathcal{E}_C^{\min}(\Phi) \quad \text{iff} \quad \mathbb{E}_s^{\min}(\mathsf{TS}^{Sat(\Phi)}) \in C.$$

The PRCTL semantic of paths formulae for MDPs is the same as the PRCTL semantic of paths formulae for MCs.

Model checking

As the PRCTL model checking is done through the recursive computation of the satisfaction set, we just need to slightly adapt the characterisation of the PCTL satisfiability set for MDPs.

Property 2.4 (PRCTL characterisation of Sat for MDPs)**.** Let $\mathcal{M}$ be an MDP with state space S and atomic proposition space AP, Φ, Ψ be PRCTL state formulae over AP, and ϕ be a PRCTL path formula over AP. The satisfaction set Sat is characterised the same way as in PCTL, with the following additional statements:

$$Sat(\mathcal{E}_C^{\min}(\Phi)) = \{s \in S \mid \mathbb{E}_s^{\min}(\mathsf{TS}^{Sat(\Phi)}) \in C\} \text{ for } C \subseteq \mathbb{N} \cup \{+\infty\}, \text{ and}$$
$$Sat(\mathcal{P}_J^{\max}(\phi)) = \{s \in S \mid \mathbb{P}_s^{\max}(Paths(s, \phi)) \in J\} \text{ for } J \subseteq [0, 1] \cap \mathbb{Q},$$

where $Paths(s, \phi) = \{\pi \in Paths(s) \mid \pi \models \phi\}$. $\mathcal{P}_J^{\max}(\phi)$ is computed according to the formula ϕ as in PCTL, with the following additional statement:

$$\mathbb{P}_s^{\max}(Paths(s, \Phi \,\mathsf{U}_{\leq \ell}\, \Psi)) = \mathbb{P}_s^{\max}(Sat(\Phi) \,\mathsf{U}_{\leq \ell}\, Sat(\Psi)) \text{ for } s \in S \text{ and } \ell \in \mathbb{N}.$$

We compute $\mathbb{P}_s^{\max}(C \,\mathsf{U}_{\leq \ell}\, T)$ for a given state s of the system, $C, T \subseteq S$, and $\ell \in \mathbb{N}$, by reduction to the cost-bounded reachability to T the same way as we compute the classical constrained reachability by reduction to the reachability to T (cf. proof of Theorem 2.7). We make all states $s^* \in S \setminus (C \cup T)$ absorbing, i.e., we replace all enabled actions in $A(s^*)$ by a unique action α_{s^*} such that $\Delta(s^*, \alpha_{s^*}, s^*) = 1$. Then, we compute $\mathbb{P}_s^{\max}(\Diamond_{\leq \ell}\, T)$ in this modified MDP and we build the linked strategy, corresponding to the optimal strategy built for the $\mathsf{SSP\text{-}P}$ problem for the state s, the subset of target states T, the cost threshold ℓ, and an arbitrary probability threshold (cf. Theorem 1.7).

Remark 2.5 (*Value iteration for the minimal expected cost-to-target*). The equation system in Appendix B.1 suggests an approximation technique, called value iteration to compute the minimal expected cost-to-target in an MDP. This value iteration technique can be defined in a similar way as for maximising the probability to reach this target (cf. Theorem 2.5).

Remark 2.6 (*PRCTL model checking and* SSP-E *problem*). Let $\mathcal{M}$ be an MDP with state space S and atomic proposition space AP, $s \in S$, $\ell \in \mathbb{N}$, and Φ be a PRCTL state formula over AP, verify $s \models \mathcal{E}^{\min}_{\leq \ell}(\Phi)$ is equivalent to solve the SSP-E problem for the state s, the cost threshold ℓ and the subset of target states $Sat(\Phi)$.

Remark 2.7 (*PRCTL model checking and* SSP-P *problem*). Let $\mathcal{M}$ be an MDP with state space S and atomic proposition space AP, $s \in S$, $\ell \in \mathbb{N}$, $\alpha \in [0,1] \cap \mathbb{Q}$, and Φ be a PRCTL state formula over AP, verify $s \models \mathcal{P}^{\max}_{\geq \alpha}(\Diamond_{\leq \ell}\, \Phi)$ is equivalent to solve the SSP-P problem for the state s, the cost threshold l, the probability threshold α and the set of target states $Sat(\Phi)$.

Example 2.7 (*PRCTL model checking: cost-bounded until operator*). Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be the MDP of Figure 2.13, we are interested in verifying $s_0 \models \mathcal{P}^{\max}_{\geq \frac{1}{5}}((a \vee b)\, \mathsf{U}_{\leq 8}\, c)$. First, we begin by computing the sets $Sat(a \vee b)$ and $Sat(c)$.

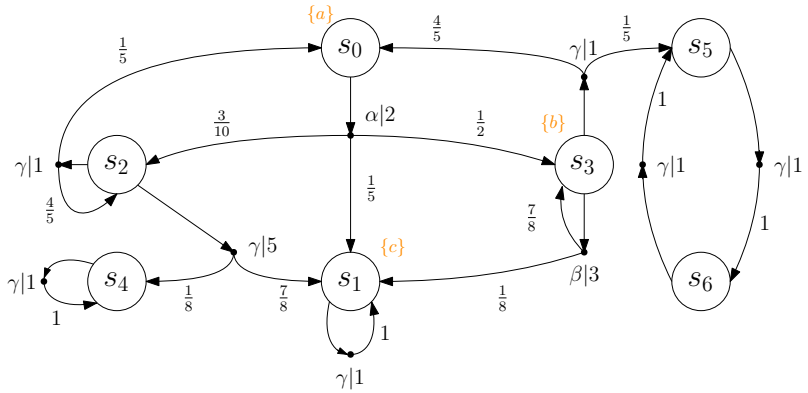


Figure 2.13. MDP $\mathcal{M}$, with state space $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_5\}$ and with space of atomic propositions $AP = \{a, b, c\}$

Let $C = Sat(a \vee b)$ and $T = Sat(c)$. We have $C = \{s_0, s_3\}$ and $T = \{s_1\}$. Then, we make all states of the set $S \setminus (C \cup T)$ absorbing, i.e., all states of $\{s_2, s_4, s_5, s_6\}$ (cf. Figure 2.14), to set up the reduction of this problem to the cost-bounded reachability to T. Now, it remains to compute $\mathbb{P}^{\max}_{s_0}(\Diamond_{\leq 8}\, T)$ in the modified MDP. To do that,

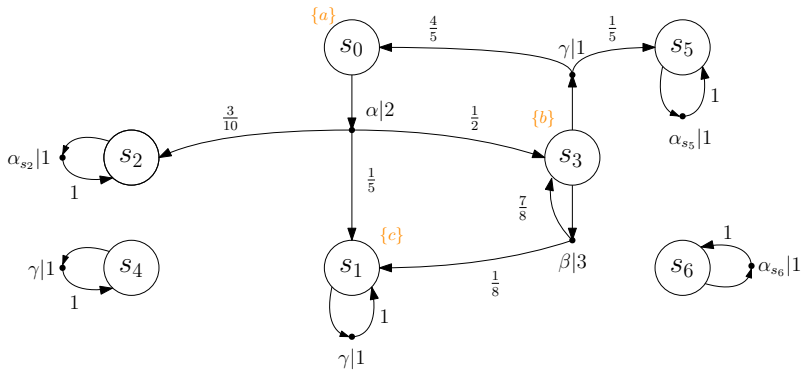


Figure 2.14. MDP $\mathcal{M}$ modified to verify the property $\mathcal{P}^{\max}_{\geq \frac{1}{5}}((a \vee b)\, \mathsf{U}_{\leq 8}\, c)$ by reduction to the cost-bounded reachability to T

we unfold it from the state s_0 up to the cost threshold $\ell = 8$ (cf. Figure 2.15).

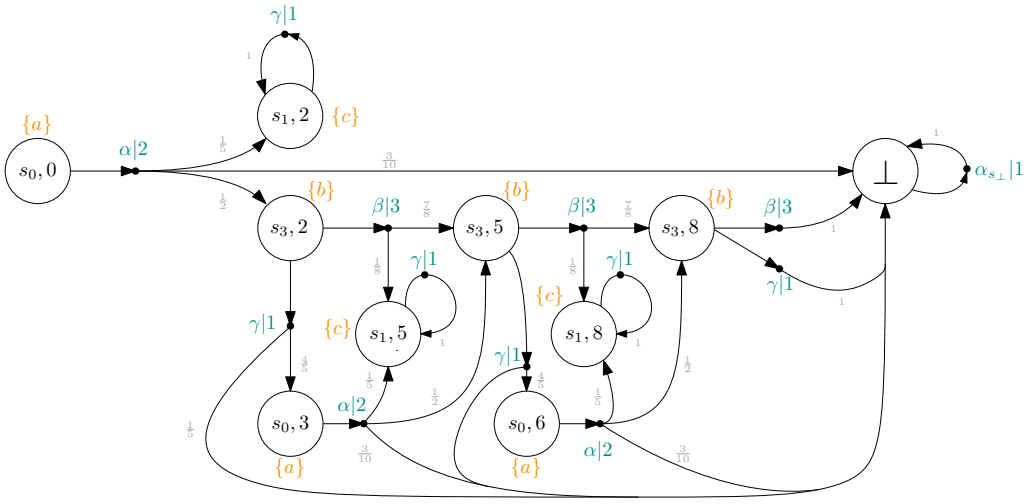


Figure 2.15. Unfolding of the modified MDP from the state s_0 up to the cost threshold $\ell = 8$

Let S_ℓ be the state space of the unfolding of $\mathcal{M}$ and $T_\ell = \{(s_1, 2), (s_1, 5), (s_1, 8)\}$ be the new set of target states. We compute $\mathbb{P}^{\max}_{(s_0,0)}(\Diamond\, T_\ell)$ in this unfolding of $\mathcal{M}$ according to the linear program defined in Appendix A.3.1: let $(x_{(s,v)})_{(s,v)\in S_\ell} \in [0,1]^{|S_\ell|}$ such that $x_{(s,v)} = \mathbb{P}^{\max}_{(s,v)}(\Diamond T_\ell)$ for all $(s,v) \in S_\ell$. We have $x_{s_\bot} = x_{(s_3,8)} = 0$ because $s_\bot$ and $(s_3, 8)$ are not connected to T_ℓ and $x_t = 1$ for all $t \in T_\ell$. Then, the vector $(x_s)_{s\in S_\ell}$ is the unique solution of the following linear program:

$$\min\ x_{(s_0,0)} + x_{(s_0,3)} + x_{(s_0,6)} + x_{(s_3,2)} + x_{(s_3,5)}$$

subject to

$$\begin{aligned}
x_{(s_0,0)} - \tfrac{1}{2}x_{(s_3,2)} &\geq \tfrac{1}{5} & \qquad x_{(s_3,2)} - \tfrac{7}{8}x_{(s_3,5)} &\geq \tfrac{1}{8}\\
-\tfrac{4}{5}x_{(s_0,3)} + x_{(s3,2)} &\geq 0 & \qquad x_{(s_3,5)} &\geq \tfrac{1}{8}\\
-\tfrac{4}{5}x_{(s_0,6)} + x_{(s_3,5)} &\geq 0 & \qquad x_{(s_0,6)} &\geq \tfrac{1}{5}\\
x_{(s_0,3)} - \tfrac{1}{2}x_{(s_3,5)} &\geq \tfrac{1}{5}
\end{aligned}$$

$$\begin{aligned}
0 \leq x_{(s_0,0)} &\leq 1 & \qquad 0 \leq x_{(s_0,3)} &\leq 1\\
0 \leq x_{(s_0,6)} &\leq 1 & \qquad 0 \leq x_{(s_3,2)} &\leq 1\\
0 \leq x_{(s_3,5)} &\leq 1
\end{aligned}$$

We can solve this linear program via the simplex method. This yields $x_{(s_0,0)} = 0.3325$. Since $x_{(s_0,0)} = \mathbb{P}^{\max}_{(s_0,0)}(\Diamond T_\ell) = \mathbb{P}^{\max}_{s_0}(\Diamond_{\leq 8}\, T)$ in the modified MDP, we have that $x_{(s_0,0)} = \mathbb{P}^{\max}_{s_0}(C\, \mathsf{U}_{\leq 8}\, T)$ in $\mathcal{M}$, and $x_{(s_0,0)} \geq \frac{1}{5}$. Then, $s_0 \models \mathcal{P}^{\max}_{\geq \frac{1}{5}}((a \vee b)\, \mathsf{U}_{\leq 8}\, c)$.

Chapter 3

Multi-objective problems

In real-world environments, considering a single reachability or shortest path problem might not be sufficient for some situations. Indeed, building an optimal strategy satisfying one problem could negatively affect the results of applying this same strategy to satisfy other relevant problems. For instance, we should be interested in a strategy that does not fail to reach a target with a cost bounded and that however optimises the expected cost to this target. Moreover, we should also be interested in getting a strategy satisfying at a time many reachability problems, and furthermore, many percentile problems. This chapter introduces *multi-objective* problems, where we are interested in solving *simultaneously* multiple sub-problems introduced in previous chapters.

Example 3.1 (*Communication between nodes in a wireless sensor network*). Consider a wireless sensor network containing three nodes n_0, n_1 and n_2. Assume that n_0, n_1 and n_3 are separated by the same distance, that the sensor n_0 has to send messages to n_2 at regular intervals and that a wall stands between n_0 and n_2 (cf. Figure 3.1). There are two possibilities for n_0. First, it can send directly the message to n_2.

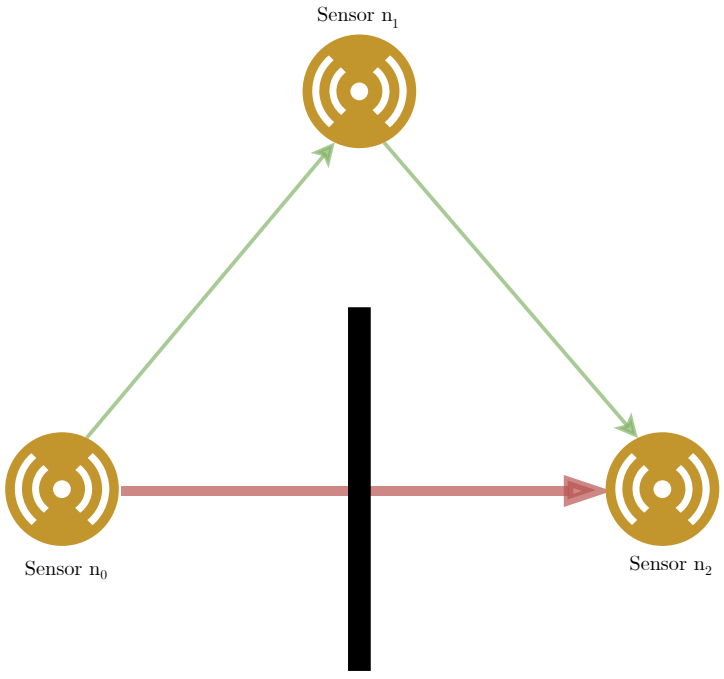


Figure 3.1. Communication between three nodes in a wireless sensor network

In that case, sending the message is quite quick, but requires more energy that a

standard message sending. Moreover, this sending has a risk of packet corruption due to the possible noise induced when the signal goes through the wall standing between the two sensors. The second possibility is to send the message to an intermediate node (i.e., n_1). This sending is obviously slower due to the sending in two steps, but requires less energy in total that a direct sending. In that case, the risk of packet corruption is negligible.

This example induces some questions. Firstly, we should be interested by a strategy ensuring a bounded duty cycle for the emitter sensor n_0 (i.e., a bounded time for its active session). Thus, we should be interested by a strategy minimising the expected energy of sending the message while ensuring this duty cycle. Finally, we should be interested by a strategy offering a good probability to have a bounded duty cycle while having a good probability to minimise the energy used by the sensor to send the message.

3.1 Worst case guarantee

Firstly, we are interested in having a worst case guarantee to reach some states in a Markov decision process, i.e., a guarantee to reach these states with a cost bounded. To do that, we approach the problem by considering it as a *shortest path game* problem [RRS15].

Definition 3.1 (SP-G **problem**). Let $\mathcal{M}$ be an MDP with state space S, $s \in S$, $T \subseteq S$, and $\ell \in \mathbb{N}$. The shortest path game problem (SP-G, for short) consists in deciding if there exists a strategy σ ensuring to reach the subset of target states T from the state s with a cost bounded ℓ, i.e.,

$$\forall \pi \in Paths^{\sigma}(s),\ \mathsf{TS}^{T}(\pi) \leq \ell.$$

Theorem 3.1 (SP-G ***problem and qualitative cost bounded reachability***). *Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP, $s \in S$, $T \subseteq S$, $\ell \in \mathbb{N}$, and σ be a strategy for $\mathcal{M}$. The following two propositions are equivalent:*

(a) $\forall \pi \in Paths^{\sigma}(s),\ \mathcal{TS}^{T}(\pi) \leq \ell$.

(b) $\mathbb{P}^{\sigma}_{s}(\Diamond_{\leq \ell} T) = 1$.

Proof:
$((a) \implies (b))$. Assume that for any σ-path $\pi \in Paths^{\sigma}(s)$, the truncated sum of this path is lower than or equal to ℓ, i.e., $\mathsf{TS}^{T}(\pi) \leq \ell$. We have $\mathbb{P}^{\sigma}_{s}(\Diamond_{\leq \ell} T) = \mathbb{P}^{\sigma}_{s}(\{\pi \in Paths(s) \mid \mathsf{TS}^{T}(\pi) \leq \ell\})$. As all σ-paths starting from s have a truncated sum lower than or equal to ℓ, we have $\{\pi \in Paths^{\sigma}(s) \mid \mathsf{TS}^{T}(\pi) \leq \ell\} = Paths^{\sigma}(s)$. Then, $\mathbb{P}^{\sigma}_{s}(\Diamond_{\leq \ell} T) = \mathbb{P}^{\sigma}_{s}(Paths(s)) = 1$.
$(\neg(a) \implies \neg(b))$. Assume there exists a σ-path $\pi \in Paths^{\sigma}(s)$ such that $\mathsf{TS}^{T}(\pi) > \ell$. Thus, consider this path π and let $\hat{\pi} = s_0 \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n} s_n \in Pref(\pi)$, where α_i is chosen by the strategy σ for each $i \in \{1, \dots, n\}$, such that $\sum_{i=1}^{n} w(\alpha_i) > \ell$. We have $\pi \in Cyl(\hat{\pi})$ and $\mathbb{P}^{\sigma}_{s}(Cyl(\hat{\pi})) = \prod_{i=0}^{n-1} \Delta(s_i, s_{i+1}) > 0$. We obviously additionally have

that $Cyl(\hat{\pi}) \cap \{\pi \in Paths^{\sigma}(s) \mid \mathsf{TS}^{T}(\pi) \leq \ell\} = \emptyset$, because $\mathsf{TS}^{T}(\hat{\pi}) > \ell$. Then, we have

$$\mathbb{P}_{s}^{\sigma}(\Diamond_{\leq \ell} T) = \mathbb{P}_{s}^{\sigma}(\{\pi \in Paths(s) \mid \mathsf{TS}^{T} \leq \ell\}) \leq 1 - \mathbb{P}_{s}^{\sigma}(Cyl(\hat{\pi})) < 1.$$

□

Referring to Theorem 3.1, it is possible to solve the SP-G problem by reduction to the SSP-P problem. However, the SSP-P problem is computed in pseudo-polynomial time in $\mathcal{M}$ and in the size of the length of ℓ, and we want to avoid this time complexity. For this approach, we consider the MDP as a *two-player turn-based game*. A two-player game on a graph is characterised by a directed graph, where vertices are partitioned between two players. A path (or a *play*) in such a game may be imagined as follows: a token is placed on some initial vertex, belonging to a player, and this player chooses an outgoing edge of this vertex. Then, the token moves through this edge, and this operation is repeated infinitely often (see [BFa02] for more details). Here, at each step, the Markov decision process is in state s and the player one chooses an enabled action α of s. Then, the player two chooses an α-successor of s. In order to solve the problem, we consider the player two as an adversary, i.e., the player two chooses the α-successor leading the worst case in terms of truncated sum at each step. Thus, we do not consider probabilities for this problem but rather consider the worst case of choosing each action. Finding an optimal strategy for the player one allowing to reach T with a cost less than or equal to ℓ under the choices of the player two solves the SP-G problem.

We compute the shortest path from s ensuring to reach T by dynamic programming in order to build a strategy satisfying the SP-G problem. Let $|S| = n$ and $\mathbb{C} : S \times \{0, \ldots, n-1\}$ be a function such that $\mathbb{C}(s, i)$ is the shortest path from s ensuring to reach T after maximum i steps. Obviously, $\mathbb{C}$ can be represented with a matrix of size $n \times n$. We have

$$\mathbb{C}(s, 0) = \begin{cases} 0 & \text{if } s \in T \\ +\infty & \text{else.} \end{cases}$$

Then, let $k \in \mathbb{N}$, such that $0 \leq k < n-1$, and assume $\mathbb{C}(s, k)$ has been computed for all $s \in S$,

$$\mathbb{C}(s,\, k+1) = \min\Big[\mathbb{C}(s,\, k),\ \min_{\alpha \in A(s)} \max_{s' \in Succ(s,\alpha)} \Big(w(\alpha) + \mathbb{C}(s',\, k)\Big)\Big].$$

This recursive definition of $\mathbb{C}$ actually means that the shortest path from s ensuring to reach T in maximum $k+1$ steps corresponds to either the shortest path from s ensuring to reach T in maximum k steps or the minimal weight of choosing an action α plus the shortest path from the worst α-successor s' of s ensuring to reach T in maximum k steps. The minimisation term according to an action in this definition corresponds to the choice of the strategy. The maximisation term in this definition corresponds to the adversary (i.e., the player two), choosing the worst α-successor of s. As the weight of each action is strictly positive, the strategy never chooses

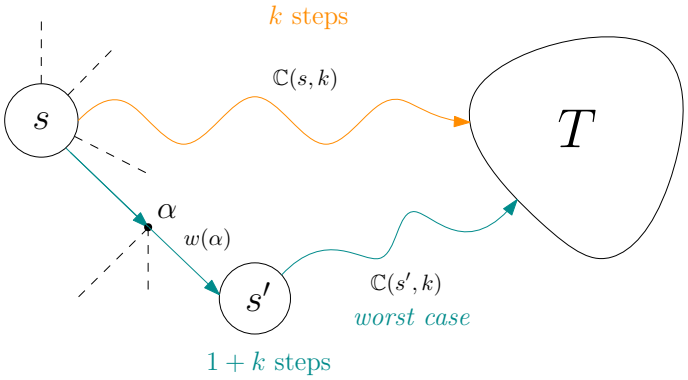


Figure 3.2. Intuitive representation of the function $\mathbb{C}$, used to solve the SP-G problem

an action possibly leading to a cycle if it can avoid it. Indeed, choosing an action possibly leading to a cycle necessarily yields an infinite truncated sum since the player two will necessarily choose the transition yielding this cycle. Thus, if a strategy satisfying this problem exists, it wins in at most n steps. It is why we only need to consider the n first steps.

Theorem 3.2 (*Solving the* SP-G *problem*). *The* SP-G *problem can be decided in polynomial time in the size of $\mathcal{M}$, and an optimal pure memoryless strategy always exists and can be built in polynomial time in the size of $\mathcal{M}$.*

Proof: By dynamic programming, for each $i \in \{1, \dots, n-1\}$, assuming we have computed $\mathbb{C}(s, i-1)$ for all $s \in S$, we can determine all values of $\mathbb{C}(s, i)$, $s \in S$, in $\mathcal{O}(|\mathcal{M}|)$ time. Indeed, at each iteration on i, we look at all successors of all states s, and we thus look at all the transitions of the system. Then, as $|\{0, \dots, n-1\}| = |S|$, computing all the values of C for all entries $(s, i) \in S \times \{0, \dots, n-1\}$ can be done in $\mathcal{O}(|S| \cdot |\mathcal{M}|)$.

Let s be a state of the MDP. By definition of $\mathbb{C}$, we have $\mathbb{C}(s,\, k+1) \leq \mathbb{C}(s,\, k)$, for all $k \in \mathbb{N}$ such that $0 \leq k < n-1$. Then, we have $\mathbb{C}(s,\, n-1) = \min_k \mathbb{C}(s,\, k)$, and a strategy satisfying the SP-G problem for $s \in S$ exists if and only if $\mathbb{C}(s,\, n-1) \leq \ell$. Furthermore, we can build this pure memoryless strategy as follows:

$$\sigma : S \to A,\ s \mapsto \arg\min_{\alpha \in A(s)} \Big[\max_{s' \in Succ(s,\alpha)} \Big(w(\alpha) + \mathbb{C}(s',\, n-1) \Big) \Big].$$

The key idea behind this definition is that σ chooses the action minimising the weight of the enabled actions plus the smallest cost to surely reach T from the worst successor, chosen by the player two.

□

Remark 3.1 (SP-G *problem and PRCTL model checking*). Let $\mathcal{M}$ be an MDP with state space S and atomic proposition space AP, $s \in S$, $\ell \in \mathbb{N}$, and Φ be a PRCTL state formula over AP, verify $s \models \mathcal{P}^{\max}_{=1}(\Diamond_{\leq \ell}\, \Phi)$ is equivalent to solve the SP-G problem for the state s, the cost threshold ℓ, the subset of target states $Sat(\Phi)$ and the probability threshold one.

Example 3.2 (SP-G *problem in the wireless sensor network*). Getting back to Example 3.1, we only consider the time dimension to model the situation of this example. We focus on the behaviour of the node n_0 (cf. Figure 3.3). When n_0 becomes active, it has to send a message to n_2. We assume here that a standard message sending takes 2 *milliseconds* (ms). If n_0 decides to send the message passing by n_1, it must wait to receive an acknowledgement of the message sent from n_1 to n_2. We assume that an acknowledgement message takes 2 ms to be received. Thus, we have that n_0 sends the message to n_1 (2 ms), then n_1 sends the message to n_2 (2 ms). When n_2 receives the message, it sends an acknowledgement to n_1 (2 ms) and n_1 sends an acknowledgement to n_0 (2 ms). Then, after that n_0 has sent the message to n_1, he waits 6 ms the acknowledgement from n_1 and go to sleep during 10 ms after that. If n_0 decides to directly send a message to n_2, then n_2 sends the message and waits 2 ms the acknowledgement. Due to the possible packet corruption, n_2 does not receive the acknowledgement with a probability $\frac{1}{8}$. In that case, it has to try sending the message again. Else, it goes to sleep during 10 ms. Assume we want to ensure a duty cycle of 12 ms for the node n_0. Then, let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be the MDP of Figure 3.3, we want to solve the following SP-G problem:

$$s_0 \overset{?}{\models} \mathcal{P}^{\max}_{=1}(\Diamond_{\leq 12}\ sleep).$$

To solve this problem, we will recursively compute $\mathbb{C}$ for all $s \in S$ from $k = 0$ to $k = 3$.

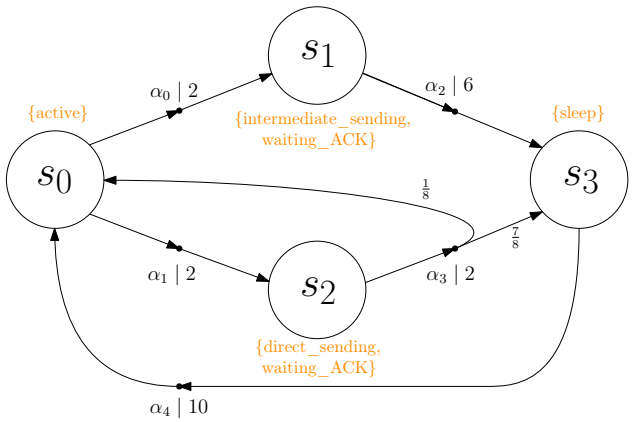


Figure 3.3. MDP representing the behaviour of the node n_0, according to the situation of Figure 3.1 from Example 3.1

$k = 0$
- $\mathbb{C}(s_0, 0) = \mathbb{C}(s_1, 0) = \mathbb{C}(s_2, 0) = +\infty$
- $\mathbb{C}(s_3, 0) = 0$

$k = 1$
- $\mathbb{C}(s_0, 1) = +\infty$
- $\mathbb{C}(s_1, 1) = 6 + 0 = 6$
- $\mathbb{C}(s_2, 1) = 2 + \infty = +\infty$

$k = 2$
- $\mathbb{C}(s_0, 2) = 2 + 6 = 8$

$k = 3$
- $\mathbb{C}(s_2, 3) = 2 + 8 = 10$

S	$k=0$	$k=1$	$k=2$	$k=3$
s_0	$+\infty$	$+\infty$	8	8
s_1	$+\infty$	6	6	6
s_2	$+\infty$	$+\infty$	$+\infty$	10
s_3	0	0	0	0

Table 3.1. Values of $\mathbb{C}$ (dynamic programming)

The result $\mathbb{C}(s_2, 3) = 10$ is interesting. It means that the player one firstly chooses the action α_3 (with a weight of 2). Then, the player two chooses the worst case, i.e., the transition $s_2 \xrightarrow{\alpha_3} s_0$, and the player one next chooses the action surely leading s_0 to s_3 with a minimal cost (i.e., $\mathbb{C}(s_0, 2) = 8$ with $s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_2} s_3$).

As $\mathbb{C}(s_0, 3) \leq 12$, the duty cycle is ensured by the strategy built with $\mathbb{C}$, and we have $s_0 \models \mathcal{P}^{\max}_{=1}(\Diamond_{\leq 12}\ \mathit{sleep})$. With this strategy, the sensor n_0 always chooses to send the message via the intermediate sensor n_1. This ensures a duty cycle of 8 ms, but it is possible to have a better expected time to reach the *sleep* state while ensuring the duty cycle of 12 ms (e.g., with a memory strategy). So, the next type of strategy that we will study is strategies offering a good expected cost-to-target while ensuring to reach this target with a cost bounded.

3.2 Good expectation under a worst case

We now study strategies ensuring simultaneously a worst case guarantee (cf. Section 3.1) and a good expected cost-to-target in a Markov decision process [RRS15, BFRR13].

Definition 3.2 (SSP-WE **problem**)**.** Let $\mathcal{M}$, be an MDP with state space S, $s \in S$, $T \subseteq S$, and $\ell_1, \ell_2 \in \mathbb{N}$. The *stochastic shortest path worst case expectation* problem (SSP-WE, for short) consists in deciding if there exists a strategy σ such that:

- $\forall \pi \in \mathit{Paths}^\sigma(s),\ \mathsf{TS}^T(\pi) \leq \ell_1$.
- $\mathbb{E}^\sigma_s(\mathsf{TS}^T) \leq \ell_2$.

The SSP-WE problem is thus a *multi-objective* problem: indeed, we actually want to solve simultaneously the SP-G problem and the SSP-E problem. As a reminder, in the previous section we have replaced the probabilities by an adversarial choice to handle the guarantee to reach a set of target states according to a worst case. Here, we combine this approach with taking into account probabilities for the expected cost-to-target part. Although the SSP-E and the SP-G problems can both be decided in polynomial time, the SSP-WE problem is more complex due to the strategy having to satisfy both problems at the same time.

Notation 3.1 (*Attractors of a set of target states in an MDP*). Let $\mathcal{M}$ be an MDP with state space S and $T \subseteq S$. In a two-player game, the attractor of T for the player one is the set of states from which the player one can ensure to reach T no matter what the player two does. In an MDP, by considering that the player one chooses an enabled action α of each state s, and that the player two chooses an α-successor of s at each turn, the attractor of T is the set

$$\mathsf{Attr}(T) = \{s \in S \mid \exists \sigma, \forall \pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \cdots \in \mathit{Paths}^\sigma(s), \exists n \in \mathbb{N},\ s_n \in T\}.$$

In order to solve an SSP-WE problem for an MDP $\mathcal{M}$, a set of target states T and a cost threshold ℓ_1, we unfold $\mathcal{M}$ up to ℓ_1. Then, we restrict the state space of

the unfolding of $\mathcal{M}$ to the attractor of the set of target states T in $\mathcal{M}_{\ell_1}$. We then build an optimal strategy offering the minimal expected cost-to-target in $\mathcal{M}_{\ell_1}$. To do that, we compute the *set of safe actions* for each state by dynamic programming. That is, we compute the set of enabled actions of each state surely leading this state to a successor being in $\mathsf{Attr}(T_{\ell_1})$, where T_{ℓ_1} is the set of target states T in $\mathcal{M}_{\ell_1}$ for which the cost threshold has not been exceeded.

More formally, let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP, the SSP-WE problem for $\mathcal{M}$, the state $s \in S$, the subset of target states $T \subseteq S$, and the cost thresholds $\ell_1, \ell_2 \in \mathbb{N}$ can be solved with the following algorithm:

Algorithm 4 Solving the SSP-WE problem

1. We compute the unfolding of $\mathcal{M}$ from s up to the cost threshold ℓ_1, i.e., $\mathcal{M}_{\ell_1}$ with state space S_{ℓ_1}.

2. Then, we compute the set of *safe actions* allowing each state in S_{ℓ_1} to surely go to a successor being in the attractor of $T_{\ell_1} = \{(s, v) \in S_{\ell_1} \mid s \in T \wedge v \leq \ell_1\}$. More formally, we compute $\mathbb{A} : S_{\ell_1} \to 2^A$,

$$(s, v) \mapsto \{\alpha \in A(s) \mid \forall (s', v') \in Succ((s, v), \alpha),\ (s', v') \in \mathsf{Attr}(T_{\ell_1})\}$$

   for all states $(s, v) \in S_{\ell_1}$. Thus, for each state $(s, v) \in S_{\ell_1}$, $\mathbb{A}(s, v)$ is the set of actions $\alpha \in A(s)$ ensuring to almost surely reach T_{ℓ_1} from (s, v).

3. Next, we compute $\mathcal{M}^{\mathbb{A}}_{\ell_1}$, the unfolding of $\mathcal{M}$ up to ℓ_1 restricted by the attractor of T_{ℓ_1}. The key idea is that we remove the states (s, v) such that $\mathbb{A}(s, v) = \emptyset$ from $\mathcal{M}_{\ell_1}$. More formally, we define $\mathcal{M}^{\mathbb{A}}_{\ell_1} = (S^{\mathbb{A}}_{\ell_1}, \mathbb{A}^*, \Delta^*_{\ell_1}, w, AP, L_{\ell_1})$ as follows:

   - $S^{\mathbb{A}}_{\ell_1} = \{(s, v) \in S_{\ell_1} \mid \mathbb{A}(s, v) \neq \emptyset \vee (s, v) \in T_{\ell_1}\}$.
   - $\mathbb{A}^*$ is the set of actions of this unfolding such that, for all action $\alpha \in A$, $\alpha \in \mathbb{A}^*$, and the set of enabled actions of each state $(s, v) \in S^{\mathbb{A}}_{\ell_1}$ is given by

$$\mathbb{A}^*(s, v) = \begin{cases} \mathbb{A}(s, v) & \text{if } \mathbb{A}(s, v) \neq \emptyset, \\ \{\alpha\}, \text{ where } \alpha \in A(s) & \text{else.} \end{cases}$$

     If $\mathbb{A}(s, v) = \emptyset$, then $(s, v) \in T_{\ell_1}$, and we make it absorbing with one of its enabled action α.

   - $\Delta^*_{\ell_1}$ is defined the same way as for the classical unfolding $\mathcal{M}_{\ell_1}$, except for the states $(s, v) \in T_{\ell_1}$ such that $\mathbb{A}(s, v) = \emptyset$. In that case, we make it absorbing with $\Delta^*_{\ell_1}((s, v), \alpha, (s, v)) = 1$ and $\Delta^*_{\ell_1}((s, v), \alpha, (s', v')) = 0$, with $\{\alpha\} = \mathbb{A}^*(s)$.
   - L_{ℓ_1} is defined as for the classical unfolding $\mathcal{M}_{\ell_1}$.

4. Finally, we solve the SSP-E problem in $\mathcal{M}^{\mathbb{A}}_{\ell_1}$ for the state $(s, 0)$, the subset of target states T_{ℓ_1}, and the cost threshold ℓ_2.

Let $(s, v) \in S_{\ell_1}$, the set of safe actions of (s, v), i.e., $\mathbb{A}(s, v)$, can be computed recursively as follows:

$$\mathbb{A}_0(s, v) = \begin{cases} A(s) & \text{if } v \leq \ell_1, \\ \emptyset & \text{else.} \end{cases} \tag{3.1}$$

Then, let $i \in \mathbb{N}$, assume that $\mathbb{A}_i(s, v)$ has already been computed for all $(s, v) \in S_{\ell_1}$,

$$\mathbb{A}_{i+1}(s, v) = \{\alpha \in \mathbb{A}_i(s, v) \mid \forall (s', v') \in Succ((s, v), \alpha) \setminus T_{\ell_1},\ \mathbb{A}_i(s', v') \neq \emptyset\}. \tag{3.2}$$

The key idea behind this definition of $\mathbb{A}_i$ is that we inductively disable the unsafe enabled actions of all states. We consider that a state that has no more enabled action is a *bad state*. At the initialisation, i.e., at $i = 0$, all enabled actions of each state are safe except if the current cost of a state has exceeded the threshold ℓ_1. In that case, this state is a bad state. By induction on i, at each step and for each state (s, v), we disable the actions α that risk to lead (s, v) to one of its α-successor considered as a bad state.

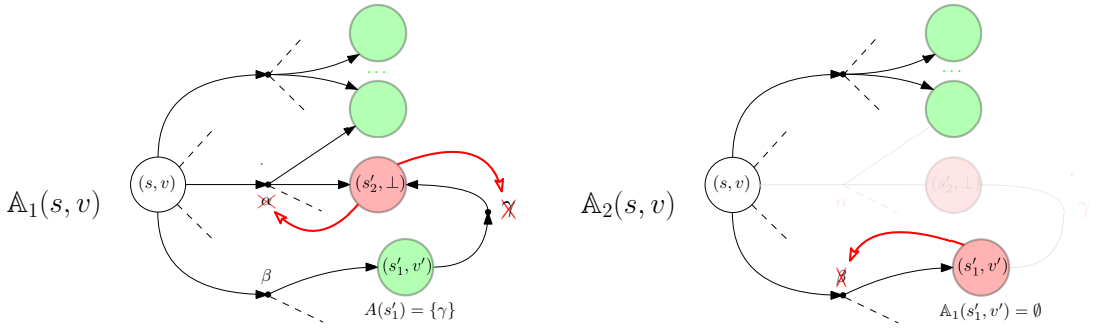


Figure 3.4. Intuition of the recursive computation of $\mathbb{A}$

By Theorem 3.1, we combine Algorithm 3 from Subsection 2.1.2 with this method, and we define an algorithm able to efficiently compute $\mathbb{A}$ (cf. Algorithm 5).

Lemma 3.1 (Correctness of Algorithm 5)**.** *Algorithm 5 is exact and allows to determine* $\mathbb{A}(s, v)$ *for all* $(s, v) \in S_{\ell_1}$.

Proof: The construction of $\mathbb{A}$ by Algorithm 5 exactly behaves like the recursive computation of $\mathbb{A}$, but allows to only consider the interesting states at each step instead of always checking all states of the system. The key idea behind this algorithm is that we remember at each iteration i which states (s', v') become "bad", in order to remove at iteration $i + 1$ all enabled actions α from each state (s, v) such that a state (s', v') tagged as bad in iteration i is in the set of α-successors of (s, v).
More formally, we prove that $\mathbb{A}$ is equal to $\mathbb{A}_i$ at each iteration i of the algorithm.

At $i = 0$, we initialise $\mathbb{A}(s, v)$ with $A_{\ell_1}(s, v)$ except for $\{s_\bot\}$ (lines 1 to 3), what exactly agrees with $\mathbb{A}_0$ (cf. Equation 3.1). Indeed, by definition of $\mathcal{M}_{\ell_1}$, being the unfolding of $\mathcal{M}$ up to ℓ_1, the state $s_\bot$ exactly captures all the states for which the threshold ℓ_1 has been exceeded. Note that we additionally initialise a variable U with $\{s_\bot\}$ (line 4).
Then, let $i \in \mathbb{N}_0$, we assume that $\mathbb{A}(s, v) = \mathbb{A}_i(s, v)$ for all states $(s, v) \in S_{\ell_1}$ at the

Algorithm 5 Computing the matrix of safe actions $\mathbb{A}$

Input : $\mathcal{M}_{\ell_1} = (S_{\ell_1}, A_{\ell_1}, \Delta_{\ell_1}, w, AP, L)$, the unfolding of an MDP $\mathcal{M}$ up to ℓ_1, and $T_{\ell_1} \subseteq S$, a subset of target states in this unfolding.
Output : The matrix of safe actions $\mathbb{A}$

1: $\mathbb{A} \leftarrow$ initialise a matrix of size $|S| \times \ell_1$ with $\emptyset$
2: **for** $(s, v) \in S_{\ell_1} \setminus \{s_\perp\}$ **do**
3: $\quad \mathbb{A}(s, v) \leftarrow A_{\ell_1}(s, v)$
4: $U \leftarrow \{s_\perp\}$
5: **while** $U \neq \emptyset$ **do**
6: $\quad R \leftarrow U$
7: $\quad$ **while** $R \neq \emptyset$ **do**
8: $\quad\quad$ let $(s', v') \in R$
9: $\quad\quad U \leftarrow U \setminus \{(s', v')\}$
10: $\quad\quad R \leftarrow R \setminus \{(s', v')\}$
11: $\quad\quad$ **if** $(s', v') \notin T_{\ell_1}$ **then**
12: $\quad\quad\quad$ **for all** $((s, v), \alpha) \in Pred((s', v'))$ such that $\alpha \in \mathbb{A}(s, v)$ **do**
13: $\quad\quad\quad\quad \mathbb{A}(s, v) \leftarrow \mathbb{A}(s, v) \setminus \alpha$
14: $\quad\quad\quad\quad$ **if** $\mathbb{A}(s, v) = \emptyset$ **then**
15: $\quad\quad\quad\quad\quad U \leftarrow U \cup \{(s, v)\}$
16: **return** $\mathbb{A}$

end of iteration i of the outermost loop (line 5). Moreover, we assume that all states tagged as bad at iteration i are strictly in U at the end of iteration i, i.e.,

$$U = U_i = \{(s', v') \in S_{\ell_1} \mid \mathbb{A}_i(s', v') = \emptyset \ \wedge \ \mathbb{A}_{i-1}(s', v') \neq \emptyset\}.$$

We are going to prove that $\mathbb{A}(s, v) = \mathbb{A}_{i+1}(s, v)$ and that $U = U_{i+1}$ at the end of iteration $i + 1$.
Let R be a new set strictly containing all states of U_i (line 6). For all bad states (s', v') of R, we start by removing (s', v') from U and R (lines 9 and 10). (*)
For each predecessor of each state removed from U, i.e., for each $((s, v), \alpha) \in Pred((s', v'))$ such that $(s', v') \notin T_{\ell_1}$ and $\alpha \in \mathbb{A}(s, v)$, we remove α from $\mathbb{A}(s, v)$, i.e., we remove each action $\alpha \in \mathbb{A}(s, v)$ such that there is a bad state $(s', v') \in Succ((s, v), \alpha) \setminus T_{\ell_1}$, where $\mathbb{A}_i(s', v') = \emptyset$ and $\mathbb{A}_{i-1} \neq \emptyset$ (lines 12 and 13). (**)
Finally, following this modification of $\mathbb{A}$, if $\mathbb{A}(s, v) = \emptyset$, we add (s, v) in U (line 15). (***)
First, by (**), we exactly have that $\mathbb{A}(s, v) = \mathbb{A}_{i+1}(s, v)$, for all $(s, v) \in S_{\ell_1}$ (cf. Equation 3.2). Indeed, at the end of iteration $i + 1$,

$$\mathbb{A}(s, v) = \mathbb{A}_i(s, v) \setminus \{\alpha \in \mathbb{A}_i(s, v) \mid \forall (s', v') \in Succ((s, v), \alpha) \setminus T_{\ell_1}, \ (s', v') \in U_i\},$$

for all $(s, v) \in S_{\ell_1}$.
By (*), when R is empty (line 7), all bad states of U_i have been considered, i.e., we have removed each state (s', v') such that $\mathbb{A}_i(s, v) = \emptyset$ from U. Then, by (***), we have that U equals to $U_{i+1} = \{(s', v') \in S_{\ell_1} \mid \mathbb{A}_{i+1}(s', v') = \emptyset \ \wedge \ \mathbb{A}_i(s', v') \neq \emptyset\}$.

Finally, it remains to show that the algorithm stops. In fact, it can be shown easily as the size of $\mathbb{A}$ is strictly decreasing at each iteration. The algorithm stops if there is no more action to remove. □

Lemma 3.2 (Time complexity of Algorithm 5)**.** *Computing* $\mathbb{A}(s,v)$ *for all* $(s,v) \in S_{\ell_1}$ *can be done in linear time in* $\mathcal{M}_{\ell_1}$*.*

Proof: As $a_\perp$ is the only action allowing a cycle in $\mathcal{M}_{\ell_1}$ and since this action is removed from $\mathbb{A}$ at initialisation, we ensure $s_\perp$ to be considered at most once, at the first iteration of the outermost loop 5. Thus, assuming $s_\perp$ is removed from $\mathcal{M}_{\ell_1}$, the underlying graph of $\mathcal{M}_{\ell_1}$ has no cycle. Since there is no cycle and we consider predecessors of states at each iteration of the algorithm, we are sure that each state can be added in U at most once (cf. lines 12, 13 and 14). Thus, in worst case, we tag all states as bad once, which exactly corresponds to $|S_{\ell_1}|$ operations. Then, as we consider all predecessors of states in U at each iteration of the algorithm, we look at all transitions of the system at most once, that exactly corresponds to $|\{(s,\alpha,s') \in S_{\ell_1} \times A_{\ell_1} \times S_{\ell_1} \mid \Delta_{\ell_1}(s,\alpha,s') > 0\}|$ operations. Adding up these two number of worst case operations, we get a a time complexity in $\mathcal{O}(|\mathcal{M}_{\ell_1}|)$. □

Theorem 3.3 (*Solving the* SSP-WE *problem*). *The* SSP-WE *problem can be decided in pseudo-polynomial time in the size of* $\mathcal{M}$ *and in the size of the length of* ℓ_1*. Pure pseudo-polynomial memory strategies are sufficient and in general necessary, and strategies satisfying this problem can be built in pseudo-polynomial time in the size of* $\mathcal{M}$ *and in the size of the length of* ℓ_1*.*

Proof: By Lemma 3.1 and Lemma 3.2, the time complexity of the construction of $\mathbb{A}$ is polynomial in the size of $\mathcal{M}_{\ell_1}$, and is thus pseudo-polynomial in the size of $\mathcal{M}$ and in the length of ℓ_1. So, as solving the SSP-E problem in $\mathcal{M}_{\ell_1}^{\mathbb{A}}$ is polynomial in the size of $\mathcal{M}_{\ell_1}^{\mathbb{A}}$, the overall algorithm presented to solve this problem is pseudo-polynomial in the size of $\mathcal{M}$ and in the size of the length of ℓ_1. Furthermore, as the strategy solving the SSP-E problem in $\mathcal{M}_{\ell_1}^{\mathbb{A}}$ is memoryless in $\mathcal{M}_{\ell_1}^{\mathbb{A}}$, it requires pseudo-polynomial memory in $\mathcal{M}$ (intuitively, the strategy has $\max_{(s,\ell)\in T_{\ell_1}} \ell$ modes).

□

Example 3.3 (SSP-WE *problem in the wireless sensor network*). Again, we get back to Example 3.1. We consider the MDP $\mathcal{M}$ of Figure 3.3. As in Example 3.2, we are interested by a strategy ensuring a duty cycle of 12 *ms* for the node n_0, but at the same time ensuring a minimal expected time to reach the *sleep* state, allowing it to go to sleep as soon as possible. Assume that we are interested by the following SSP-WE problem:

$$?\exists\sigma \;\; \mathbb{P}^\sigma_{s_0}(\Diamond_{\leq 12}\{s_2\}) = 1 \;\wedge\; \mathbb{E}^\sigma_{s_0}(\mathsf{TS}^{\{s_2\}}) \leq 6,$$

where 6 *ms* is half the time of the duty cycle that we want to ensure. First, we begin by unfolding $\mathcal{M}$ from s_0 up to the cost threshold $\ell_1 = 12$ (cf. Figure 3.5), yielding the MDP $\mathcal{M}_{\ell_1}$. Then, we compute iteratively $\mathbb{A}_i$ until reaching an index i^* such that $\mathbb{A}_{i^*} = \mathbb{A}_{i^*+1}$ (all sets $\mathbb{A}_j(s,v)$ such that $j \geq i^*$ are equal to $\mathbb{A}_{i^*}(s,v)$ for $(s,v) \in S_{\ell_1}$), and we thus have $\mathbb{A} = \mathbb{A}_{i^*}$. In our case, we compute $\mathbb{A}_i$ by dynamic programming (cf. Table 3.2), and we see that $\mathbb{A}_4 = \mathbb{A}_5$. Thus, we have $\mathbb{A} = \mathbb{A}_4$. Finally, we limit $\mathcal{M}_{\ell_1}$ to the safe actions of $\mathbb{A}$, yielding the MDP $\mathcal{M}_{\ell_1}^{\mathbb{A}}$ (cf. Figure 3.6). We solve the SSP-E problem in $\mathcal{M}_{\ell_1}^{\mathbb{A}}$ for s_0, the set of target states $T_{\ell_1} = \{(s_3,4), (s_3,8), (s_3,12)\}$, and the cost threshold $\ell_2 = 6$. In $\mathcal{M}_{\ell_1}^{\mathbb{A}}$, we can easily compute the minimal expected cost-

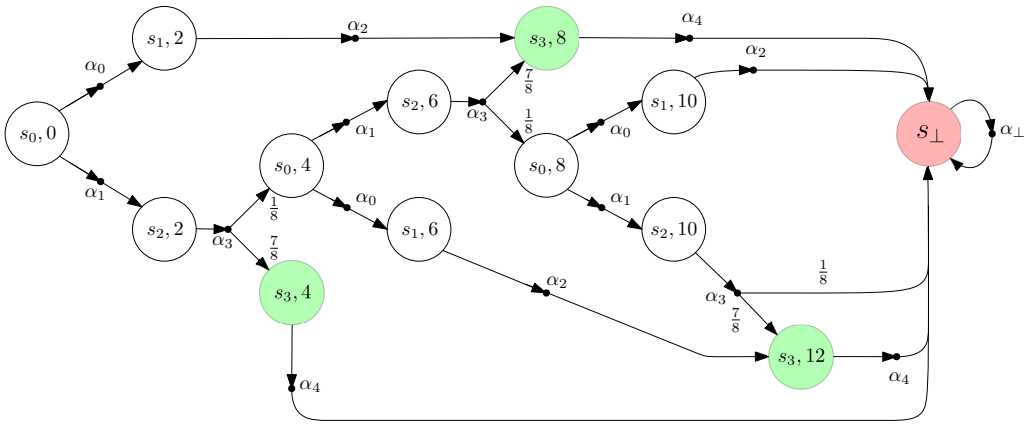


Figure 3.5. Unfolding of $\mathcal{M}$ (cf. Figure 3.3) from s_0 (i.e., from the active state) up to 12 ms

S_{ℓ_1}	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
s_0, 0	α_0, α_1	α_0, α_1	α_0, α_1	α_0, α_1	α_0, α_1	α_0, α_1
s_1, 2	α_2	α_2	α_2	α_2	α_2	α_2
s_2, 2	α_3	α_3	α_3	α_3	α_3	α_3
s_0, 4	α_1, α_0	α_1, α_0	α_1, α_0	α_1, α_0	α_0	α_0
s_3, 4	α_4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
s_1, 6	α_2	α_2	α_2	α_2	α_2	α_2
s_2, 6	α_3	α_3	α_3	$\varnothing$	$\varnothing$	$\varnothing$
s_0, 8	α_0, α_1	α_0, α_1	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
s_3, 8	α_4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
s_1, 10	α_2	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
s_2, 10	α_3	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
s_3, 12	α_4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
$s_\perp$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

Table 3.2. Computation of $\mathbb{A}_i$ by dynamic programming

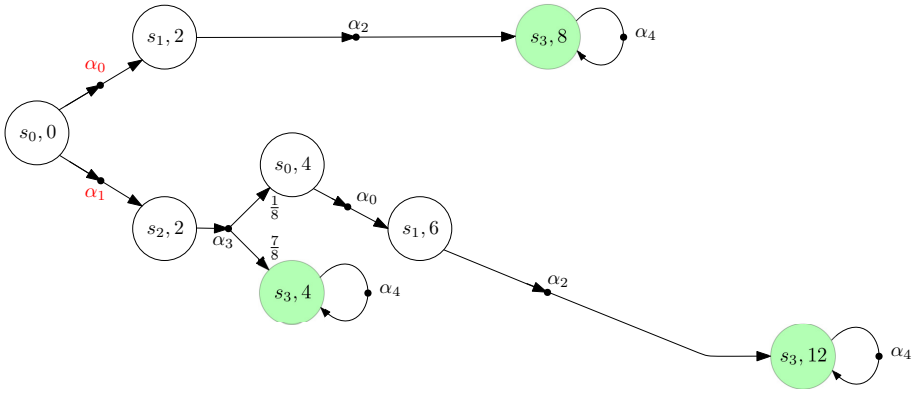


Figure 3.6. $\mathcal{M}_{\ell_1}$ limited to safe actions of $\mathbb{A}$

to-target for T_{ℓ_1} since there is actually only one nondeterministical choice: either the strategy chooses α_0 for the state $(s_0, 0)$ and the expected cost-to-target corresponding

to this choice is 8, or it chooses α_1 (we denote this strategy σ_{α_1}), and the expected cost-to-target is

$$\mathbb{E}^{\sigma_{\alpha_1}}_{(s_0,0)}(\mathsf{TS}^{T_{\ell_1}}) = \frac{7}{8} \cdot 4 + \frac{1}{8} \cdot 12 = 5.$$

Thus, the minimal expected cost from s to T while ensuring to reach T with probability one is $5 \leq 6$, and thus the SSP-WE problem is solved. The optimal pure memoryless strategy σ_{α_1} in $\mathcal{M}^{\mathbb{A}}_{\ell_1}$ corresponds in $\mathcal{M}$ to the memory strategy trying once α_1, and then choosing α_0 if it has failed, i.e., trying once a direct sending to the node n_2, and then sending the message via the intermediate node n_1 if it has failed.

3.3 Multi-objective reachability

Now, we interest us in strategies satisfying simultaneously multiple reachability problems. The problem consisting in deciding the existence of such a strategy is the *multi-objective stochastic reachability* [EKVY08]. In order to know if we can define a strategy satisfying simultaneously n reachability problems with given probabilities, we interest us in the simultaneously *achievable* vectors $(p_1, \ldots, p_n)$ of probabilities for these problems, i.e., such that there exists a strategy ensuring the reachability problem i with a probability p_i for all $i \in \{1, \ldots, n\}$. Furthermore, we interest us in the "trade-off" curve, named *Pareto curve*, being the set of optimal achievable vectors. Indeed, there can be some compromises between different reachability properties. For instance, assume we define a strategy ensuring the reachability to two set of target states T_1 and T_2 with probabilities (p_1, p_2). Modify the strategy to increase the probability p_1 of reaching the set T_1 may necessitate to lower the probability p_2 of reaching the set T_2.

Definition 3.3 (**Multi-objective stochastic reachability problem**)**.** Let $\mathcal{M}$ be an MDP with state space S, action space A, and probability transition function Δ, $s \in S$, and $r \in \mathbb{N}$ reachability properties. These properties are described by a set of target states $T_i \subseteq S$, and a probability threshold $\alpha_i \in [0,1] \cap \mathbb{Q}$, for each $i \in \{1, \ldots, r\}$. The *multi-objective stochastic reachability* problem (MO-SR, for short) consists in deciding if there exists a strategy σ satisfying

$$\bigwedge_{i=1}^{r} \mathbb{P}^{\sigma}_{s}(\Diamond T_i) \geq \alpha_i.$$

3.3.1 Pareto curve

We now define formally what are achievable vectors for a given MO-SR problem and what is the Pareto curve associated. We actually need these notions to introduce the resolution of all MO-SR problems. Indeed, solving such a problem requires to investigate a multi-objective optimisation problem that is highly linked to this Pareto curve.

Let $\mathcal{M}$ be an MDP with state space S, action space A, and transition probability function Δ, and $\mathcal{Q}_{s,r} := ?\exists\sigma \ \bigwedge_{i=1}^{r} \mathbb{P}^{\sigma}_{s}(\Diamond T_i) \geq \alpha_i$ be a MO-SR problem for $s \in S$, $r \in \mathbb{N}$, $(T_i)_{i\in\{1,\ldots,r\}} \subseteq S^r$, and $\alpha \in ([0,1] \cap \mathbb{Q})^r$.

Notation 3.2 (*Coordinate-wise inequality*). Let $p, p' \in ([0,1] \cap \mathbb{Q})^r$, we denote by $\leq$ and $\geq$ the coordinate-wise inequality relations between two vectors, i.e., $p \leq p'$ if and only if $p_i \leq p'_i$, and $p \geq p'$ if and only if $p_i \geq p'_i$, for all $i \in \{1, \ldots, r\}$.

Definition 3.4 (**Achievable vectors**). An *achievable vector of* $\mathcal{Q}_{s,r}$ is a vector $(p_i)_{i\in\{1,\ldots,r\}}$ such that there exists a strategy σ for $\mathcal{M}$ where $p_i = \mathbb{P}^\sigma_s(\Diamond T_i) \geq \alpha_i$ for all $i \in \{1, \ldots, r\}$. Furthermore, the *achievable set of* $\mathcal{Q}_{s,r}$ is given by

$$U_{\mathcal{Q}_{s,r}} = \{\, p \in ([0,1]) \cap \mathbb{Q})^r \mid \exists \sigma,\ p = (\mathbb{P}^\sigma_s(\Diamond T_i))_{i\in\{1,\ldots,r\}} \ \wedge\ p \geq \alpha \}$$

Definition 3.5 (**Pareto optimal vector**). Let $p \in U_{\mathcal{Q}_{s,r}}$ be an achievable vector of $\mathcal{Q}_{s,r}$. This vector p is *Pareto optimal* if and only if there does not exists another vector domininating p, i.e., $\neg\exists p' \in U_{\mathcal{Q}_{s,r}}$ such that $p \leq p'$ and $p \neq p'$.

Definition 3.6 (**Pareto curve**). The *Pareto curve* $\mathcal{P}_{\mathcal{Q}_{s,r}}$ of the MO-SR problem $\mathcal{Q}_{s,r}$ is the set of Pareto optimal vectors inside $U_{\mathcal{Q}_{s,r}}$, i.e.,

$$\mathcal{P}_{\mathcal{Q}_{s,r}} = \{p \in U_{\mathcal{Q}_{s,r}} \mid \neg\exists p' (p' \in U_{\mathcal{Q}_{s,r}} \wedge p \leq p' \wedge p \neq p')\}.$$

For a given MO-SR problem, the associated Pareto curve is in general an infinite set. Actually, this Pareto optimal set is in general a polyhedral set. A way to solve the MO-SR would be to enumerate all the vertices of the polytope defining the Pareto curve, or enumerating its facets, but it is not possible to do this in polynomial time (the vertex enumeration problem for polyhedra is NP-hard [BEGM09]). However, the Pareto curve can be efficiently approximated with an approximation factor $\epsilon > 0$ [PY00].

Definition 3.7 (**Approximated Pareto curve**). Let $\epsilon > 0$, the *ϵ-approximated Pareto curve* of the MO-SR problem $\mathcal{Q}_{s,r}$ is the set of achievable vectors $\mathcal{P}_{\mathcal{Q}_{s,r}}(\epsilon) \subseteq U_{\mathcal{Q}_{s,r}}$ such that, for all achievable vector $p' \in U_{\mathcal{Q}_{s,r}}$, there exists an ϵ-optimal vector $p \in \mathcal{P}_{\mathcal{Q}_{s,r}}(\epsilon)$ such that $p' \leq (1+\epsilon)p$.

Notation 3.3 (MO-SR *context*). When the MO-SR problem considered is clear from the context, we use $\mathcal{P}$, and $\mathcal{P}(\epsilon)$ to denote respectively the Pareto curve, and ϵ-approximated Pareto curve.

Example 3.4 (*Pareto curve for a simple* MO-SR *problem* [EKVY08]). Let $\mathcal{M}$ be the MDP of Figure 3.7. Let $P_1 = \{s_2\}$ and $P_2 = \{s_3\}$. We are interested in the Pareto optimal vectors of the following MO-SR problem: $\mathcal{Q} :=?\exists\sigma\ \bigwedge_{i=1}^{2} \mathbb{P}^\sigma_{s_0}(\Diamond P_i) \geq 0$. These Pareto optimal vectors describe the probabilities of strategies offering the *optimal compromises* of reaching P_1 and P_2, i.e., the dominant probabilities of $\mathcal{Q}$. We start from the state s_0 with enabled actions α_1, α_2, and α_3. By choosing α_1, the system goes to P_1 with probability 0.6 and P_2 with probability zero. Then, if α_2 is chosen, the system goes to P_1 and P_2 with probability $\frac{1}{2}$. Finally, if α_3 is chosen, the system goes to P_1 with probability zero and P_2 with probability 0.8. These three choices lead to the three Pareto optimal vectors $(0, 0.6)$, $(0.5, 0.5)$, and $(0.8, 0)$. Indeed, none of these vectors can be dominated by another vectors yielded from another strategies. The three vertices of the Pareto curve $\mathcal{P}$ of Figure 3.7 describe these three Pareto optimal vectors.

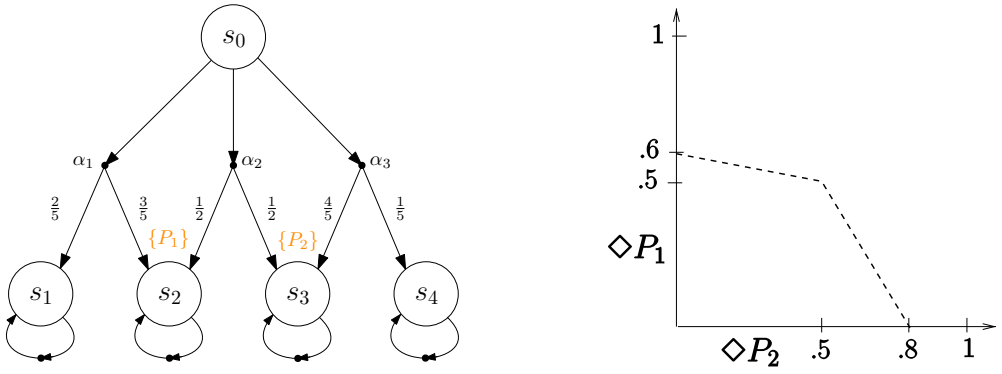


Figure 3.7. MDP with $P_1 = \{s_2\}$, $P_2 = \{s_3\}$, two reachability events, $\Diamond P_1$ and $\Diamond P_2$, and the associated Pareto curve $\mathcal{P}$, defined by the points on the dotted lines

3.3.2 Randomised strategies

Before detailing an algorithm to solve the MO-SR problem for a given state and given reachability properties in an MDP, we are going to interest us in type of strategies required for such a problem. Indeed, let $\mathcal{M}$ be the MDP of Figure 3.8 and consider the MO-SR problem $\mathcal{Q} :=? \exists \sigma \ \bigwedge_{i=1}^{3} \mathbb{P}_{s_0}^{\sigma}(\Diamond T_i) = 1$. It is clear that this problem requires memory. Indeed, starting from s_0, as T_1 and T_2 have to be almost surely reached, the actions α_1 and α_2 have to be chosen by the strategy while T_1 or T_2 has not been reached yet. When T_1 and T_2 have been reached, the strategy switches mode and chooses the action α_3 in order to almost surely reach T_3. Actually, memory is not

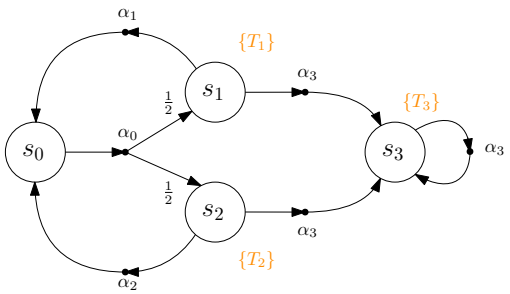


Figure 3.8. MDP requiring memory to satisfy $\mathcal{Q} :=? \exists \sigma \ \bigwedge_{i=1}^{3} \mathbb{P}_{s_0}^{\sigma}(\Diamond T_i) = 1$

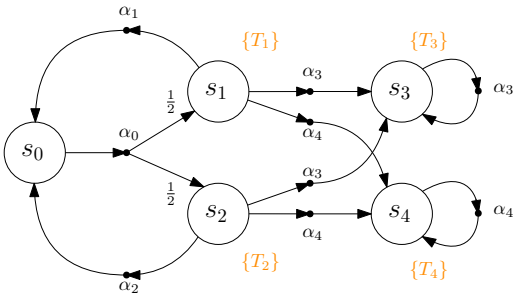


Figure 3.9. MDP requiring both memory and randomisation to satisfy $\mathcal{Q} :=? \exists \sigma \ \bigwedge_{i=1}^{2} \mathbb{P}_{s_0}^{\sigma}(\Diamond T_i) = 1 \wedge \bigwedge_{i=3}^{4} \mathbb{P}_{s_0}^{\sigma}(\Diamond T_i) \geq \frac{1}{2}$

sufficient in some cases. Indeed, assume that we slightly modify $\mathcal{M}$ (cf. Figure 3.9) and consider the MO-SR problem $\mathcal{Q} :=? \exists \sigma \ \bigwedge_{i=1}^{2} \mathbb{P}_{s_0}^{\sigma}(\Diamond T_i) = 1 \wedge \bigwedge_{i=3}^{4} \mathbb{P}_{s_0}^{\sigma}(\Diamond T_i) \geq \frac{1}{2}$. Again, a strategy σ satisfying $\mathcal{Q}$ has to choose actions α_1 and α_2 until that T_1 and T_2 are reached. Then, the strategy switches mode in order to satisfy $\bigwedge_{i=3}^{4} \mathbb{P}_{s_0}^{\sigma}(\Diamond T_i) \geq \frac{1}{2}$. Here, deterministically choosing between α_1 and α_2 is not sufficient: choosing α_3 almost surely ensures to reach T_3 but implies reaching T_4 with probability zero, and choosing α_4 implies reaching T_3 with probability zero but almost surely ensures reaching T_4. The optimal solution consists in choosing α_3 and α_4 with probability $\frac{1}{2}$.

Such strategies, choosing actions randomly, are *randomised strategies*. In fact, getting back to Example 3.4, points on the dotted lines of the Pareto curve of Figure 3.7 refer to strategies using randomisation.

Definition 3.8 (**Randomised finite-memory strategies**)**.** Let $\mathcal{M}$ be an MDP with state space S and action space A. A *finite-memory randomised strategy* $\sigma = (Q, \sigma_{act}, \delta, \delta_0)$ is a *stochastic Moore machine*, where

- Q is a finite set of *modes*,
- $\sigma_{act} : Q \times S \to \mathcal{D}(A)$ is the *stochastic next action function* giving, for each $s \in S$, a probability distribution on actions of $A(s)$ following a mode $q \in Q$ in which the machine is currently,
- $\delta : Q \times S \to \mathcal{D}(Q)$ is the *stochastic transition function*,
- $\delta_0 : S \to Q$ is the *initialisation function*, defined as for pure finite-memory strategies, giving the initial mode $q \in Q$, following a state $s \in S$ from which the machine is initialised.

According to this definition, the probability distribution defined on the successors of each state s of a Markov chain induced by such a strategy additionally depends on the probability distribution defined on enabled actions of s given by the stochastic next action function as well as the probability distribution defined on modes of the Moore Machine given by the stochastic transition function.

Definition 3.9 (**Markov chain induced by a randomised finite-memory strategy**)**.** Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP and $\sigma = (Q, \sigma_{act}, \delta, \delta_0)$ be a randomised finite-memory strategy for $\mathcal{M}$. The product of $\mathcal{M}$ by σ is given by

$$\mathcal{M} \otimes \sigma = \mathcal{M}^\sigma = (S \times Q, \Delta^\sigma, AP, L^\sigma)$$

where $\mathcal{M}^\sigma$ is the MC induced by the randomised finite-memory strategy σ and where, for all states $s, s' \in S$, and for all modes $q, q' \in Q$,

- $\Delta^\sigma((s,q),(s',q')) = \delta(q,s)(q') \cdot \sum_{\alpha \in A(s)} \sigma_{act}(q,s)(\alpha) \cdot \Delta(s,\alpha,s')$
- $L^\sigma(s,q) = L(s)$.

Remark 3.2 (*Weight function in an MC induced by a randomised strategy*). We do not consider the weight function of any induced MC by a randomised strategy σ in this context. Indeed, the weight of a transition depends on the action chosen by the strategy. Here, as actions are chosen randomly following a probability distribution defined on enabled actions of each state, this information is lost in the induced MC. It is however possible to define a weight function referring to the expected weight of each transition in the induced MC, but this definition is irrelevant in this context.

As for pure strategy, there exists a subset of randomised finite-memory strategies with only one mode for a given MDP. These randomised strategies are said to be *memoryless*.

Definition 3.10 (**Randomised memoryless strategy**)**.** Let $\mathcal{M}$ be an MDP with state space S and action space A. A randomised memoryless strategy σ is a function $\sigma : S \to \mathcal{D}(A)$, giving, following a state $s \in S$, a probability distribution defined on the enabled action of s.

3.3.3 Multi-objective reachability with absorbing target states

In this subsection, we present an efficient way to solve an MO-SR problem where the set of target states are only composed of absorbing states. More formally, let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP, and $\mathcal{Q}_{s^*,r} := ?\exists\sigma \ \bigwedge_{i=1}^{r} \mathbb{P}^{\sigma}_{s^*}(\Diamond T_i) \geq \alpha_i$ be a MO-SR problem for $s^* \in S$, $r \in \mathbb{N}$, $(T_i)_{i\in\{1,\dots,r\}} \subseteq S^r$, and $\alpha \in ([0,1] \cap \mathbb{Q})^r$. We assume that for all $i \in \{1,\dots,r\}$ and for all targets $t \in T_i$, t is absorbing, i.e., there exists an enabled action of t, $\alpha \in A(t)$, such that $\Delta(t,\alpha,t) = 1$.

In order to solve $\mathcal{Q}_{s^*,r}$, we need to do some preprocessing on $\mathcal{M}$, to ignore some useless and bad states. Let $T = \bigcup_{i\in\{1,\dots,r\}} T_i$ and $S_{=0} = \{s \in S \mid \forall\sigma,\ \mathbb{P}^{\sigma}_{s}(\Diamond T) = 0\}$. Remind that this set can be computed efficiently in polynomial time in the size of $\mathcal{M}$ by checking if each state s is connected to T in the underlying graph of $\mathcal{M}$. Let $\Delta_{>0} : S \times A \times S$ be the *safe transition function* of $\mathcal{M}$, $s, s' \in S$, and $\alpha \in A(s)$, this function is defined as follows:

$$\Delta_{>0}(s,\alpha,s') = \begin{cases} \Delta(s,\alpha,s') & \text{if } s \notin S_{=0} \text{ and } s' \notin S_{=0}, \\ 0 & \text{else.} \end{cases}$$

Let $s \in S$ and $\alpha \in A(s)$, by considering this transition function, we have $\sum_{s'\in S} \Delta_{>0}(s,\alpha,s') \leq 1$, and furthermore, $\sum_{s'\in S_{=0}} \Delta(s,\alpha,s') = 1 - \sum_{s'\in S} \Delta_{>0}(s,\alpha,s')$.

Notation 3.4 (*Cleaned-up MDP*). Defined as above, we say that the safe transition function $\Delta_{>0}$ *clean-up* $\mathcal{M}$ for $T = \bigcup_{i\in\{1,\dots,r\}} T_i$.

We are going to define a *multi-objective linear program* (MOLP, for short) to solve $\mathcal{Q}_{s^*,r}$. An MOLP is defined the same way as a classical LP except that the optimisation consists in simultaneously maximising (or minimising) multiple objective functions. A feasible solution of an MOLP is *efficient* if it is a *dominant solution*, i.e., where neither value of objective function can be optimised without degrading the value of another objective function. Thus, the set of efficient solutions of an MOLP is a Pareto curve.

We assume that $\mathcal{M}$ is cleaned-up for $T = \bigcup_{i=1}^{r} T_i$ with $\Delta_{>0}$. Let $(y_t)_{t\in T} \in \mathbb{Q}^r_{\geq 0}$, and $(y_{s,\alpha})_{s\in S\setminus T,\, \alpha\in A(s)} \in \mathbb{Q}^r_{\geq 0}$. With these variables, we define the MOLP of Figure 3.10. This MOLP is derived from the dual LP of the standard LP for single-objective reachability obtained from Bellman's equation for optimal reachability (cf. Theorem 2.5 and Appendix A.3.1).

$$\textbf{Objectives } i \in \{1,\dots,r\} \ : \ \max \sum_{t \in T_i} y_t$$

subject to the following constraints:

$$\mathbb{1}(s) + \sum_{s' \in S} \sum_{\alpha' \in A(s')} \Delta_{>0}(s', \alpha', s) \cdot y_{s',\alpha'} = \sum_{\alpha \in A(s)} y_{s,\alpha} \qquad \forall s \in S \setminus T \tag{3.3}$$

$$\sum_{s' \in S \setminus T} \sum_{\alpha' \in A(s')} \Delta_{>0}(s', \alpha', t) \cdot y_{s',\alpha'} = y_t \qquad \forall t \in T \tag{3.4}$$

$$y_{s,\alpha} \geq 0 \qquad \forall s \in S \setminus T \text{ and } \alpha \in A(s)$$

$$y_t \geq 0 \qquad \forall t \in T$$

$$\text{where } \mathbb{1}(s) = \begin{cases} 1 & \text{if } s = s^*, \\ 0 & \text{else.} \end{cases}$$

Figure 3.10. MOLP for the MO-SR problem $\mathcal{Q}_{s^*,r}$

Theorem 3.4 (*Feasible solution and satisfying strategy*). *Let $\mathcal{M}$ be an MDP with state space S, $s^* \in S$, multiples target sets $T_i \subseteq S$, for $i \in \{1,\dots,r\}$, where every target $t \in \bigcup_{i=1}^{r} T_i$ is an absorbing state, and a vector of probability thresholds $\alpha \in ([0,1] \cap \mathbb{Q})^r$. The two following propositions are equivalent:*

(a.) There exists a randomised memoryless strategy σ such that

$$\bigwedge_{i=1}^{r} \mathbb{P}_s^{\sigma}(\Diamond T_i) \geq \alpha_i.$$

(b.) There is a feasible solution y' for the MOLP in Figure 3.10 such that

$$\bigwedge_{i=1}^{r} \sum_{t \in T_i} y'_t \geq \alpha_i.$$

A proof of this theorem is provided in [EKVY08]. Let y' be a feasible solution of the MOLP in Figure 3.10, we intuitively explain it as follows:

- $y'_{s,\alpha}$, with $s \in S \setminus T$, $\alpha \in A(s)$, is the *frequency* of the transition $s \xrightarrow{\alpha}$, i.e., the *"expected number of time that the action α is chosen to reach T when the system is in the state s"*.

- $\sum_{\alpha \in A(s)} y'_{s,\alpha}$, with $s \in S \setminus T$, is thus the *visit frequency* of the state s, i.e., the *"expected number of times that the state s is visited to reach T"* (cf. constraint 3.3).

- y'_t, with $t \in T$, is the visit frequency of the state t *by states other than t*. Since t is absorbing, this visit frequency can be translated into the probability of eventually reaching T (cf. constraint 3.4).

Moreover, we can build a randomised strategy σ from any feasible solution y' of this MOLP. Indeed, let $S_{freq>0} = \{s \in S \setminus T \mid \sum_{\alpha \in A(s)} y'_{s,\alpha} > 0\}$ be the set of states with a nonzero visit frequency, and $s \in S$, $\alpha \in A(s)$,

$$\sigma(s)(\alpha) = \begin{cases} \dfrac{y'_{s,\alpha}}{\sum_{\alpha' \in A(s)} y'_{s,\alpha'}} & \text{if } \ s \in S_{freq>0}, \\ \mu(s)(\alpha) & \text{else,} \end{cases} \tag{3.5}$$

where $\mu : S \to \mathcal{D}(A)$, and $\mu(s)$ is an arbitrary distribution on enabled actions of s.

Corollary 3.1 (*Multi-objective reachability with absorbing target states*). *Let $\mathcal{M}$ be an MDP with state space S, and $\mathcal{Q}_{s^*,r}$ be the* MO-SR *problem for the state $s^* \in S$ and the $r \in \mathbb{N}$ constraints described by target sets $T_i \subseteq S$, for $i \in \{1, \ldots, r\}$, where every target $t \in \bigcup_{i=1}^{r} T_i$ is an absorbing state, and probability thresholds $\alpha \in ([0,1] \cap \mathbb{Q})^r$.*

(a.) The MO-SR *problem $\mathcal{Q}_{s^*,r}$ can be decided in polynomial time in the size of $\mathcal{M}$ and in the number of set of target states to reach (i.e., r), and if a strategy exists for this problem, we can build a randomised memoryless strategy that satisfies it.*

(b.) For $\epsilon > 0$, we can compute an ϵ-approximated Pareto curve $\mathcal{P}(\epsilon)$ for the MO-SR *problem $\mathcal{Q}_{s^*,r}$ in polynomial time in the size of $\mathcal{M}$ and $\frac{1}{\epsilon}$.*

Proof: For (a.), consider the MOLP in Figure 3.10 and add the constraint $\sum_{t \in T_i} y_t \geq \alpha_i$ for each $i \in \{1, \ldots, r\}$. Then, forget the r objective functions of this MOLP and consider the unique objective function $\max \sum_{i=1}^{r} \sum_{t \in T_i} y_t$. This yields a classical LP. If there exists an optimal solution to this LP, then we can build a randomised strategy from this solution (cf. Equation 3.5).
The part. (b.) directly follows from Theorem 3.4 and the results of [PY00].

□

3.3.4 General multi-objective reachability

We have presented a way to solve an MO-SR problem only involving sets of absorbing target states. We present in this section a more general approach, without the absorbing restriction for target states.

Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP, and $\mathcal{Q}_{s^*,r} := ?\exists\sigma \ \bigwedge_{i=1}^{r} \mathbb{P}^{\sigma}_{s^*}(\Diamond T_i) \geq \alpha_i$ be a MO-SR problem for $s^* \in S$, $r \in \mathbb{N}$, $(T_i)_{i \in \{1,\ldots,r\}} \subseteq S^r$, and $\alpha \in ([0,1] \cap \mathbb{Q})^r$. The technique we use to solve $\mathcal{Q}_{s^*,r}$ is inspired of which used in [EKVY08] to solve a multi-objective model-checking query for multiple *ω-regular properties* by doing the product of $\mathcal{M}$ with multiple *Büchi automata* representing these properties. Our technique requires to introduce the notion of *end-components* of an MDP [BK08].

Definition 3.11 (End component). An *end-component* (EC, for short) of an MDP $\mathcal{M} = (S, A, \Delta, w, AP, L)$ is an MDP $\mathcal{C} = (S^*, A^*, \Delta^*)$ such that

- $S^* \subseteq S$ and $S^* \neq \emptyset$,
- for all $s \in S^*$, $A^*(s) \subseteq A(s)$, $A^*(s) \neq \emptyset$, and for each $\alpha \in A^*(s)$, $Succ(s, \alpha) \subseteq S^*$,
- $\Delta^* = \Delta|_{S^* \times A^*}$, and
- $\mathcal{C}$ is *strongly connected*, i.e., there exists a path in $\mathcal{C}$ between any pair of states in S'.

Definition 3.12 (**Maximal end-component**). $\mathcal{C} = (S^*, A^*, \Delta^*)$ is a *maximal end-component* of $\mathcal{M}$ if there is no end-component $\mathcal{C}' = (S', A', \Delta')$ such that $(S^*, A^*) \neq (S', A')$ and $S^* \subseteq S'$ and $A^*(s) \subseteq A'(s)$ for all $s \in S^*$. We let $\mathsf{MEC}(\mathcal{M})$ denote the set of maximal ECs of $\mathcal{M}$, computable in polynomial time in the size of $\mathcal{M}$ with purely graph theory algorithms.

The key properties of ECs that we are going to exploit in order to build a satisfying strategy for any MO-SR are the following.

Lemma 3.3 (Recurrence property of end components). *For end component $\mathcal{C} = (S^*, A^*, \Delta^*)$ of MDP $\mathcal{M}$, there exists a finite-memory strategy σ for $\mathcal{M}$ such that for any $s \in S^*$:*

$$\mathbb{P}_s^\sigma(\{\pi \in Paths(s) \mid \pi \models \Box S^* \ \wedge \ inf(\pi) = S^*\}) = 1.$$

Theorem 3.5 (Limiting behaviour of MDPs). *For each state s of a finite MDP $\mathcal{M}$ and strategy σ for $\mathcal{M}$:*

$$\mathbb{P}_s^\sigma(\{\pi \in Paths(s) \mid inf(\pi) \text{ is the state space of an EC of } \mathcal{M}\}) = 1,$$

i.e., whatever the strategy, each path starting in s ends up in an EC of $\mathcal{M}$ almost surely.

Thus, any strategy of $\mathcal{M}$ almost surely ensures to reach an EC of $\mathcal{M}$, and once a state of this EC $\mathcal{C}$ is reached in $\mathcal{M}$, there exists a strategy that almost surely enforces staying forever in $\mathcal{C}$ while visiting all of its states infinitely often.

For each reachability property $i \in \{1, \ldots, r\}$, we define a *reachability transition function* $\delta_i : \{0,1\} \times S \to \{0,1\}$ as follows:

$$\delta_i(q, s) = \begin{cases} 1 & \text{if } q = 1 \text{ or } s \in T_i \\ 0 & \text{else.} \end{cases}$$

In order to solve $\mathcal{Q}_{s^*,r}$, we build a new MDP $\mathcal{M}' = (S', A', \Delta')$ with each of these reachability transition functions, defined as follows:

- $S' \subseteq (S \cup \{s_0\}) \times \{0,1\}^r$ is composed of states $(s, q_1, \ldots, q_r)$ such that $s \in S$, $(q_1, \ldots, q_i) \in \{0,1\}^r$, intuitively recording through the boolean variable q_i if T_i has been reached along a path starting from s^*, for each $i \in \{1, \ldots, r\}$. We additionally add a dummy state $(s_0, 0, \ldots, 0)$ in S'.

- $A' = A \cup \{\alpha_0\}$ and $A'(s, q_1, \ldots, q_r) = A(s)$ for all $(s, q_1, \ldots, q_r) \in S'$, and $A'(s_0, 0, \ldots, 0) = \{\alpha_0\}$.

- Δ' is a probability transition function, defined as follows: let $x = (s, q_1, \ldots, q_r) \in S'$, $x' = (s', q'_1, \ldots, q'_r) \in S'$, and $\alpha \in A'(x)$,

$$\Delta(x, \alpha, x') = \begin{cases} 1 & \text{if } s = s_0,\ s' = s^*, \text{ and } \forall i \in \{1, \ldots, q\},\ \delta_i(q_i, s) = q'_i, \\ \Delta(s, \alpha, s') & \text{if } s, s' \in S \text{ and } \forall i \in \{1, \ldots, q\},\ \delta_i(q_i, s) = q'_i, \\ 0 & \text{otherwise.} \end{cases}$$

  This probability transition function suggests the additional dummy transition $(s_0, 0, \ldots, 0) \xrightarrow{\alpha_0} (s^*, \delta_1(0, s^*), \ldots, \delta_r(0, s^*))$, with probability one. If this transition is not added, s^* is never read by the reachability transition function δ_i, which could be a problem if $s^* \in T_i$ for some $i \in \{1, \ldots, r\}$.

Furthermore, as we consider paths of $\mathcal{M}$ starting from s^*, we limit the state space S' with the states reachable from $(s_0, 0, \ldots, 0)$.

Definition 3.13 (**Favoured reachability property**)**.** Let $\mathcal{C} = (S^*, A^*, \Delta^*)$ be an EC of $\mathcal{M}$, we say that $\mathcal{C}$ favours a set of reachability property $R \subseteq \{1, \ldots, r\}$ if and only if for all $(s, q_1, \ldots, q_r) \in S^*$ and for all $i \in R$, $q_i = 1$.

Lemma 3.4 (Favoured reachability and property of end-components)**.** *Let $\mathcal{C} = (S^*, A^*, \Delta^*)$ be an EC of $\mathcal{M}'$, and $(s, q_1, \ldots, q_r) \in S^*$ be any state of $\mathcal{C}$, then $\mathcal{C}$ favours all the subsets of $R = \{i \in \{1, \ldots, r\} \mid q_i = 1\}$ (R included). Moreover, for all $(s', q'_1, \ldots, q'_r) \in S^*$, $q'_i = 0$ for $i \in \{1, \ldots, r\} \setminus R$.*

Proof: Let $(s, q_1, \ldots, q_r) \in S^*$ be any state of $\mathcal{C}$, $R = \{i \in \{1, \ldots, r\} \mid q_i = 1\}$, $S_R = \{(s, q_1, \ldots, q_r) \in S' \mid \forall i \in R,\ q_i = 1\}$ and $s_R = (s, q_1, \ldots, q_r) \in S_R$. Actually, by definition of Δ' and δ_i for all $i \in \{1, \ldots, r\}$, for all $\alpha \in A(s_R)$, $Succ(s_R, \alpha) \subseteq S_R$. Thus, as any EC is strongly connected, if there exists a state $(s, q_1, \ldots, q_r) \in S^*$ such that $q_i = 1$ for all $i \in R$, then $S^* \subseteq S_R$. (1.)
Moreover, let $i \in \{1, \ldots, r\} \setminus R$, and assume we have a state $u = (s, q_1, \ldots, q_r) \in S^*$ with $q_i = 0$ such that there exists an α-successor $(s', q'_1, \ldots, q'_r) \in Succ(u, \alpha)$ for $\alpha \in A^*(s)$, with $q'_i = 1$, then, by (1.), $q_i = 1$, that leads a contradiction. □

Lemma 3.5 (Multiple reachability in $\mathcal{M}'$)**.** *Defined as above, $\mathcal{M}'$ has the following properties: for each $R \in 2^{\{1,\ldots,q\}}$, there is a subset of target states $T_R \subseteq S'$ that satisfies the following conditions:*

(1.) If a path starting from $(s_0, 0, \ldots, 0)$ visit a state in T_R at some point, then all the sets T_i, $i \in R$, have been reached along this path.

(2.) For every strategy, the set of paths reaching all sets of target states T_i, $i \in R$, and never visiting some state of T_R has probability zero.

Proof: Let $R \in 2^{\{1,\dots,r\}}$. We are going to look at all maximal end-components of $\mathcal{M}'$ favouring R in order to define T_R. More formally, let $S_R = \{(s, q_1, \dots, q_r) \in S' \mid \forall i \in R,\ q_i = 1\}$ and $S_R^{\mathcal{C}} = \{S^* \subseteq S_R \mid \exists \mathcal{C} \in \mathsf{MEC}(\mathcal{M}') \text{ with state space } S^*\}$,

$$T_R = \bigcup_{S^* \in S_R^{\mathcal{C}}} S^*.$$

As $\mathsf{MEC}(\mathcal{M}')$ can be computed in polynomial time in the size of $\mathcal{M}'$, we can compute at a time all sets T_R for all $R \in \{1, \dots, r\}$.
By definition of Δ' and by definition of δ_i, for each $i \in R$, if a state $t = (s, q_1, \dots, q_r) \in S'$ is visited at some point with $s \in T_i$, then the boolean value q_i is updated with 1 (this is actually the only condition needed for q_i to go from 0 to 1) and remains 1 forever, i.e., for all $\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \cdots \in Paths(t)$, for all $k \in \mathbb{N}$, we have $s_k = (s', q'_1, \dots, q'_r)$, with $q'_i = 1$. (*)
Thus, assume that a state in T_R is reached along a path $\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \cdots \in Paths((s_0, 0, \dots, 0))$, i.e., assume there exists an index $k \in \mathbb{N}$ such that $s_k = (s, q_1, \dots, q_r) \in T_R$. We have by definition of T_R that $q_i = 1$ for all $i \in R$. Then, as the path starts from $(s_0, 0, \dots, 0)$ and by (*), there exists for all $i \in R$ an index $k' \leq k$ such that $s_{k'} = (s', q'_1, \dots, q'_r)$ with $s' \in T_i$. We have shown (1.).
Now, let σ be a strategy for $\mathcal{M}'$ and $\pi \in Paths^{\sigma}((s_0, 0, \dots, 0))$. Assume that all the sets of target states T_i have been reached for $i \in R$ along π. By Lemma 3.5, this path ends up in an EC almost surely. By (*), we necessarily have that this EC favours R, as all the target states T_i have been reached. Thus, as T_R is the union of the state space of each EC favouring R,

$$\mathbb{P}^{\sigma}_{(s_0,0,\dots,0)}(\{\pi \in Paths((s_0, 0, \dots, 0)) \mid inf(\pi) \subseteq T_R\}) = 1,$$

which directly implies

$$\mathbb{P}^{\sigma}_{(s_0,0,\dots,0)}(\{\pi \in Paths((s_0, 0, \dots, 0)) \mid inf(\pi) \cap T_R = \emptyset\}) = 0.$$

We are done with (2.). That concludes the proof. □

The Lemma 3.4 and the Lemma 3.5 allow us to define an algorithm ensuring to solve any MO-SR problem (cf. Algorithm 6). The key idea behind this algorithm is that we add a new absorbing state s_R in $\mathcal{M}'$ for each subset of constraints $R \subseteq \{1, \dots, r\}$. Then, for each state $t \in S'$, and for each maximal subset R such that $t \in T_R$, we add a new transition from t to s_R. By considering the new sets of target states $F_i = \{s_R \mid i \in R\}$, for $i \in \{1, \dots, r\}$, we solve the MO-SR problem for these sets of absorbing target states and the thresholds $\alpha \in ([0, 1] \cap \mathbb{Q})^r$ as described in Subsection 3.3.3.

Lemma 3.6 (Recover a strategy for $\mathcal{M}$ from $\mathcal{M}'$)**.** *The set of achievable points $U_{\mathcal{Q}_{s^*,r}}$ is equal to $U_{\mathcal{Q}_{(s_0,0,\dots,0),r}}$. Moreover, from a randomised memoryless strategy σ' that satisfies $\mathcal{Q}_{(s_0,0,\dots,0),r}$ in $\mathcal{M}'$, we can recover a randomised finite-memory strategy σ that satisfies $\mathcal{Q}_{s^*,r}$ in $\mathcal{M}$.*

Algorithm 6 Multi-objective reachability

Input : $\mathcal{M} = (S, A, \Delta, AP, L)$, a finite MDP, and $\mathcal{Q}_{s^*,r}$, a MO-SR problem.
Output : A strategy satisfying $\mathcal{Q}_{s^*,r}$ if it is possible to answer $True$ to $\mathcal{Q}_{s^*,r}$, $False$ else.

build $\mathcal{M}' = (S', A', \Delta')$ with δ_i, $i \in \{1, \dots r\}$
for $R \in 2^{\{1,\dots,r\}}$ **do**
  $T_R \leftarrow \emptyset$
for $\mathcal{C} = (S^*, A^*, \Delta^*) \in \mathsf{MEC}(\mathcal{M}')$ **do**
  let $(s, q_1, \dots, q_r)$ be an arbitrary state of S^* *(* cf. Lemma 3.4 *)*
  $R^* \leftarrow \{i \in \{1, \dots, r\} \mid q_i = 1\}$ *(* maximal subset of $\{1, \dots, r\}$ favoured by $\mathcal{C}$ *)*
  for $R \subseteq R^*$ **do**
    $T_R \leftarrow T_R \cup S^*$
for $R \in 2^{\{1,\dots,r\}} \setminus \emptyset$ **do**
  $S' \leftarrow S' \cup \{s_R\}$
  $A' \leftarrow A' \cup \{\alpha_R\}$; $A'(s_R) \leftarrow \{\alpha_R\}$
  add the transition $s_R \xrightarrow{\alpha_R} s_R$ in $\mathcal{M}'$ with $\Delta'(s_R, \alpha_R, s_R) = 1$
for $t \in \bigcup_{\mathcal{C}=(S^*,A^*,\Delta^*) \in \mathsf{MEC}(\mathcal{M}')} S^*$ **do**
  let $R \in 2^{\{1,\dots,r\}}$ be the maximal subset of $\{1, \dots, r\}$ such that $t \in T_R$
  $A'(t) \leftarrow A'(t) \cup \{\alpha_R\}$
  add the transition $t \xrightarrow{\alpha_R} s_R$ in $\mathcal{M}'$ with $\Delta'(t, \alpha_R, s_R) = 1$
for all $i \leftarrow 1$ to r **do**
  $F_i \leftarrow \{s_R \mid i \in R\}$
let $\mathcal{Q}_{(s_0,0\dots,0),r}$ be the MO-SR problem for the set of **absorbing target states** F_i and cost thresholds α_i for $i \in \{1, \dots, r\}$ in $\mathcal{M}'$, from the state $(s_0, 0, \dots, 0)$.
solve $\mathcal{Q}_{(s_0,0,\dots,0),r}$ *(* cf. Corollary 3.1 *)*
if there exists a strategy σ' for $\mathcal{Q}_{(s_0,,0,\dots,0),r}$ **then**
  build σ' for $\mathcal{M}'$ *(* cf. Equation 3.5 *)*
  return σ'
else
  return $False$

Proof: Let σ' be a strategy for the MDP $\mathcal{M}'$ of Algorithm 6. Assume that $\mathcal{Q}_{(s_0,0,\dots,0),r}$ is satisfied by σ'. Thus, we have $\bigwedge_{i=1}^{r} \mathbb{P}^{\sigma'}_{(s_0,0,\dots,0)}(\Diamond F_i) \geq \alpha_i$. The strategy σ for $\mathcal{M}$ skips the first choice of σ' (necessarily being α_0), and follows then the choices of σ' in $\mathcal{M}$ from s^* until just before it transitions to a state s_R, at which point it must be in a state $t \in T_R$, with R being a maximal set of $\{R' \subseteq \{1, \dots, r\} \mid t \in T_{R'}\}$, and at that point our strategy σ switches to the strategy allowing to almost surely visiting all states of T_R. That guarantees that $\mathbb{P}^{\sigma}_{s^*}(\Diamond T_i) \geq \mathbb{P}^{\sigma'}_{(s_0,0,\dots,0)}(\Diamond F_i) \geq \alpha_i$, for all $i \in \{1, \dots, r\}$.

□

Theorem 3.6 (*Solving the* MO-SR *problem*). *Given an MDP $\mathcal{M}$ with state space S, we can solve the* MO-SR *problem for the state $s \in S$, the $r \in \mathbb{N}$ reachability properties described by the set of target states $T_i \subseteq S$ and the probability threshold α_i for each $i \in \{1, \dots, r\}$ in polynomial time in the size of $\mathcal{M}'$, i.e., in polynomial time in the size of $\mathcal{M}$ but in exponential time in the number of reachability properties r, and*

if so we can build a randomised exponential-memory strategy satisfying the problem. Moreover, we can build an ϵ-approximated Pareto curve $\mathcal{P}(\epsilon)$ in time polynomial in $\mathcal{M}$ and in $\frac{1}{\epsilon}$, but exponential time in the number of reachability properties r.

Example 3.5 (*Applying Algorithm 6 on a simple MDP*). Let $\mathcal{M}$ be the MDP of Figure 3.11. We are interested to know if there exists a strategy σ such that

$$\underbrace{\mathbb{P}^{\sigma}_{s_0}(\Diamond\{s_1\}) \geq 0.8}_{\text{reachability property 1}} \wedge \underbrace{\mathbb{P}^{\sigma}_{s_0}(\Diamond\{s_2\}) = 1}_{\text{reachability property 2}} .$$

As s_1 is not absorbing, we cannot use the technique described in Subsection 3.3.3 to

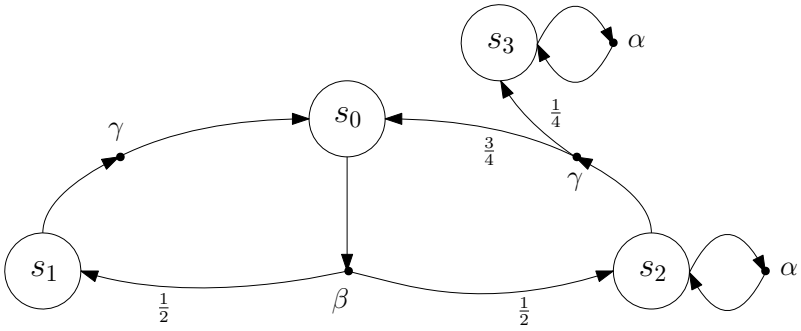


Figure 3.11. Simple MDP $\mathcal{M}$ with state space $S = \{s_0, s_1, s_2, s_3\}$ and action space $A = \{\alpha, \beta, \gamma\}$

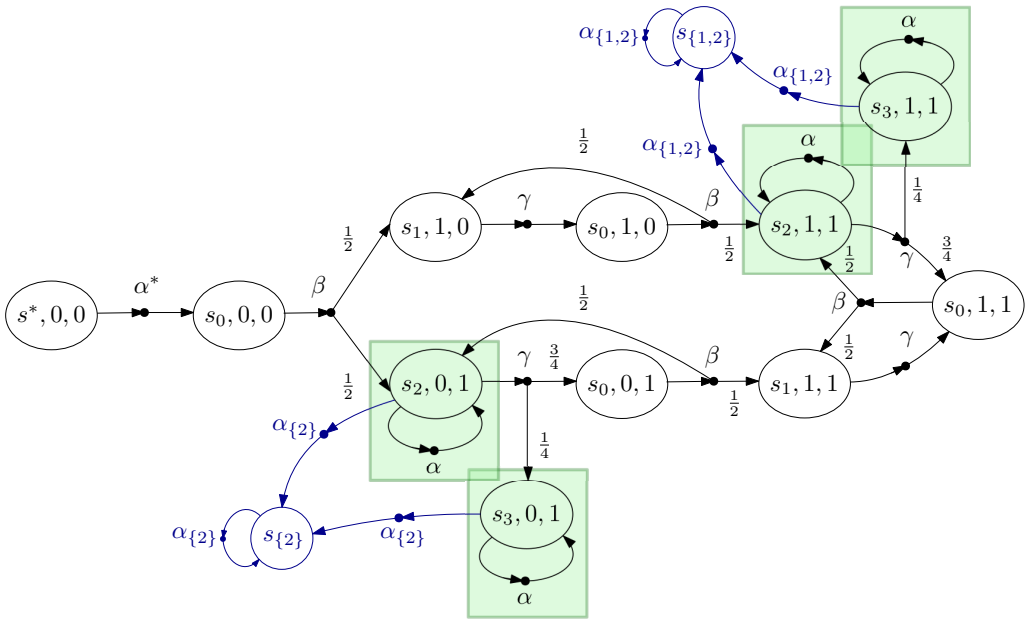


Figure 3.12. MDP $\mathcal{M}'$ resulting of the modification of $\mathcal{M}$ in Algorithm 6

solve this MO-SR problem. We apply Algorithm 6 and build the associated MDP $\mathcal{M}'$ to solve the problem (cf. Figure 3.12). The MECs of $\mathcal{M}'$ are the sub-MDPs highlighted in green, and the state spaces of these MECs form the sets T_R, with $R \in 2^{\{1,2\}}$, following the reachability properties $i \in R$ that each MEC favours. Here,

the MEC containing the unique state $(s_2, 0, 1)$ and the one containing $(s_3, 0, 1)$ favour the second reachability property and thus are in $T_{\{2\}}$. Then, the MEC containing the state $(s_2, 1, 1)$ and the one containing $(s_3, 1, 1)$ favour the properties 1 and 2 and thus are is $T_{\{1\}}$, $T_{\{2\}}$, and $T_{\{1,2\}}$. We then add the new absorbing target states s_R, for each subset of properties $R \subseteq \{1, \dots, r\}$, and we add a new transition from all states of each MEC $\mathcal{C}$ to s_R such that R is the maximal subset of properties favoured by $\mathcal{C}$ (in Figure 3.12, we ignore the target states s_R unconnected to MECs of $\mathcal{M}'$). We finally define the new set of absorbing target states $F_i = \{s_R \mid i \in R\}$ for $i \in \{1, \dots, r\}$, i.e., $F_1 = \{s_{\{1,2\}}\}$ and $F_2 = \{s_{\{2\}}, s_{\{1,2\}}\}$. It remains to decide if there exists a strategy σ' satisfying

$$\mathbb{P}^{\sigma'}_{(s^*,0,0)}(\Diamond F_1) \geq 0.8 \ \wedge \ \mathbb{P}^{\sigma'}_{(s^*,0,0)}(\Diamond F_2) = 1.$$

3.4 Percentile queries in multi-dimensional Markov decision processes

In this section, we define some strategies allowing to deal with multiple *percentile queries* in *multi-dimensional Markov decision processes* [RRS17]. A multi-dimensional MDP is an MDP where the weight of each action has multiple dimensions. Remind the SSP-P problem in Section 1.2.2 from Chapter 1, consisting in deciding the existence of a strategy satisfying one single percentile query. We are now interested in defining a strategy solving *simultaneously* multiple SSP-P problems on different weight dimensions and for different set of target states.

Definition 3.14 (Multi-dimensional Markov decision process). A d-dimensional MDP is a tuple $\mathcal{M} = (S, A, \Delta, w, AP, L)$ where

- S, A, Δ, AP, L are defined as for classical MDPs, and
- w is a d-dimension weight function $w : A \to \mathbb{N}_0^d$. For $k \in \{1, \dots, d\}$, we denote $w_k : A \to \mathbb{N}_0$ the projection of w on the k^{th} dimension, i.e., the function mapping each action α to the k^{th} element of the vector $w(\alpha)$.

Example 3.6 (*Multi-dimensional MDP for the wireless sensor network*). We update the MDP presented in Example 3.2 in order to fill in the details left open in the situation described in Example 3.1. We modify the weight function in order to support the energy cost of each action. This modification yields the MDP of Figure 3.13. We assume here that waiting an ACK from a node uses 100 *mili Joules* (mJ), sending a message to n_1 uses 196 mJ, sending a message to n_2 uses 294 mJ, and staying in the *sleep* state during 10 ms uses 20 mJ.

Remark 3.3 (*Truncated sum in a multi-dimensional MDP*). The truncated sum function TS must be slightly modified in order to handle multidimensional MDPs. Indeed, let $\mathcal{M}$ be a d-dimensional MDP with state space S and weight function w. We denote by TS_k^T the truncated sum up to T on the k^{th} dimension, with $k \in \{1, \dots, d\}$. More formally, let $\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \cdots \in Paths(s)$ be a path of $\mathcal{M}$ starting

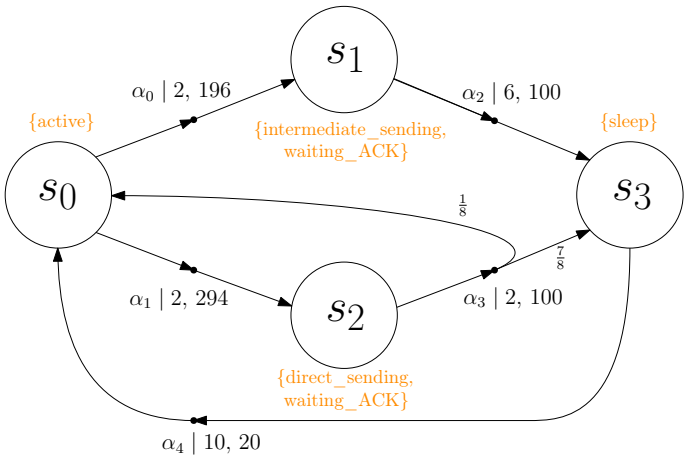


Figure 3.13. Multi-dimensional MDP modelling the situation of Example 3.1

from $s \in S$,

$$\mathsf{TS}_k^T(\pi) = \begin{cases} \sum_{i=1}^{n} w_k(\alpha_i) & \text{if } \forall i \in \{0, \dots, n-1\},\ s_i \notin T \wedge s_n \in T, \\ +\infty & \text{else.} \end{cases}$$

Furthermore, let σ be a strategy for $\mathcal{M}$, we can define the event $\Diamond_{k:\leq \ell} T$ on σ-paths starting from $s \in S$ as follows:

$$\Diamond_{k:\leq \ell} T = \{\pi \in Paths^{\sigma}(s) \mid \mathsf{TS}_k^T(\pi) \leq \ell\}.$$

Definition 3.15 (SSP-PQ **problem**). Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be a d-dimensional MDP, $s \in S$, and $q \in \mathbb{N}$ percentile constraints. These percentile constraints are described by a set of target states $T_i \subseteq S$, the dimensions $k_i \in \{1, \dots, d\}$, the cost thresholds $\ell_i \in \mathbb{N}$, and the probability thresholds $\alpha_i \in [0, 1] \cap \mathbb{Q}$, for each $i \in \{1, \dots, q\}$. The *stochastic shortest path percentile queries* problem (SSP-PQ, for short) consists in deciding if there exists a strategy σ such that

$$\bigwedge_{i=1}^{r} \mathbb{P}_s^{\sigma}(\Diamond_{k_i:\leq \ell_i} T_i) \geq \alpha_i.$$

Example 3.7 (SSP-PQ *problem in the wireless sensor network*). Let $\mathcal{M}$ be the MDP of Figure 3.13, modelling the situation described in Example 3.1 for the node n_0. We are interested to satisfy multiple percentile queries in $\mathcal{M}$, from the state s_0 (i.e., the *active* state of the sensor n_0). Firstly, we want to ensure with a probability of 80% that the message will be sent to the sensor n_2 and acknowledged within 4 ms (cf. query $\mathcal{Q}_1$). Secondly, we want to ensure with a probability of 90% that the message will be sent to the sensor n_2 and acknowledged with an energy cost inferior than or equal to 700 mJ (cf. query $\mathcal{Q}_2$).

- $\mathcal{Q}_1 := ?\exists\sigma \ \ \mathbb{P}_{s_0}^{\sigma}(\Diamond_{1:\leq 4} \{s_3\}) \geq 0.8$
- $\mathcal{Q}_2 := ?\exists\sigma \ \ \mathbb{P}_{s_0}^{\sigma}(\Diamond_{2:\leq 700} \{s_3\}) \geq 0.9$

Let σ_1 and σ_2 be two pure memoryless strategies for $\mathcal{M}$ such that

- σ_1 corresponds to the strategy always trying a direct sending to n_2, and
- σ_2 corresponds to the strategy sending the message via n_1.

The strategy σ_1 satisfies $\mathcal{Q}_1$ but does not satisfies $\mathcal{Q}_2$. Indeed, this strategy yields the events

$$\Diamond_{1:\leq 4}\{s_3\} = \Diamond_{2:\leq 700}\{s_3\} = \{\pi \in Paths^{\sigma_1}(s_0) \mid s_0 s_2 s_3 \in Pref(\pi)\},$$

because $\mathsf{TS}_1^{\{s_3\}}(s_0 \xrightarrow{\alpha_1} s_2 \xrightarrow{\alpha_3} s_3 \dots) = 4$, $\mathsf{TS}_2^{\{s_3\}}(s_0 \xrightarrow{\alpha_1} s_2 \xrightarrow{\alpha_3} s_3 \dots) = 394$, and paths of the form $s_0 \xrightarrow{\alpha_1} s_2 \xrightarrow{\alpha_3} s_0 \dots$ necessarily exceed the cost thresholds 4 and 700. Then, $\mathbb{P}^{\sigma_1}_{s_0}(\Diamond_{1:\leq 4}\{s_3\}) = 0.875 \geq 0.8$ and $\mathbb{P}^{\sigma_1}_{s_0}(\Diamond_{2:\leq 700}\{s_3\}) = 0.875 < 0.9$. In the same way, σ_2 satisfies $\mathcal{Q}_2$ but does not satisfy $\mathcal{Q}_1$. Indeed, σ_2 yields the events

$$\Diamond_{1:\leq 4}\{s_3\} = \emptyset, \text{ and } \Diamond_{2:\leq 700}\{s_3\} = \{\pi \in Paths^{\sigma_2}(s_0) \mid s_0 s_1 s_3 \in Pref(\pi)\},$$

because all σ_2-paths starting from s_0 have the form $s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_2} s_3 \dots$, $\mathsf{TS}_1^{\{s_3\}}(s_0 \xrightarrow{\alpha_0} s_2 \xrightarrow{\alpha_2} s_3 \dots) = 8 > 4$, and $\mathsf{TS}_2^{\{s_3\}}(s_0 \xrightarrow{\alpha_0} s_2 \xrightarrow{\alpha_2} s_3 \dots) = 296 \leq 700$. Then, $\mathbb{P}^{\sigma_2}_{s_1}(\Diamond_{1:\leq 4}\{s_3\}) = 0 < 0.8$ and $\mathbb{P}^{\sigma_2}_{s_0}(\Diamond_{2:\leq 700}\{s_3\}) = 1 \geq 0.9$.

In order to satisfy both queries at the same time, we will consider a finite-memory strategy $\sigma_{1\wedge 2}$ with two modes, representing a *compromise* between these two strategies (cf. Figure 3.14). Indeed, we assume that $\sigma_{1\wedge 2}$ is initialised in the mode m_1 and

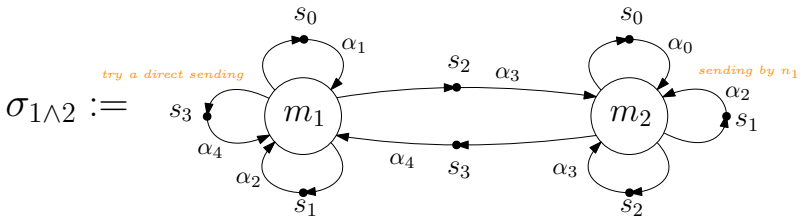


Figure 3.14. Strategy representing a compromise between σ_1 and σ_2 for $\mathcal{M}$

first try a direct sending to n_2. If it fails, it then sends the message via n_1. As $\sigma_{1\wedge 2}$ tries a direct sending to n_2, the cost of this try is 4 ms, 394 mJ and succeeds with a probability $\frac{7}{8}$. If it fails, the strategy next sends the message via n_1 and the time threshold is exceeded. It does however not matter because the probability threshold to reach s_3 within 4 ms is satisfied. Then, assuming the direct sending has failed, the message is sent via n_1. In that case, the total cost of the sending is thus exactly $394 + 296 = 690 < 700$ mJ. Thus,

- $\mathbb{P}^{\sigma_{1\wedge 2}}_{s_0}(\Diamond_{1:\leq 4}\{s_3\}) = 0.875 \geq 0.8 \implies \sigma_{1\wedge 2}$ satisfies $\mathcal{Q}_1$, and
- $\mathbb{P}^{\sigma_{1\wedge 2}}_{s_0}(\Diamond_{2:\leq 700}\{s_3\}) = 1 \geq 0.9 \implies \sigma_{1\wedge 2}$ satisfies $\mathcal{Q}_2$.

Assume now that we want to additionally ensure that the message will be sent and acknowledged within 8 ms with a probability of 0.9. That yields the following SSP-PQ problem:

$$?\exists\sigma \;\; \mathbb{P}^{\sigma}_{s_0}(\Diamond_{1:\leq 4}\{s_3\}) \geq 0.8 \;\wedge\; \mathbb{P}^{\sigma}_{s_0}(\Diamond_{1:\leq 8}\{s_3\}) \geq 0.9 \;\wedge\; \mathbb{P}^{\sigma}_{s_0}(\Diamond_{2:\leq 700}\{s_3\}) \geq 0.9$$

None pure strategy can satisfy simultaneously these percentile queries, even finite-memory strategies. In order to satisfy this SSP-PQ problem, we additionally need the notion of randomisation in our strategies.

Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be a d-dimensional MDP, $s^* \in S$, and $q \in \mathbb{N}$ percentile constraints described by $T_i \subseteq S$, the dimension $k_i \in \{1, \dots, d\}$, the cost threshold $\ell_i \in \mathbb{N}$, and the probability threshold $\alpha_i \in [0, 1] \cap \mathbb{Q}$, for each constraints $i \in \{1, \dots, q\}$. We will now introduce an algorithm to solve the SSP-PQ problem for the state s^* and these q constraints. This algorithm allows to build an optimal strategy with both finite-memory and randomisation to satisfy the problem. Note that we assume all dimensions of $\mathcal{M}$ being relevant for this problem, i.e., $\bigcup_{i \in \{1,\dots q\}} k_i = \{1, \dots, d\}$. On the other hand, we simply drop the n irrelevant dimensions of $\mathcal{M}$ for the problem and we only consider the $d - n$ remaining one. In order to describe this algorithm, we need to define the *multi-dimensional unfolding of an MDP*.

Definition 3.16 (**Multi-dimensional unfolding of an MDP**)**.** The *d-dimensional unfolding of the MDP $\mathcal{M}$ from the state s^* up to the highest cost threshold of $(\ell_i)_{i \in \{1,\dots,q\}}$ on each dimension of $(k_i)_{i \in \{1,\dots,q\}}$* is a 1-dimensional MDP $\mathcal{M}' = (S', A', \Delta', AP, L')$ defined as follows:

- $S' \subseteq S \cup \times(\{0, \dots, \ell^{\max}\} \cup \{\bot\})^d$, with $\ell^{\max} = \max_{i \in \{1,\dots,q\}} \ell_i$, is the set of states composed of states $(s, v_1, \dots, v_d)$ such that $s \in S$ and, for each dimension $k \in \{1, \dots, d\}$, $v_k \in \{0, \dots, \ell_k^{\max}\} \cup \{\bot\}$, where $\ell_k^{\max}$ is the maximum threshold on the dimension k, i.e.,

$$\ell_k^{\max} = \max_{i \in \{1,\dots,q\} \mid k_i = k} \ell_i,$$

  and we consider that $\bot > \ell_k^{\max}$ and $\bot + v = \bot$, for all $v \in \mathbb{N}$. Intuitively, the vector $(v_1, \dots, v_d)$ records the cost of paths on all the dimensions while unfolding $\mathcal{M}$. Moreover, we add a sink state $s_\bot$ in S', corresponding to the (absorbing) state for which the maximum cost thresholds have been exceeded on all the dimensions.

- A' is the set of action of the unfolding such that $\alpha \in A'$ for all actions $\alpha \in A$, and $A'(s, v_1, \dots, v_d) = A(s)$ for all $(s, v_1, \dots, v_d) \in S'$. Moreover, there is a special action $\alpha_\bot$, making the sink state $s_\bot$ absorbing, such that $A'(s_\bot) = \{\alpha_\bot\}$.

- Δ' is the probability transition function defined as follows: let $x = (s, v_1, \dots, v_d) \in S'$, $x' = (s', v'_1, \dots, v'_d) \in S'$, $\alpha \in A'(s)$, and $succ_k : S' \times A' \to \{0, \dots, \ell_k^{\max}\}$,

$$((s, v_1, \dots, v_d),\, \alpha) \mapsto \begin{cases} v_k + w_k(\alpha) & \text{if } v_k + w_k(\alpha) \leq \ell_k^{\max} \\ \bot & \text{else,} \end{cases}$$

  referring to the α-successor of $(s, v_1, \dots, v_d)$ on the cost dimension $k \in \{1, \dots, d\}$,

$$\Delta'(x, \alpha, x') = \begin{cases} \Delta(s, \alpha, s') & \text{if } \forall k \in \{1, \dots, d\},\ v'_k = succ_k(x) \\ & \quad \text{and } \exists k \in \{1, \dots, d\},\ v_k \neq \bot, \\ 1 & \text{if } \forall k \in \{1, \dots, d\},\ succ_k(x) = \bot \\ & \quad \text{and } x' = s_\bot, \\ 0 & \text{otherwise.} \end{cases}$$

- $L' : S' \to AP$, $(s, v_1, \ldots, v_d) \mapsto L(s)$ is the labelling function of the unfolding.

Furthermore, as we unfold $\mathcal{M}$ from s^*, we limit the state space S' with the states reachable from $(s^*, 0, \ldots, 0)$.

This definition allows us to describe an algorithm defined to solve the SSP-PQ problem for the state s^* and the q constraints (cf. Algorithm 7).

Algorithm 7 Solving the SSP-PQ problem

1. We build the MDP $\mathcal{M}'$, being the d-dimensional unfolding of $\mathcal{M}$ up to the highest cost thresholds on each of these dimensions. A path is thus discarded only if its current cost exceeds all highest threshold on each of these dimensions. Indeed, some paths exceeding the cost threshold of a request could still interesting to satisfy the others.

2. Then, we compute the set of target states

$$T'_i = \{(s, v_1, \ldots, v_d) \in S' \mid s \in T_i \ \wedge \ v_{k_i} \leq \ell_i\}$$

   in $\mathcal{M}'$ for each request $i \in \{1, \ldots, q\}$.

3. Finally, we are left with a MO-SR problem on $\mathcal{M}'$: we are looking for a strategy ensuring that each of these sets T'_i are reached from $(s^*, 0, \ldots, 0)$ with a probability α_i (cf. Section 3.3).

Remark 3.4 (SSP-PQ *problem with single-target queries*). Assume that all constraints are described by the same set of target states, i.e., $\forall i \in \{1, \ldots, r\}$, $\cup_{j=1}^{r} T_j = T_i$. In that case, as soon as a path visit a target state in the unfolding, it is not useful to consider and visit successors of this target state until that the highest cost thresholds have been exceeded on all the dimensions. Indeed, remind that given any prefix of a path, extending it can never decrease its current cost (the weights are strictly positive).

Thus, under the assumption that the constraints of an SSP-PQ problem for a given MDP have the same set of target states, we can make absorbing all of these states in the unfolding to avoid visiting some useless states. Solving this SSP-PQ problem is then polynomial in the size of the multi-dimensional unfolding of the MDP (cf. Corollary 3.1).

Theorem 3.7 (*Solving the* SSP-PQ *problem*). *The* SSP-PQ *problem can be solved in general in polynomial time in the size of $\mathcal{M}$ and exponential time in the query size (exponential in the number of constraints and pseudo-polynomial in the length of the largest threshold). Furthermore, it can be decided in pseudo-polynomial time in the size of $\mathcal{M}$ and in the length of the highest threshold for simultaneously single-dimension and single-target queries (cf. Remark 3.4, and Corollary 3.1). Randomised exponential-memory strategies are always sufficient and in general necessary.*

Example 3.8 (SSP-PQ *problem in the wireless sensor network*). We apply Algorithm 7 to answer to the percentile request left open in Example 3.7:

$$?\exists\sigma \quad \underbrace{\mathbb{P}^{\sigma}_{s_0}(\Diamond_{1:\leq 4}\{s_3\}) \geq 0.8}_{\mathcal{Q}_1} \wedge \underbrace{\mathbb{P}^{\sigma}_{s_0}(\Diamond_{1:\leq 8}\{s_3\}) \geq 0.9}_{\mathcal{Q}_2} \wedge \underbrace{\mathbb{P}^{\sigma}_{s_0}(\Diamond_{2:\leq 700}\{s_3\}) \geq 0.9}_{\mathcal{Q}_3}$$

In order to solve this SSP-PQ problem, we build the 2-dimensional unfolding $\mathcal{M}'$ of the MDP of Figure 3.13, for the state s_0, the cost thresholds $\ell_1 = 4$, $\ell_2 = 8$, $\ell_3 = 700$, and the dimensions $k_1 = k_2 = 1$, $k_3 = 2$ (cf. Figure 3.15). As each constraint

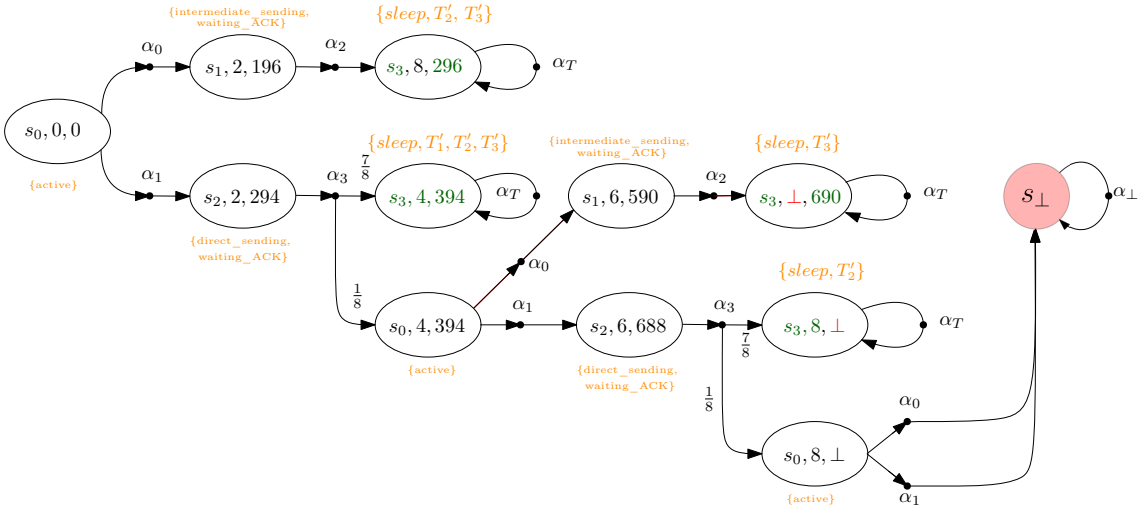


Figure 3.15. Unfolding of the MDP of Figure 3.13 from s_0 up to the cost thresholds 4 and 8 on the time dimension, and 700 on the energy dimension

describes a percentile problem, and as the set of target states for all constraints is $\{s_3\}$, we do not need to consider states after that $\{s_3\}$ is reached in all paths starting from $(s_0, 0, 0)$ in $\mathcal{M}'$. We can thus make all states referring to s_3 absorbing in $\mathcal{M}'$ (cf. Remark 3.4). The new sets of target states are $T'_1 = \{(s_3, 4, 394)\}$, $T'_2 = \{(s_3, 8, 296), (s_3, 4, 394), (s_3, 8, \bot)\}$, $T'_3 = \{(s_3, 8, 296), (s_3, 4, 494)\}$ for all query $\mathcal{Q}_i$, with $i \in \{1, 2, 3\}$. As all target states are absorbing, we can build a satisfying randomised memoryless optimal strategy in polynomial time in the size of $\mathcal{M}'$ (cf. Corollary 3.1). We build the randomised memoryless strategy

$$\sigma(s)(\alpha) = \begin{cases} 1 & \text{if } \alpha \in A'(s) \text{ and } |A'(s)| = 1, \text{ or if } s = (s_0, 0, 0) \text{ and } \alpha = \alpha_1, \\ \frac{1}{2} & \text{if } s = (s_0, 4, 394) \text{ and } \alpha \in \{\alpha_0, \alpha_1\}, \\ 0 & \text{otherwise,} \end{cases}$$

for $s \in S'$ and S', A' being respectively state and action space of $\mathcal{M}'$. The MC induced by σ is given on Figure 3.16. We have $\mathbb{P}^{\sigma}_{(s_0,0,0)}(\Diamond T'_1) = \frac{7}{8}$, $\mathbb{P}^{\sigma}_{(s_0,0,0)}(\Diamond T'_2) = \frac{7}{8} + \frac{1}{8} \cdot \frac{1}{2} \cdot \frac{7}{8} \geq 0.9$, and $\mathbb{P}^{\sigma}_{(s_0,0,0)}(\Diamond T'_3) = \frac{7}{8} + \frac{1}{8} \cdot \frac{1}{2} \geq 0.9$. Thus, we have that σ satisfies $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{Q}_3$. The strategy σ is memoryless in $\mathcal{M}'$, and exponential-memory in $\mathcal{M}$. Thus, the SSP-PQ problem is solved.

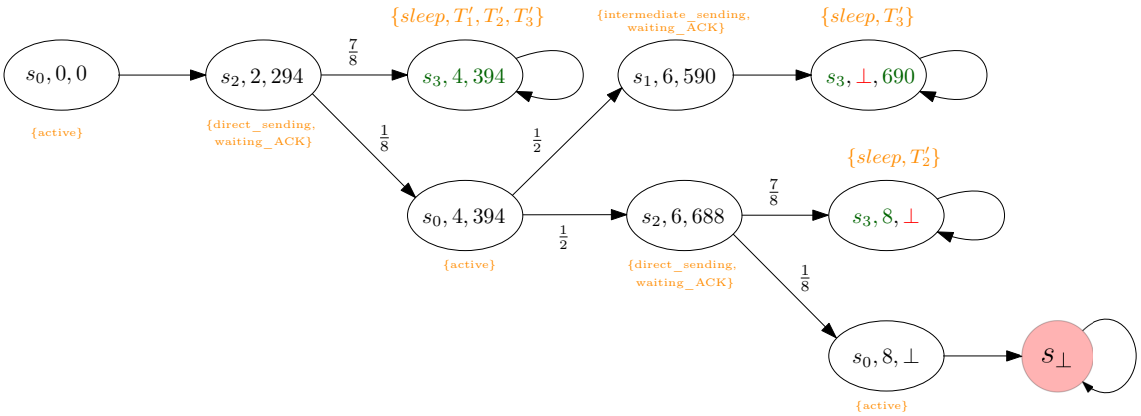


Figure 3.16. Product of σ with $\mathcal{M}'$, yielding the induced MC by σ

Chapter 4

Overview of the tool Storm

Storm[1] [DJKV17] is a probabilistic model checker implemented in `C++` recently released. It is able to analyse four Markov models, featuring discrete-time Markov chains and Markov decision processes. The main aim of this tool is to be competitive in terms of performance, to be updated with new verification algorithms, to be able to deal with a large panel of modelling languages,... This tool provides a lot of model checking features, including solvers for all the problems presented in the previous chapters, including multi-objective problems. In this chapter, we present how MDPs can be defined in this powerful tool, how they are represented and with which different approaches problems are solved.

4.1 Models

As mentioned above, Storm allows to model-check four types of Markov models. The first family of models is discrete-time Markov models, that are covered in the first chapter, composed by discrete-time Markov chains and Markov decision processes. The second family of models is continuous-time Markov models, composed by continuous-time Markov chains and Markov automata.

Continuous-time Markov chains (CTMC, for short) [Gil16] are similar to discrete-time Markov chains, but each state s is characterised by an exit rate λ_s. The time t spent in each state s of the system is negatively exponentially distributed with this rate, i.e., the probability to stay in s after t time units passed in s is $e^{-\lambda_s t}$. This exit rate and a probability transition function allow to induce a generator function with which it is possible to compute the probability that the system is currently in a state s' while the system was in the state s, t time units ago. Note that the "step" notion is replaced by the "time" notion in comparison with discrete-time models. Furthermore, a Markov automaton (MA, for short) is a continuous-time nondeterministic Markov model where the key idea is the same as between MCs and MDPs in discrete-time.

We will not dwell on continuous-time models anymore and focus primarily on discrete-time models.

[1] We consider Storm in version 1.2.0

4.2 Input formats

Storm supports various native input formats, including Prism, Jani, generalised stochastic Petri nets, dynamic fault trees, cpGCL and explicit format. We will introduce the three input languages supported by the main executable of Storm by presenting how to use them to model our discrete-time systems.

4.2.1 Prism

The Prism language [KNP09] allows to deal with MCs, CTMCs and MDPs in Storm. It is a state-based language using reactive *modules*. We will not define the complete syntax and semantic of the Prism language, but rather provide a way to define our MDPs. Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be a finite MDP, with $|S| = n$, and $s_i \in S$ be the i^{th} state of S. In Prism, this state s_i corresponds to the initial state of the system. That is, the probability measure that we consider is defined on paths starting from s_i.

In the Prism language, a *module* represents a component of the system. By definition of our MDPs, we just need a single module to represent them. A module allows to characterise S, A and Δ. We begin by enumerating states of $\mathcal{M}$:

$$x : [0 \,..\, n-1] \text{ init } i;$$

In the Prism language, x is considered as a *variable* of the module, ranging over states of $\mathcal{M}$. In this manner, we express that for all $j \in \{0, \ldots, n-1\}$, $s_j \in S$ is the j^{th} state of S represented by the index j of the variable x (i.e., $x = j$), and s_i is the initial state from which events start, represented by the index i of the variable x (i.e., $x = i$). Note that a variable represents a set of states of the MDP. Thus, it is possible that a module has more than one variable if the state space of the MDP is formed by a Cartesian product of set of states. Thanks to variables of the module, it is possible to form predicates Φ on the variables of the module with the syntax described with the following grammar:

$$\begin{aligned} \Phi &::= \mathit{true} \mid x\varphi p \mid (X) \mid (\mathcal{E}) \\ X &::= \Phi \mid \Phi \& X \\ \mathcal{E} &::= \Phi \mid \Phi | \mathcal{E} \end{aligned}$$

where x is a variable of the module, $\varphi \in \{<, \leq, >, \geq, =, \neq\}$ and $p \in \mathbb{N}$. Let $s \in S$ be a state of $\mathcal{M}$. We have that $s \models \Phi$ iff Φ is true for the state s, i.e.,

- $s \models \mathit{true}$,
- $s \models x\varphi p$ iff there exists an index $j \in \mathbb{N}$ in the range of indices of the module variable x by which the state s is represented (i.e., s is represented by $x = j$) and such that $j\varphi p$.
- $s \models (\Phi_1 \& \ldots \& \Phi_m)$ iff for all $k \in \{1, \ldots, m\}$, $s \models \Phi_k$, and
- $s \models (\Phi_1 | \ldots | \Phi_m)$ iff there exists $k \in \{1, \ldots, m\}$ such that $s \models \Phi_k$.

We denote by $Sat(\Phi)$ the set of states satisfying the predicate Φ, i.e., $Sat(\Phi) = \{s \in S \mid s \models \Phi\}$.

Example 4.1 (*Satisfaction of a predicate on variables of an MDP module*). Consider an MDP $\mathcal{M}$ with state space S and a Prism program containing a module M defining this MDP. Assume that the size of the state space S is $|S| = 5$ and that the module M contains a unique variable x, ranging over all states of S. Let $\Phi = (x \leq 1 \,||\, x > 3)$ be a predicate on the variables of the module M, then $Sat(\Phi) = \{s_0, s_1, s_4\}$.

The behaviours of the module, i.e., transitions of the set $\rightarrow$[2] are then described inside the module by a set of *guarded commands*, taking the following form:

$$[\alpha]\ \Phi \rightarrow \delta_1 : \phi_1 + \cdots + \delta_m : \phi_m;$$

where Φ is a predicate over the variables of the module such that, for all $s \in Sat(\Phi)$, $\alpha \in A(s)$, $i \in \{1, \ldots, m\}$, $\delta_i \in [0, 1]$ with $\sum_{i=1}^{m} \delta_i = 1$ and where ϕ_i is an *update formula* describing a state represented by a variable of the module. This guarded command basically means that all states that satisfy the predicate Φ go to the state described by ϕ_i with a probability δ_i, when the action α is chosen. An update formula ϕ is of the form

$$\phi ::= (x'_1 = u_1)\&(x'_2 = u_2)\& \ldots \&(x'_k = u_k)$$

where $x_1, x_2, \ldots, x_k$ are variables of the module and $u_1, u_2, \ldots, u_k$ are *expressions* (i.e., literal values, variables, constants and operators) over all variables.

Example 4.2 (*Guarded command for a simple module*). Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP. Consider that all states of $\mathcal{M}$ are represented by the values of the single variable x, ranging on all states of the set S, with $|S| = n$, the transitions of $\mathcal{M}$ can be described with the following guarded command: let s_j be the j^{th} state of S, for all enabled actions $\alpha \in A(s_j)$ of s_j,

$$[\alpha]\ x = j \rightarrow \delta_0 : x' = j_0 + \cdots + \delta_{m-1} : x' = j_{m-1};$$

where $s_{j_0}, \ldots, s_{j_{m-1}}$ are α-successors of s_j and $\delta_k = \Delta(s_j, \alpha, s_{j_k})$, with $m = |Succ(s_j, \alpha)|$, $k \in \{0, \ldots, m-1\}$, $j_k \in \{0, \ldots, n-1\}$ and where s_{j_k} is the k^{th} α-successor of s_j and the j_k^{th} state of S.

The set of atomic propositions AP and the labelling function L of $\mathcal{M}$ are characterised in the Prism language as follows: for all $a \in AP$,

$$\text{label } \text{“}a\text{”} = \Phi;$$

where the set of states satisfying the predicate Φ is actually the set of states labelled with a, i.e., $Sat(\Phi) = \{s \in S \mid a \in L(s)\}$.

Finally, the weight function w can be characterised with the notion of *rewards*. In the Prism language, it is possible to associate to each transition of the set $\rightarrow$ a real value. A *dimension rewards* is a set of rewards taking the following form:

$$[\alpha]\ \Phi : x;$$

[2]Remind that the transition relation $\rightarrow$ is the set $\{(s, \alpha, s') \in S \times A \times S \mid \Delta(s, \alpha, s') > 0\}$

where $\alpha \in A$ is an action, Φ is a predicate and $x \in \mathbb{R}$ is the value of the reward of going from a state $s \in S$, such that $\alpha \in A(s)$, to states of $Sat(\Phi)$. Note that if the action α is omitted (giving a reward of the form $[]\ \Phi : x^*;$), each transition $s \xrightarrow{\alpha} s'$ such that $s' \in Sat(\Phi)$ and $\alpha \in A(s)$ has a reward of x^*. Note also that it is possible to specify multiple dimensions rewards, allowing to define multi-dimensional MDPs. We can adapt the reward notion to describe the weight function of our MDPs with a set of rewards formed as follows: let $\alpha \in A$ be an action of $\mathcal{M}$ and $x = w(\alpha)$, the reward formed by

$$[\alpha]\ true : x;$$

is actually the weight of the action α.

Example 4.3 (*Defining an MDP in the Prism language*). Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be the MDP of Figure 4.1. This MDP can be defined in the Prism language as follows:

```
mdp

module simple_mdp

s: [0..2] init 0;

[beta] s=0 -> 0.5: (s'=1) + 0.5:
    (s'=2);
[gamma] (s=1 | s=2) -> 1: (s'=0);
[alpha] s=2 -> 1: (s'=2);

endmodule

label "a" = s=0;
label "b" = s=1;

rewards "weights"
  [alpha] true: 5;
  [beta] true: 3;
  [gamma] true: 2;
endrewards
```

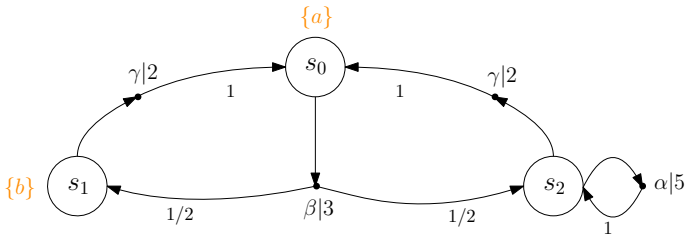


Figure 4.1. MDP $\mathcal{M}$, with $S = \{s_0, s_1, s_2\}$, $A = \{\alpha, \beta, \gamma\}$ and $AP = \{a, b\}$

Example 4.4 (*Unfolding of an MDP in the Prism language*). Let $\mathcal{M}$ be the MDP of Figure 4.1 with state space S and $\ell \in \mathbb{N}$ be a length threshold such that $\ell = 8$. We can define the unfolding of $\mathcal{M}$ from the state s_0 until the length threshold ℓ for the set of target states $T = \{s_1\}$ (cf. Figure 1.5) in the Prism language as follows:

```
mdp

const int l = 8;

module unfolded_simple_mdp

s: [0..2] init 0;
v: [0..l+1] init 0;

[beta] s=0 -> 0.5: (s'=1) & (v'=min(v+3, l+1)) + 0.5: (s'=2) & (v'=min(v+3, l+1));
[gamma] (s=1 | s=2) -> 1.0: (s'=0) & (v'=min(v+2, l+1));
[alpha] s=2 -> 1.0: (s'=2) & (v'=min(v+5, l+1));

endmodule
```

```
label "a" = s=0;
label "b" = s=1;
label "target" = (s=1 & v<=1);

rewards "weights"
  [alpha] true: 5;
  [beta] true: 3;
  [gamma] true: 2;
endrewards
```

Since the state space of the unfolded MDP $\mathcal{M}_\ell$ is a Cartesian product between the state space S of $\mathcal{M}$ and the set $V = \{0, \ldots, \ell, \bot\} \subseteq \mathbb{N} \cup \{\bot\}$, we can define the variables of the unfolded MDP $\mathcal{M}_\ell$ with the variables of $\mathcal{M}$ and a new variable v, ranging from 0 to $\ell + 1$ (where the $(\ell+1)^{\text{th}}$ state actually represents the $\bot$ value). The guarded commands of the module are obviously defined, following the definition of the probability transition function Δ_ℓ of any unfolded MDP. Finally, a label *target* is added, labelling each state of the set $T_\ell = \{(t, v) \in S \times V \mid t \in T \ \wedge \ v \leq \ell\}$.

4.2.2 Explicit format

Storm also supports input models specified in explicit format. The syntactic power of such a format is lower compared to the Prism language, but is very simple and intuitive. Moreover, this format has the advantage of being supported by several model checkers.

An MDP specified in explicit format consists of a transition matrix file, describing explicitly transitions of the system, a reward matrix file, describing explicitly rewards of these transitions, and a labelling set file. More formally, let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP. We assume the state space S is enumerated as $S = \{0, \ldots, n-1\}$ and the set of enabled action $A(s)$ is enumerated as $A(s) = \{0, \ldots, m-1\}$, for all $s \in S$. The transition matrix P and the reward matrix R of sizes $(| \rightarrow |, 4)$ are defined as follows. Let (s, α, s') be the l^{th} transition of $\rightarrow$, we have $P_{l,1} = R_{l,1} = s$, $P_{l,2} = R_{l,2} = \alpha$, $P_{l,3} = R_{l,3} = s'$, and $P_{l,4} = \Delta(s, \alpha, s')$ and $R_{l,4} = w(\alpha)$. Additionally, labels of this MDP are described with a set of vectors Lab defined as follows :

$$Lab = \{(s, a_1, \ldots, a_k) \in \mathbb{N} \times AP^k \mid k = |L(s)| \wedge \{a_1, \ldots, a_k\} = L(s)\}.$$

Note that if $L(s) = \emptyset$ for a given state of the system, this state can be omitted in Lab.

Example 4.5 (*Define an MDP in explicit format*). Let $\mathcal{M}$ be the MDP of Figure 4.1. The explicit transition matrix P, the explicit reward matrix R and the explicit set of labels Lab of $\mathcal{M}$ are defined as follows:

$$P = \begin{pmatrix} 0 & 0 & 1 & \frac{1}{2} \\ 0 & 0 & 2 & \frac{1}{2} \\ 1 & 0 & 0 & 1 \\ 2 & 0 & 2 & 1 \\ 2 & 1 & 0 & 1 \end{pmatrix}, \qquad R = \begin{pmatrix} 0 & 0 & 1 & 3 \\ 0 & 0 & 2 & 3 \\ 1 & 0 & 0 & 2 \\ 2 & 0 & 2 & 5 \\ 2 & 1 & 0 & 2 \end{pmatrix}, \qquad Lab = \{(0, a), (1, b)\}$$

For instance, the first transition is explained as follows: the state s_0 go to s_1 with a probability $\frac{1}{2}$, choosing its first enabled action, and the cost of this transition is 3. Moreover, the state s_0 is labelled with a. Note that the enumeration of actions do not corresponds to the enumeration of the set A, but rather the enumeration of the set of enabled action $A(s)$ of each state s.

4.2.3 Jani

Jani [BDH+17] is a recent modeling language whose syntax is based on Json and whose design is based on networks of communicating automata. The purpose of this language is to provide a stable and uniform interface between tools and, more particularly, model checkers. This format supports MCs, CTMCs, MDPs and MAs.

The semantic model of the Prism language forms the conceptual basis of Jani and is extended to also support MAs. Furthermore, Jani is designed to be easy to generate and parse programmatically without library dependencies. However, Jani has not been designed to be created manually by users, but rather to be automatically generated from higher-level and domain-specific languages, while remaining human-readable, in contrast to binary encodings. Storm actually allows to convert multiple input types to Jani. An example of conversion in available in Appendix C.1.

4.3 Engines

Storm is decomposed in five engines that pursue different approaches to solve a problem.

- *Sparse*: this engine is the default one. It uses sparse matrices as in-memory representation of models. Numerical analysis methods allow efficient memory allocation and fast operations for this data structure on small and moderately sized models. Actually, this data structure is very powerful where most states of the model have transitions to a small subset of the set of states of the model. This situation actually represents most of the cases: in general, underlying graphs of Markov models are rarely complete and the transition matrix representing the probability transition function of most of the models is thus sparse.
- *DD*: this engine uses multi-terminal binary decision diagrams (MTBDDs) [FMY97] as their primal representation. This data structure allows to handle gigantic models. Intuitively, each state of the system are encoded on n bits and the probability transition function is a boolean function over the bits representation of two states. Binary decision diagrams (BDDs) [BK08] are used to represent boolean functions and are actually compressed binary decision trees through directed acyclic graphs (DAGs), with two terminal states (1 and 0). MTBDDs are BDDs allowing numerical values for terminal states.
- *Hybrid*: this engine uses MTBDDs as representation of models, but sometimes uses sparse matrices if operations are judged more appropriate with this format.

- *Abstraction refinement*: this engine *abstracts* (non-necessary finite) discrete-time Markov models to finite *stochastic games* and automatically *refines* the *abstraction* as necessary. This engine is suited for models with an infinite state space (other engines cannot handle this type of models).

  Intuitively, the abstraction consists of constructing a smaller model by removing details from the concrete system. The motivation is that when a property hold in the abstract model, then it also holds in the concrete system. If the property does not hold in the abstraction, then either information from the model checking process allows to exhibit a counter-example to show that the property is false in the concrete system, or the abstraction is refined.

  Constructing an abstraction of an MDP implies a greater degree of nondeterminism. The process of abstraction must maintain a distinction between the nondeterminism from the original MDP and from the nondeterminism induced during the abstraction process, and this is why stochastic games are used [KKNP10].

- *Exploration*: this engine uses sparse representation for models and explores the state space of models with machine learning methods.

4.4 Solvers

Several solvers are available in Storm. Storm uses the suited solver according to the situation (i.e., the input model, its format, the engine used and its in-memory representation).

- *Linear equation systems*: Storm provides a solver for linear equation systems, allowing to handle reachability and expectation properties in MCs.

- *Bellman equations*: these equations form systems defined in the next chapter for MDPs (cf. Theorem 2.5 and Appendix B.1) whose solutions describe reachability and expectation properties. The associated solver is based on an approximation technique named *value iteration* (cf. Subsection 2.1.2). Intuitively, this technique approximates values of the solution of the equation by iteratively updating the approximation. This approximation is guaranteed to converge to the solution of the equation system. Therefore, the difficulty of this technique is to find a threshold used to stop the iteration process.

  Since numerical methods are prone to numerical problems, Storm supports enabling *exact arithmetic* to obtain precise results.

- *Linear programming*: a solver for (mixed-integer) linear programs is implemented in Storm, allowing to describe reachability and expectation properties for MDPs.

  If the size of the considered MDP is sufficiently moderate, Storm uses this solver as the primary.

- *SMT*: an SMT problem is a decision problem for first-order logical formulae with equality and without quantifier. SMT solvers are used to minimise the size of models by *bisimulation*. A bisimulation is an equivalence relation between two systems having the same behaviour.
- *Stochastic games*: although Storm does not support stochastic games as input format, solvers for them are available for the abstraction refinement engine which uses these as representation.
- *Multi-objective*: this solver allows to compute ϵ-approximations of Pareto curves (cf. Subsection 3.3.1 and Section 4.6), for a given $\epsilon > 0$.

4.5 Model checking

Storm properties are expressed in a language including PRCTL and CSL (i.e., *continuous stochastic logic*, for continuous Markov models). The syntax of the subset of this language supporting PRCTL is slightly different than the classical PRCTL, and we present in this section the slight changes in comparison to the classical PRCTL syntax presented in Chapter 2. We denote by *Storm-PRCTL* the subset of the Storm language supporting PRCTL.

Notation 4.1 (*MC and MDP support by the syntax of Storm-PRCTL*). Let $\mathcal{M}$ be an MDP. In this section, we cover at a time the syntax of Storm-PRCTL for MCs and MDPs. As a reminder, we can consider an MC as an MDP such that there is only one enabled action for each state. The differences between PRCTL for MCs and for MDPs are probabilistic operator and expected operator. Indeed, in the context of MCs, these operators are $\mathcal{P}_J(\phi)$ and $\mathcal{E}_C(\Phi)$, referring to the probability of an event formed by a path formula ϕ and the expected cost-to-target of the satisfaction set of a state formula Φ. In the context of MDPs, these operators are $\mathcal{P}_J^{\max}(\phi)$ and $\mathcal{E}_C^{\min}(\Phi)$, referring to the maximum probability of an event formed by a path formula ϕ and the minimal expected cost-to-target of the satisfaction set of a state formula Φ. So, let the words

$$\lambda = \begin{cases} \varnothing & \text{if } \mathcal{M} \text{ is an MC} \\ \max & \text{else} \end{cases} \quad \text{and} \quad \mu = \begin{cases} \varnothing & \text{if } \mathcal{M} \text{ is an MC} \\ \min & \text{else} \end{cases}$$

where $\varnothing$ is the empty word. We can now generalise the syntax of these operators with $\mathcal{P}_J^{\lambda}(.)$ and $\mathcal{E}_C^{\mu}(.)$ according to $\mathcal{M}$ in PRCTL and we use this notation for the syntax of Storm-PRCTL.

Definition 4.1 (**Syntax of Storm-PRCTL**). Let $\mathcal{M}$ be an MDP with space of atomic proposition AP,

- Storm-PRCTL *state formulae* over AP are formed according to the following grammar:

$$\Phi ::= \mathsf{true} \mid \text{“}a\text{”} \mid \Phi_1 \,\&\, \Phi_2 \mid !\Phi \mid \mathsf{P}\lambda\,\varphi\,j\,[\phi] \mid \mathsf{R}\mu\,\varphi\,c\,[\mathsf{F}\,\Phi]$$

  where $a \in AP$ is an atomic proposition, ϕ is a path formula, $\varphi \in \{=, <, >, \leq, \geq\}$, $j \in [0, 1]$, and $c \in \mathbb{N} \cup \{+\infty\}$.

- Storm-PRCTL *path formulae* over AP are formed according to the following grammar:

$$\phi ::= \mathsf{X}\Phi \mid \Phi_1 \,\mathsf{U}\, \Phi_2 \mid \Phi_1 \,\mathsf{U} \leq n\, \Phi_2 \mid \Phi_1 \,\mathsf{U}\{\text{“d”}\} \leq \ell\, \Phi_2$$

  where Φ, Φ_1 and Φ_2 are state formulae, d is a *dimension reward* (cf. Subsection 4.2.1), and $n, \ell \in \mathbb{N}$.

Definition 4.2 (Equivalence between Storm-PRCTL formulae and PRCTL formulae). Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP, Φ, Ψ be PRCTL state formulae over AP, and Φ, Ψ be Storm-PRCTL state formulae over AP, there is an equivalence relation $\equiv$ over Storm-PRCTL and classical PRCTL state formulae:

$$\begin{aligned} \mathsf{true} &\equiv \mathit{true}, & \text{“}a\text{”} &\equiv a, \\ \Phi \,\&\, \Psi &\equiv \Phi \wedge \Psi, & !\,\Phi &\equiv \neg\Phi, \\ \mathsf{P}\lambda\, \varphi\, j\, [\phi] &\equiv \mathcal{P}^{\lambda}_{\varphi j}(\phi), & \mathsf{R}\mu\, \varphi\, c\, [\mathsf{F}\Phi] &\equiv \mathcal{E}^{\mu}_{\varphi c}(\Phi), \end{aligned}$$

$$\text{if and only if } \Phi \equiv \Phi \text{ and } \Psi \equiv \Psi,$$

where $a \in AP$, $j \in [0, 1]$, $c \in \mathbb{N} \cup \{+\infty\}$, and ϕ is a PRCTL path formula over AP. Additionally, $\Phi \,|\, \Psi \equiv \Phi \vee \Psi \equiv \neg(\neg\Phi \wedge \neg\Psi)$ if and only if $\Phi \equiv \Phi$ and $\Psi \equiv \Psi$. Then, there also exists an equivalence between Storm-PRCTL path formulae and classical PRCTL path formulae:

$$\begin{aligned} \mathsf{X}\Phi &\equiv \bigcirc\Phi, & \Phi \,\mathsf{U}\, \Psi &\equiv \Phi \,\mathsf{U}\, \Psi, \\ \Phi \,\mathsf{U} \leq n\, \Psi, &\equiv \Phi \,\mathsf{U}^{\leq n}\, \Psi, & \Phi \,\mathsf{U}\{\text{“d”}\} \leq \ell\, \Psi &\equiv \Phi \,\mathsf{U}_{\leq \ell}\, \Psi, \end{aligned}$$

$$\text{if and only if } \Phi \equiv \Phi \text{ and } \Psi \equiv \Psi,$$

where $n, \ell \in \mathbb{N}$ and d is a dimension reward. Additionally,

$$\mathsf{F}\Phi \equiv \Diamond\Phi, \text{ and } \qquad \mathsf{G}\Phi \equiv \Box\Phi,$$

$$\text{if and only if } \Phi \equiv \Phi.$$

The semantic of Storm-PRCTL is the same as the semantic of PRCTL following this equivalence between formulae. Optimised model checking algorithms are implemented in Storm and the main engine supports all properties formed with this syntax.

Example 4.6 (SSP-E *problem in Storm*). Let $\mathcal{M}$ be the MDP of Figure 4.1. We are interested to solve the following SSP-E problem:

$$?\exists\sigma \quad \mathbb{E}^{\sigma}_{s_0}(\mathsf{TS}^{\{s_1\}}) \leq 10,$$

and we can actually solve this problem by verifying that the following PRCTL property holds in s_0:

$$\mathcal{E}^{\min}_{\leq 10}(b).$$

This property is equivalent to the following in Storm-PRCTL:

$$\mathsf{Rmin} \leq 10\, [\mathsf{F}\text{“}b\text{”}].$$

```
>> storm --prism simple_mdp.prism --prop "Rmin<=10 [F \"b\"]"

Storm 1.2.0

-------------------------------------------------------------
Model type:  MDP (sparse)
States:  3
Transitions:  5
Choices:  4
Reward Models:  weights
State Labels:  3 labels
   * deadlock -> 0 item(s)
   * init -> 1 item(s)
   * b -> 1 item(s)
Choice Labels:  none
-------------------------------------------------------------

Model checking property R[exp]min<=10 [F "b"] ...
Result (for initial states): true

Time for model checking: 0.008s.
```

Figure 4.2. Verifying that $\mathcal{E}^{\min}_{\leq 10}(b)$ holds in the state s_0 of the MDP in Figure 4.1 in Storm

We consider the initial states of the Prism program as being the states that we want to verify that properties hold in it. As we have specified in the Prism program that the initial state is s_0, then Storm will model check the input property for this state (cf. Figure 4.2).
First, we see that Storm builds the MDP following the input Prism model with the sparse representation. It informs us that the system has five transitions and four nondeterministic choices. Thus, we see that the default dimension reward considered by Storm for this system is the one named *weights* in the program, that is actually the only one in this system. It also informs us that the system has no deadlock, one initial state (here s_0) and one state labelled with b (here s_1). Finally, Storm gives us the result of the model checking by answering *true* to the satisfiability of the property by the state s_0.

Example 4.7 (SSP-P *problem in Storm*). Let $\mathcal{M}$ be the MDP of Figure 4.1. We are interested to solve the following SSP-P problem:

$$?\exists\sigma \quad \mathbb{P}^{\sigma}_{s_0}(\Diamond_{\leq 8}\{s_1\}) \geq \frac{7}{10},$$

and we can actually solve this problem by verifying that the following PRCTL property holds in s_0:

$$\mathcal{P}^{\max}_{\geq \frac{7}{10}}(\Diamond_{\leq 8}\, b).$$

This property is equivalent to the following one in Storm-PRCTL:

$$\mathsf{Pmax} \geq 0.7\,[\mathsf{F}\{\text{“weights”}\} \leq 8\,\text{“}b\text{”}].$$

As in the previous example, running Storm with the Prism program modelling $\mathcal{M}$ as input with this Storm-PRCTL property allows to verify that this property holds in s_0 (cf. Figure 4.3).

```
storm --prism simple_mdp.prism --prop "Pmax>=0.7 [F{\"weights\"}<= 8 \"b\"]"
Storm 1.2.0

-------------------------------------------------------------
Model type:  MDP (sparse)
States:  3
Transitions:  5
Choices:  4
Reward Models:  weights
State Labels:  3 labels
   * init -> 1 item(s)
   * deadlock -> 0 item(s)
   * b -> 1 item(s)
Choice Labels:  none
-------------------------------------------------------------

Model checking property Pmax>=7/10 [true Urew{"weights"}<=8 "b"] ...
Result (for initial states): true

Time for model checking: 0.009s.
```

Figure 4.3. Verifying that $\mathcal{P}^{\max}_{\geq \frac{7}{10}}$ holds in the state s_0 of the MDP in Figure 4.1 in Storm

Storm confirms that this property holds in s_0.

Additionally, Storm allows to formulate probabilistic and expectation queries by replacing the probabilistic and expectation operators by $\mathsf{P}\lambda =?[.]$ and $\mathsf{R}\mu =?[.]$.

Definition 4.3 (Syntax of Storm-PRCTL queries). Let $\mathcal{M}$ be an MDP with state space S and atomic proposition space AP, Storm-PRCTL queries over AP are formed according to the following grammar:

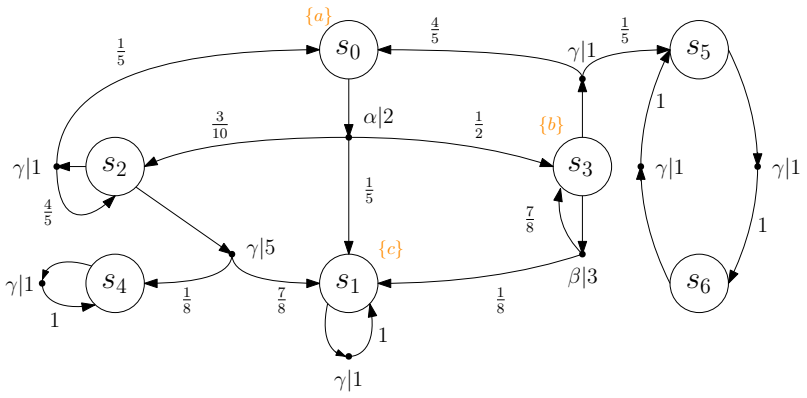
$$\Phi^{?} ::= \mathsf{P}\lambda =? [\phi] \mid \mathsf{R}\mu =? [\mathsf{F}\,\Phi]$$

where Φ is a Storm-PRCTL state formula and ϕ is a Storm-PRCTL path formula.

Definition 4.4 (Semantic of Storm-PRCTL queries). Let $\mathcal{M}$ be an MDP with state space S and atomic proposition space AP, $s \in S$, Φ be a Storm-PRCTL state formula over AP and ϕ be a Storm-PRCTL path formula over AP, the semantic of Storm-PRCTL queries over AP is defined as follows:

- $\mathsf{P}\lambda =? [\phi] = \mathbb{P}^{\lambda}_{s}(\{\pi \in Paths(s) \mid \pi \models \phi\})$.
- $\mathsf{R}\mu =? [\mathsf{F}\,\Phi] = \mathbb{E}^{\mu}_{s}(\mathsf{TS}^{Sat(\Phi)})$.

Example 4.8 (*Storm-PRCTL query*). Let $\mathcal{M}$ be the MDP of Figure 4.4 with atomic proposition space AP. We have shown in Example 2.7 that $s_0 \models \mathcal{P}^{\max}_{\geq \frac{1}{5}}((a \vee b)\,\mathsf{U}_{\leq 8}\, c)$,




```
mdp

module mdp_example

s: [0..6] init 0;

[alpha] s=0 -> 1/5: (s'=1) + 3/10: (s'=2) + 1/2: (s'=3);
[gamma] s=1 -> 1: (s'=1);
[gamma] s=2 -> 1/5: (s'=0) + 4/5: (s'=2);
[beta]  s=3 -> 1/8: (s'=1) + 7/8: (s'=3);
[gamma] s=3 -> 4/5: (s'=0) + 1/5: (s'=5);
[gamma] s=4 -> 1: (s'=4);
[gamma] s=5 -> 1: (s'=6);
[gamma] s=6 -> 1: (s'=5);

endmodule

label "a" = s=0;
label "b" = s=3;
label "c" = s=1;

rewards "weights"
  [alpha] true: 2;
  [beta] true: 3;
  [gamma] true: 1;
endrewards
```

Figure 4.4. MDP $\mathcal{M}$ from Example 2.7, with state space $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_5\}$ and with space of atomic propositions $AP = \{a, b, c\}$

and we are interested by the maximum probability with which this property holds in s_0. Since s_0 is the initial state of the Prism program, we can formulate the following Storm-PRCTL query over AP for the state s_0:

$$\mathsf{Pmax} =? \left(\left(\text{“}a\text{”} \mid \text{“}b\text{”}\right) \mathsf{U}\{\text{“weights”}\} <= 8 \text{ “}c\text{”}\right).$$

We run Storm by passing the Prism program modelling $\mathcal{M}$ as input with this property (cf. Figure 4.5) Storm informs us that the result of this query is 0.3325,

```
Model checking property Pmax=? [("a" | "b") Urew{"weights"}<=8 "c"] ...
Result (for initial states): 0.3325
Time for model checking: 0.000s.
```

Figure 4.5. Storm-PCRTL query for the MDP in Figure 4.4

that corresponds to the solution of the LP defined to verify that the property hold in s_0 in Example 2.7.

4.6 Multi-objective model checking

Storm supports multi-objective model checking for MDPs, and allows us to formulate properties extending the Storm-PRCTL language in order to solve multi-objective problems encountered in Chapter 3.

Definition 4.5 (**Syntax of multi-objective Storm-PRCTL**)**.** Let $\mathcal{M}$ be an MDP with state space S and atomic proposition space AP, and $n \in \mathbb{N}$ Storm-PRCTL state formulae over AP Φ_i, *multi-objective Storm-PRCTL formulae* over AP are formed according to the following grammar:

$$\mathsf{multi}(\Phi_1, \ldots, \Phi_n).$$

Definition 4.6 (**Syntax of multi-objective Storm-PRCTL queries**)**.** Let $\mathcal{M}$ be an MDP with state space S and atomic proposition space AP, $k \in \mathbb{N}_0$ Storm-PRCTL state queries over AP $\Phi_i^?$ for $i \in \{1, \ldots, k\}$, and l Storm-PRCTL state formulae over AP Ψ_j for $j \in \{1, \ldots, l\}$, *multi-objective Storm-PRCTL queries* over AP are formed according to the following grammar:

$$\mathsf{multi}(\Phi_1^?, \ldots, \Phi_k^?, \Psi_1, \ldots, \Psi_l).$$

Definition 4.7 (**Achievable points of a multi-objective Storm-PRCTL query**)**.** Let $\mathcal{M}$ be an MDP with state space S and set of atomic proposition AP, $s \in S$, and $\mathcal{Q}$ be a multi-objective Storm-PRCTL query over AP for the state s. This query is composed by $k \in \mathbb{N}_0$ queries of the form $\mathsf{Pmax} =?(\phi_i)$ and $l \in \mathbb{N}$ state formulae of the form $\mathsf{Pmax}(\phi_{k+j}) \geq \alpha_i$, where $\alpha_j \in [0,1] \cap \mathbb{Q}$, ϕ_i, $\phi_k + j$ are Storm-PRCTL path formulae over AP for $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, l\}$. The set of achievable points $U_{\mathcal{Q}}$ is given by

$$\begin{aligned} U_{\mathcal{Q}} = \{\, p \in [0,1]^{k+l} \mid \exists \sigma, \sigma', \forall i \in \{1, \ldots, k\} \, \forall j \in \{1, \ldots, l\}, \\ \mathbb{P}_s^{\sigma}(Paths(s, \phi_i)) = p_i \ \wedge \ p_j = \mathbb{P}_s^{\sigma'}(Paths(s, \phi_{k+j})) \geq \alpha_j \,\}, \end{aligned}$$

where $Paths(s, \phi) = \{\pi \in Paths(s) \mid \pi \models \phi\}$, for a given path formula ϕ.

In order to verify multi-objective reachability properties, the multi-objective solver build an approximation of the Pareto curve for the n properties and return *safe* points of the Pareto curve, i.e., points contained in the under-approximation of the Pareto curve, following an approximation value $\epsilon > 0$. The default approximation technique used to build this Pareto curve is an adaptation of the one presented in [FKP12].

Example 4.9 (SSP-WE *problem in Storm*). Getting back to Example 3.3, let $\mathcal{M}$ be the MDP of Figure 4.6. Consider the following SSP-WE problem (covered in Example 3.3):

$$?\exists \sigma \ \ \mathbb{P}_{s_0}^{\sigma}(\Diamond_{\leq 12} \{s_2\}) = 1 \ \wedge \ \mathbb{E}_{s_0}^{\sigma}(\mathsf{TS}^{\{s_2\}}) \leq 6.$$

```
mdp

module simple_wireless_sensor_network

s: [0..3] init 0;

[alpha0] s=0 -> 1 : (s'=1);
[alpha1] s=0 -> 1 : (s'=2);
[alpha2] s=1 -> 1 : (s'=3);
[alpha3] s=2 -> 0.125 : (s'=0) +
    0.875 : (s'=3);
[alpha4] s=3 -> 1 : (s'=0);

endmodule

label "active" = s=0;
label "sleep" = s=3;
label "waiting_ACK" = s=1 | s=2;
label "intermediate_sending" = s=1;
label "direct_sending" = s=2;

rewards "time"
  [alpha0] true : 2;
  [alpha1] true : 2;
  [alpha2] true : 6;
  [alpha3] true : 2;
  [alpha4] true : 10;
endrewards

rewards "energy"
  [alpha0] true : 196;
  [alpha1] true : 294;
  [alpha2] true : 100;
  [alpha3] true : 100;
  [alpha4] true : 20;
endrewards
```

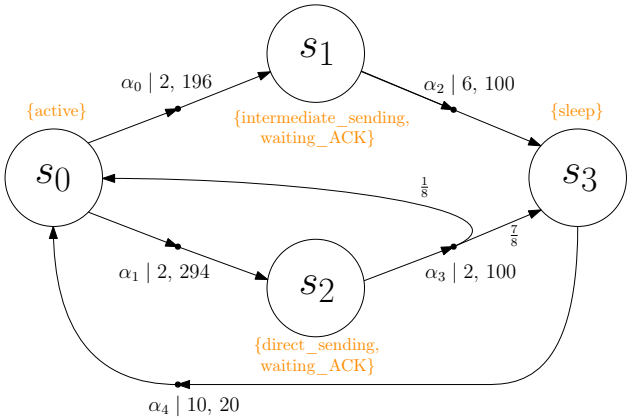


Figure 4.6. Multi-dimensional MDP modelling the situation of Example 3.1

```
Model checking property
  multi(Pmax>=1 [true Urew{"time"}<=12 "sleep"],
        R[exp]{"time"}min=? [F "sleep"]) ...
Result (for initial states): 5
Time for model checking: 0.049s.
```

Figure 4.7. Storm-PRCTL query for the SSP-WE problem of Example 3.3

Let us re-formulate this problem as a multi-objective Storm-PRCTL query:

$$\mathsf{multi}(\mathsf{Pmax} >= 1\,[\mathsf{F}\{\text{“time”}\} <= 12\,\text{“}\mathit{sleep}\text{”}],\ \mathsf{Rmin} =?\,[\mathsf{F}\text{“}\mathit{sleep}\text{”}]).$$

We run Storm by passing the Prism program of Figure 4.6 as input with this property (cf. Figure 4.7). Storm informs us that the result of this query is 5, what agrees with results of Example 3.3.

Example 4.10 (SSP-PQ *problem in Storm*). Getting back to Example 3.8, let $\mathcal{M}$ be the MDP of Figure 4.6 Consider the following SSP-PQ problem:

$$?\exists\sigma\ \ \mathbb{P}^{\sigma}_{s_0}(\Diamond_{1:\leq 4}\{s_3\}) \geq 0.8\ \wedge\ \mathbb{P}^{\sigma}_{s_0}(\Diamond_{1:\leq 8}\{s_3\}) \geq 0.9\ \wedge\ \mathbb{P}^{\sigma}_{s_0}(\Diamond_{2:\leq 700}\{s_3\}) \geq 0.9.$$

Let us re-formulate this problem as a multi-objective Storm-PRCTL query:

- $\mathcal{Q}_1 :=$ Pmax $=?[$F$\{$"time"$\}$ $<=$ 4 "*sleep*"$]$,
- $\mathcal{Q}_2 :=$ Pmax $=?[$F$\{$"time"$\}$ $<=$ 8 "*sleep*"$]$,
- $\mathcal{Q}_3 :=$ Pmax $=?[$F$\{$"energy"$\}$ $<=$ 700 "*sleep*"$]$,

$$\text{multi}(\mathcal{Q}_1,\ \mathcal{Q}_2,\ \mathcal{Q}_3).$$

We run Storm by passing the Prism program of Figure 4.6 as input with this multi-objective property (cf. Figure 4.8). The results displayed by Storm are interesting

```
Model checking property multi(
  Pmax=? [true Urew{"time"}<=4 "sleep"],
  Pmax=? [true Urew{"time"}<=8 "sleep"],
  Pmax=? [true Urew{"energy"}<=700 "sleep"]) ...
Result (for initial states):
Underapproximation of achievable values: Polytope with 5 Halfspaces:
   ( 0.0178571,          1,          0) * x <= 1
   (  0.142857,          1,      0.875) * x <= 1.875
   (         0,          0,          1) * x <= 1
   (         0,          1,          0) * x <= 1
   (         1,          0,          0) * x <= 0.875

Overapproximation of achievable values: Polytope with 6 Halfspaces:
   (         1,          0,          0) * x <= 0.875
   (         0,          1,          0) * x <= 1
   (         0,          0,          1) * x <= 1
   (     0.125,      0.875,          0) * x <= 0.970703
   ( 0.0707965,   0.495575,   0.433628) * x <= 0.929204
   ( 0.0175439,   0.982456,          0) * x <= 0.982456

3 pareto optimal points found (Note that these points are safe, i.e.,
contained in the underapproximation, but there is no guarantee for optimality):
   (     0.875,      0.875,          1 )
   (     0.875,   0.984375,      0.875 )
   (         0,          1,          1 )


Time for model checking: 0.079s.
```

Figure 4.8. Storm-PRCTL query for the SSP-PQ problem of Example 3.8

since it gives some informations about how the approximation of the Pareto-curve has been computed. Indeed, we see that an underapproximation of the curve is computed as well as an overapproximation, allowing the multi-objective solver to infer an ϵ-(under)-approximation of this curve . Note that the ϵ value can be tuned with the parameter `--multiobjective:precision`. Note also that Storm answers *true* to the SSP-PQ problem defined above, since $(0.8, 0.9, 0.9)$ is an achievable point of the query, and is situated in a close area between the ϵ-Pareto-optimal points $(0.875, 0.875, 1)$ and $(0.875, 0.984375, 0.875)$.

In order to interpret results of multi-objective Storm-PRCTL queries, it would be useful to plot the ϵ-approximated Pareto curve of these queries and visualise it. In fact, Storm also allows to compute the strategies offering the *Pareto-minimal*

probabilities, i.e., the worst achievable points to reach some set of target states in an MDP for a multi-objective context, with a slightly adapted syntax for the probability operator (Pmin =?). This allows us to determine the set of achievable points under a Pareto-curve.

Example 4.11 (*Achievable points for Example 3.15*). Getting back to Example 3.15, and considering the MDP $\mathcal{M}$ of Figure 4.6, we solved the SSP-PQ problem

$$?\exists\sigma\ \mathbb{P}^{\sigma}_{s_0}(\Diamond_{1:\leq 4}\{s_3\}) \geq 0.8 \wedge \mathbb{P}^{\sigma}_{s_0}(\Diamond_{2:\leq 700}\{s_3\}) \geq 0.9$$

by considering the pure finite-memory strategy σ^* trying once a direct sending and then sending the message via an intermediate node, if the previous sending has failed. This strategy offers an achievable point $(\frac{7}{8}, 1)$. We are here interested in plotting the Pareto curve and determining the set of achievable point $U_{\mathcal{Q}}$ of the following Storm-PRCTL query $\mathcal{Q}$:

multi(Pmax =?[F{"time"} <= 4"*sleep*"], Pmax =?[F{"energy"} <= 700"*sleep*"])

First, we run Storm by passing the Prism program of Figure 4.6 as well as the Storm-PRCTL query $\mathcal{Q}$. The result of this query is described in Figure 4.9. The

```
Model checking property multi(
  Pmax=? [true Urew{"time"}<=4 "sleep"],
  Pmax=? [true Urew{"energy"}<=700 "sleep"]) ...
Result (for initial states):
Underapproximation of achievable values: Polytope with 2 Halfspaces:
   (         1,          0) * x <= 0.875
   (         0,          1) * x <= 1

Overapproximation of achievable values: Polytope with 2 Halfspaces:
   (         1,          0) * x <= 0.875
   (         0,          1) * x <= 1

1 pareto optimal points found (Note that these points are safe, i.e.,
contained in the underapproximation, but there is no guarantee for optimality):
   (     0.875,          1 )
```

Figure 4.9. Storm-PRCTL query for the SSP-PQ problem of Example 3.15

under and over approximations of the Pareto curve are the same, that means that the optimal point found is precisely the unique point of the Pareto Curve, i.e., $\mathcal{P} = \{(\frac{7}{8}, 1)\}$. This optimal point exactly corresponds to the strategy σ^*. Then, in order to determine the worst strategies for this query, we formulate the following Storm-PRCTL query:

multi(Pmin =?[F{"time"} <= 4"*sleep*"], Pmin =?[F{"energy"} <= 700"*sleep*"]).

The result of this query is described in Figure 4.10. The minimal points found for this problem $(\frac{7}{8}, \frac{7}{8})$ and $(0, 1)$, correspond to the two simple pure memoryless strategies consisting in either always trying a direct sending or just always sending the message via the intermediate node. These two points allow us to define lower bounds for achievable points $U_{\mathcal{Q}}$ (cf. Figure 4.11). The blue area marks the set of achievable

```
Model checking property multi(
  Pmin=? [true Urew{"time"}<=4 "sleep"],
  Pmin=? [true Urew{"energy"}<=700 "sleep"]) ...
Result (for initial states):
Underapproximation of achievable values: Polytope with 3 Halfspaces:
   (        -1,          0) * x <= 0
   (         0,         -1) * x <= -0.875
   ( -0.142857,         -1) * x <= -1

Overapproximation of achievable values: Polytope with 3 Halfspaces:
   (        -1,          0) * x <= 0
   (         0,         -1) * x <= -0.875
   (    -0.125,     -0.875) * x <= -0.875

2 pareto optimal points found (Note that these points are safe, i.e.,
contained in the underapproximation, but there is no guarantee for optimality):
   (     0.875,      0.875 )
   (         0,          1 )
```

Figure 4.10. Pareto-minimal points for the query describing the problem of Example 3.15

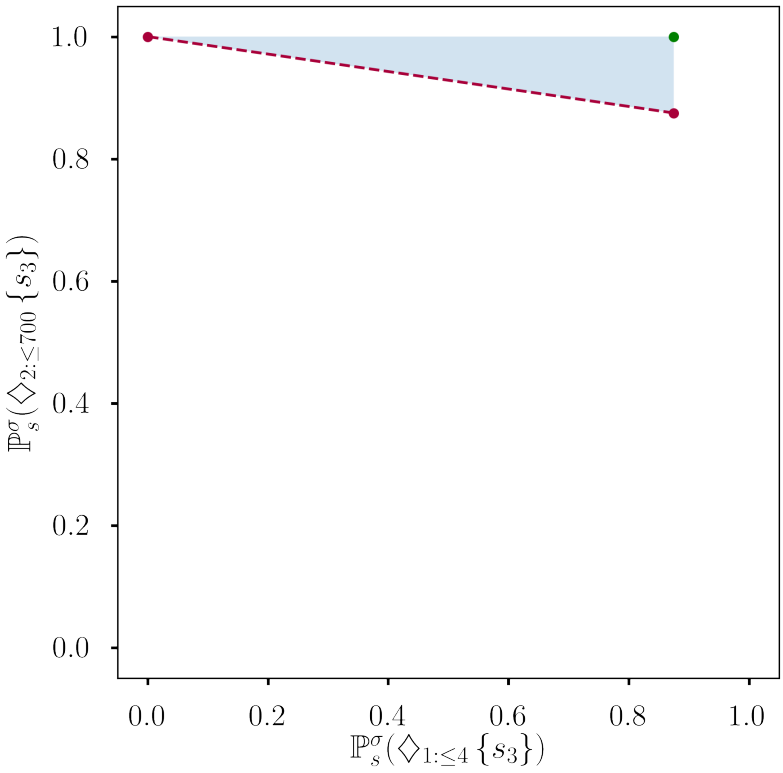


Figure 4.11. Pareto curve $\mathcal{P}$ (in green), worst achievable points (in red) and achievable points $U_{\mathcal{Q}}$ (in blue)

point $U_{\mathcal{Q}}$ and most of points of this set corresponds to a randomised strategy. Note that no strategy exist for points in the area under the red dotted line, corresponding to worst achievable points.

Conclusion

Summary

This thesis presented many decision problems related to the minimisation of costs to reach one or multiple sets of target states in MDPs. In order to solve these problems, we presented algorithms allowing to build optimal strategies satisfying them. A summary of the results for each problem is depicted in Table 4.1.

Problem	**Decision time**	**Satisfying strategy**		
		type	*memory*	*optimal*
SR	$\mathrm{P}(\mathcal{M})$	pure	memoryless	yes
SSP-E	$\mathrm{P}(\mathcal{M})$	pure	memoryless	yes
SSP-P	$\mathrm{P}(\mathcal{M}) \cdot \mathrm{P}_{ps}(\ell)$	pure	$\mathrm{P}_{ps}(\ell)$	yes
SP-G	$\mathrm{P}(\mathcal{M})$	pure	memoryless	yes
SSP-WE	$\mathrm{P}(\mathcal{M}) \cdot \mathrm{P}_{ps}(\ell)$	pure	$\mathrm{P}_{ps}(\ell)$	yes
MO-SR (absorbing target states)	$\mathrm{P}(\mathcal{M})$	randomised	memoryless	ϵ
MO-SR	$\mathrm{P}(\mathcal{M}) \cdot \mathrm{E}(\mathcal{Q})$	randomised	$\mathrm{E}(\mathcal{Q})$	ϵ
SSP-PQ (unique set of target states + single dimension)	$\mathrm{P}(\mathcal{M}) \cdot \mathrm{P}_{ps}(\ell_{\max})$	randomised	$\mathrm{P}_{ps}(\ell)$	ϵ
SSP-PQ	$\mathrm{P}(\mathcal{M}) \cdot \mathrm{E}(\mathcal{Q})$	randomised	$\mathrm{E}(\mathcal{Q})$	ϵ

Table 4.1. Results of problems approached in the thesis. $\mathrm{P}(x)$, $\mathrm{E}(x)$ and $\mathrm{P}_{ps}(x)$ respectively denote polynomial, exponential and pseudo-polynomial time in parameter x. The symbol $\mathcal{M}$ denotes the size of the model, ℓ denotes a cost threshold, and $\mathcal{Q}$ denotes the query size. The optimal strategy column describes if it is possible to get an optimal satisfying strategy with the same time complexity that the decision time for a given problem. Moreover, the value ϵ denotes that it is possible to have an ϵ-approximation of an optimal strategy.

We saw that verifying reachability, persistence, and repeated reachability properties in an MDP requires to solve an SR problem, which consists in deciding the existence of a strategy allowing to reach a set of target states with a probability threshold. Such a problem can be solved in polynomial time in the size of the model, by linear programming or with an iterative approximation approach called value iteration.

By considering cost of paths, we first assumed actions of MDPs having a

single-dimension weight, and we saw that the SSP-E problem, consisting in deciding the existence of a strategy ensuring good expected cost-to-target, can be decided in polynomial time in the size of the model, by linear programming or with the value iteration approach. Then, we saw that verifying a cost-bounded property in any MDP requires to solve an SSP-P problem, consisting in deciding the existence of a strategy allowing to reach a set of target states with a cost bounded and with a probability threshold. The resolution of this problem requires to build an unfolding of the model up to the cost threshold, leading a pseudo-polynomial time in the size of the model and in the length of this cost threshold.

Afterwards, we presented the SP-G problem, consisting in deciding the existence of a strategy allowing to guarantee a worst case threshold (in terms of cost) to reach a set of target states. Inspired by a turn-based two-player game approach, we saw that this problem can be solved by dynamic programming, without unfolding the model. This allowed us to present the multi-objective SSP-WE problem, consisting in deciding the existence of a strategy offering a worst case guarantee of reaching a set of target states while ensuring a good expectation to reach this set of target states. A solution to this problem was approached by unfolding the MDP up to the worst case threshold, and limiting its state space to the attractor of the set of target states for which cost of paths has not exceeded the worst case threshold.

Then, we presented the MO-SR problem, consisting in deciding the existence of a strategy satisfying multiple reachability to multiple set of target states, with multiple probability thresholds. Such a problem induces compromises between satisfying strategies, and we defined the notion of Pareto curve, allowing to deal with these compromises. Indeed, according to a given multi-objective problem, each point of this Pareto curve is actually linked to a (Pareto-)optimal strategy satisfying the problem. We saw that in the case where all target states are absorbing, the problem can be decided in polynomial time in the size of the model by linear programming. Furthermore, building Pareto-optimal strategies for the problem can be done by enumerating vertices of the Pareto curve, that can be ϵ-approximated in polynomial time in the model and in $\frac{1}{\epsilon}$. Based on these results, we approached the general case, where all the targets states are not necessarily absorbing, by exponentially increasing the size of the state space of the MDP, according to the number of reachability properties to satisfy, and by considering the maximal end components of this modified MDP. In that case, an ϵ-approximation of the Pareto curve can be built in polynomial time in the size of the model and $\frac{1}{\epsilon}$, but in exponential time in the number of reachability properties.

For the last problem, we have considered multi-dimensional MDPs, where actions have multiple weights. We approached the SSP-PQ problem, consisting in deciding the existence of a strategy allowing to satisfy multiple percentile queries. Each of these queries consists in reaching a set of target states with a cost bounded on a given dimension and with a probability threshold. We approached the problem by unfolding the MDP up to the highest cost threshold, on all its dimensions, necessarily leading to an exponential time decision, and by solving then an MO-SR problem for

the target states in this unfolding. The results for absorbing target states in the MO-SR problem allowed to improve this exponential time to a pseudo-polynomial time in the size of the model and in the length of the highest cost threshold for simultaneously single-dimension and single-target queries.

Finally, we introduced a modern model checker called Storm, allowing to model-check Markov decision processes, and supporting all the problems presented in this thesis. We presented some input languages allowing to encode MDPs in Storm, and the probabilistic branching time logic of Storm, allowing to express properties to verify and query a given MDP.

Future work

Other cost measures

In this thesis, we have considered cost of paths in MDPs with the truncated sum function (i.e., TS), that is perfectly suited for variations of the shortest path problem in MDPs. There actually are many other measures:

- the *discounted sum*, modelling that short-term costs are more important than long-term ones [RRS17],
- the *mean-payoff*, describing the long-run average cost per executed action in a path [BFRR13],
- etc.

It should be interesting to consider them and study multi-objective problems related to these measures.

Game-based abstraction

It is possible to reduce the size of MDPs by stochastic two-player game-based abstraction for a given reachability, or expected cost-to-target problem [KKNP10]. It could be useful to consider such abstraction techniques for multi-objective problems.

Reinforcement learning for strategies

Multiple machine learning techniques have been presented in [BCC+14] to improve performance by avoiding an exhaustive exploration of the state space, yielding precise lower and upper bounds to verify required properties. Again, it could be interesting to investigate the use of related techniques to verify multi-objective properties in Markov decision processes.

Reward-epoch model

In order to optimise the time complexity for the multi-objective problems requiring an unfolding, a way to avoid this unfolding have been introduced in [HJKQ18] by only looking at interesting states of the unfolding (intuitively, the model is implicitly

unfolded along reward epochs, and the regularities of a modification of the MDP, called epoch-model of the MDP, are exploited). It is actually the method used in Storm to verify multi-objective cost-bounded properties in a given MDP.

Storm

Moreover, as Storm is open-source, we can enrich this model checker with new algorithms. Indeed, for instance, the exploration and abstraction engines (cf. Section 4.3) do not support yet[3] the multi-objective problems.

[3] in the version of Storm 1.2.0

Bibliography

[BCC+14] Tomáš Brázdil, Krishnendu Chatterjee, Martin Chmelík, Vojtěch Forejt, Jan Křetínský, Marta Kwiatkowska, David Parker, and Mateusz Ujma. Verification of markov decision processes using learning algorithms. In Franck Cassez and Jean-François Raskin, editors, *Automated Technology for Verification and Analysis*, pages 98–114, Cham, 2014. Springer International Publishing.

[BDH+17] Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. Jani: Quantitative model and tool interaction. In *Proceedings, Part II, of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems - Volume 10206*, pages 151–168, New York, NY, USA, 2017. Springer-Verlag New York, Inc.

[BEGM09] Endre Boros, Khaled Elbassioni, Vladimir Gurvich, and Kazuhisa Makino. Generating vertices of polyhedra and related problems of monotone generation, 2009.

[BFa02] Wolfgang Thomas Thomas Wilke (eds.) Berndt Farwer (auth.), Erich Grädel. *Automata Logics, and Infinite Games: A Guide to Current Research*. Lecture Notes in Computer Science 2500. Springer-Verlag Berlin Heidelberg, 1 edition, 2002.

[BFRR13] Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *CoRR*, abs/1309.5439, 2013.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[Del17] Florent Delgrange. Stochastic shortest path in markov decision processes, 2017.

[DJKV17] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In Rupak Majumdar and Viktor Kunčak, editors, *Computer Aided Verification*, pages 592–600, Cham, 2017. Springer International Publishing.

[EKVY08] Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-objective model checking of markov decision processes. *Logical Methods in Computer Science*, 4(4), 2008.

[FKP12] Vojtech Forejt, Marta Z. Kwiatkowska, and David Parker. Pareto curves for probabilistic model checking. *CoRR*, abs/1206.6295, 2012.

[FMY97] M. Fujita, P. C. McGeer, and J. C.-Y. Yang. Multi-terminal binary decision diagrams: An efficient datastructure for matrix representation. *Form. Methods Syst. Des.*, 10(2-3):149–169, April 1997.

[Gil16] Nicolas Gillis. *Stochastic models for operational research*. UMONS, 2016.

[HJKQ18] Arnd Hartmanns, Sebastian Junges, Joost-Pieter Katoen, and Tim Quatmann. Multi-cost bounded reachability in mdp. In Dirk Beyer and Marieke Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 320–339, Cham, 2018. Springer International Publishing.

[KKNP10] Mark Kattenbelt, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. A game-based abstraction-refinement framework for markov decision processes. *Formal Methods in System Design*, 36(3):246–280, 2010.

[KNP09] M. Kwiatkowska, G. Norman, and D. Parker. Prism: Probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):40–45, 2009.

[PY00] Christos H. Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 86–92, 2000.

[Ran18] Mickael Randour. *Formal verification of computer systems*. ULB, 2018.

[RRS15] Mickael Randour, Jean-François Raskin, and Ocan Sankur. Variations on the stochastic shortest path problem. In *Verification, Model Checking, and Abstract Interpretation - 16th International Conference, VMCAI 2015, Mumbai, India, January 12-14, 2015. Proceedings*, pages 1–18, 2015.

[RRS17] Mickael Randour, Jean-François Raskin, and Ocan Sankur. Percentile queries in multi-dimensional markov decision processes. *Formal Methods in System Design*, 50(2-3):207–248, 2017.

Appendix

A.1 Systems of linear equations

The definitions of this section are inspired of [BK08].

A.1.1 Reachability in a Markov Chain

Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC, $s \in S$ be a state of $\mathcal{M}$ and $T \subseteq S$ be a set of target states in $\mathcal{M}$. We say that s *is not connected to* T if and only if, for all $\pi = s_0 s_1 \cdots \in Paths(s)$ and for all $k \in \mathbb{N}$, $s_k \notin T$. This definition allows us to compute the probability to reach T from s: let $(x_s)_{s \in S} \in [0, 1]$,

- if s is not connected to T, then we have $x_s = 0$,
- else, if $s \in T$, then we have $x_s = 1$,
- else, for all $s \in S \setminus T$ such that s is connected to T in the underlying graph of $\mathcal{M}$,

$$x_s = \underbrace{\sum_{s' \in S \setminus T} \Delta(s, s') \cdot x_{s'}}_{\text{reach } T \text{ via } s' \in S \setminus T} + \underbrace{\sum_{t \in T} \Delta(s, t)}_{\text{reach } T \text{ in one transition}}. \tag{1}$$

This defines a system of linear equations. Let $S_{=0}$ be the subset of states of S that are not connected to T in the underlying graph of $\mathcal{M}$, $S_{=1} = T$ and $S_{=?} = S \setminus (S_{=0} \cup S_{=1})$. The solution $(x_s)_{s \in S_{=?}}$ of the system of linear equations 1 is unique and $x_s = \mathbb{P}_s(\Diamond T)$ for all $s \in S$.

A.1.2 Expected cost-to-target paths in a Markov Chain

Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC, $s \in S$ be a state of $\mathcal{M}$ and $T \subseteq \mathrm{S}$ be a set of target states. $\mathbb{E}_s(\Diamond T)$ can be computed through a linear equations system defined as follows: let $succ(s) = \{s' \in S \mid \Delta(s, s') > 0\}$ be the set of successors of s,

$$x_s = \begin{cases} \infty & \text{if } \mathbb{P}_s(\Diamond T) < 1, \\ 0 & \text{if } s \in T, \text{ and} \\ \displaystyle\sum_{s' \in succ(s)} \Delta(s, s') \cdot (w(s, s') + x_{s'}) & \text{otherwise.} \end{cases} \tag{2}$$

Let $S_{=?} = \{s \in S \mid \mathbb{P}_s(\Diamond T) = 1\} \setminus T$. The solution $(x_s)_{s \in S_{=?}}$ of the system of linear equations 2 is unique and $x_s = \mathbb{E}_s(\Diamond T)$ for all $s \in S$.

A.2 Cost bounded reachability in a Markov chain

Let $\mathcal{M} = (S, \Delta, w, AP, L)$ be an MC. We denote by $\mathbb{P}_s^{\mathcal{M}}$ the probability measure $\mathbb{P}_s$ such that $s \in S$ on $\mathcal{M}$. Let $s \in S$ be a state of $\mathcal{M}$, $T \subseteq S$ be a set of target states and $\ell \in \mathbb{N}$ be a threshold. We can compute $\mathbb{P}_s^{\mathcal{M}}(\Diamond_{\leq \ell} T)$ by reduction to the reachability problem on $\mathcal{M}_\ell = (S_\ell, \Delta_\ell)$ to the set of target states $T_\ell \subseteq S_\ell$ that we build as follows:

- S_ℓ contains all states (s, v) such that $s \in S$ and $v \in \{0, \ldots, \ell\} \cup \{\bot\}$. We consider that $\bot > \ell$, with $\bot + v = \bot$ for all $v \in \mathbb{N}$. Intuitively, we record in v the cost of paths while unfolding $\mathcal{M}$. Target states are states of $T_\ell = \{(s, v) \in S_\ell \mid s \in T \wedge v \leq \ell\}$.
- $\Delta_\ell : S_\ell \times S_\ell \to [0, 1]$ is the probability transition function given by: $\forall (s, v), (s', v') \in S_\ell$,

$$\Delta_\ell((s, v), (s', v')) = \begin{cases} \Delta(s, s') & \text{if } v' = v + w(s, s') \text{ and } v' \leq \ell \text{ or} \\ & \text{if } v' = \bot \text{ and } v + w(s, s') > \ell, \\ 0 & \text{otherwise.} \end{cases}$$

Remark: here, the weight function is omitted in $\mathcal{M}_\ell$. So, $\mathcal{M}_\ell$ is unweighted. Resolving the cost bounded reachability by the threshold ℓ from s to T in $\mathcal{M}$ can be done by resolving the reachability problem from $(s, 0)$ to T_ℓ in $\mathcal{M}_\ell$, i.e., $\mathbb{P}_s^{\mathcal{M}}(\Diamond_{\leq \ell} T) = \mathbb{P}_{(s,0)}^{\mathcal{M}_\ell}(\Diamond T_\ell)$.

A.3 Linear programs

A.3.1 Stochastic reachability problem

Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP and $T \subseteq S$ be a set of target states of $\mathcal{M}$. Let $\alpha \in [0, 1]$ be a probability threshold. We will build an optimal strategy σ for the SR problem [BK08]. Indeed, we will compute through an LP $\max_\sigma \mathbb{P}_s^\sigma(\Diamond T) = \mathbb{P}_s^{\max}(\Diamond T)$ for all $s \in S$. If $\mathbb{P}_s^\sigma(\Diamond T) = \mathbb{P}_s^{\max}(\Diamond T) \geq \alpha$, σ is *optimal* for the SR problem from s to T. Otherwise, no strategy satisfying the SR problem exist. Let $S_{=1}$ be a subset of states such that $T \subseteq S_{=1} \subseteq \{s \in S \mid \mathbb{P}_s^{\max}(\Diamond T) = 1\}$ and $(x_s)_{s \in S} \subseteq [0, 1]^{|S|}$ be a vector for the following LP:

$$\min \sum_{s \in S} x_s$$

subject to the constraints :

$$\begin{aligned}
&x_s = 1 && \forall s \in S_{=1}, \\
&x_s = 0 && \forall s \notin S_{=1} \text{ such that } s \text{ is not connected to } T, \\
&x_s \geq \sum_{s' \in S} \Delta(s, \alpha, s') \cdot x_{s'} && \forall s \notin S_{=1}, \forall \alpha \in A(s) \text{ such that } s \text{ is connected to } T, \\
&0 \leq x_s \leq 1 && \forall s \in S
\end{aligned}$$

The optimal solution $(v_s)_{s \in S}$ of this LP is unique and gives the following result:

$$v_s = \mathbb{P}_s^{\max}(\Diamond T) \quad \forall s \in S$$

From this result, we can build an optimal memoryless strategy σ such that $\mathbb{P}_s^{\sigma}(\Diamond T) = \mathbb{P}_s^{\max}(\Diamond T)$. To do that, for each state s, we build $A^{\max}(s)$, the set of actions $\alpha \in A(s)$ such that $v_s = \sum_{s' \in S} \Delta(s, \alpha, s') \cdot v_{s'}$. Thus, since we have $v_s = \mathbb{P}_s^{\max}(\Diamond T)$, actions of $A^{\max}(s)$ maximise the probability of reaching T from s. Building a strategy that would arbitrarily choose a state of $A^{\max}(s)$ is not sufficient. Indeed, let us assume that we have a state $s \in S \setminus T$ in the MDP such that $A^{\max}(s) = \{\alpha, \beta\}$, where $\Delta(s, \beta, t) = 1$ for a certain $t \in T$ and $\Delta(s, \alpha, s) = 1$, we obviously have that choosing α doesn't allow to reach T passing by s.

A selection of actions ensuring the reachability to T in the MC induced by σ is required. Let $\mathcal{M}^{\max}$ be the MDP that corresponds to $\mathcal{M}$ where actions $\beta \in A(s) \setminus A^{\max}(s)$ are deleted from $A(s)$, for all s connected to T. By definition, $\mathbb{P}_s^{\max}(\Diamond T)$ is not affected by this simplification of $\mathcal{M}$.

For all s such that s is connected to T in the underlying graph of $\mathcal{M}^{\max}$, we denote by $||s||$ the length of the *shortest path* (in terms of steps) of s to any state of T in the underlying graph of $\mathcal{M}^{\max}$. Intuitively, computing $||s||$ allow to avoid that σ chooses actions that prevent s of reaching T.

- $||s|| = 0$ iff $s \in T$.
- Let $n \in \mathbb{N}_0$. By induction on n, we define $\sigma(s)$ for each s connected to T in the underlying graph of $\mathcal{M}^{\max}$ and such that $||s|| = n$. The strategy chooses an action $\sigma(s) \in A^{\max}(s)$ such that there exists a successor $s' \in Succ(s, \sigma(s))$, with s' connected to T in the underlying graph of $\mathcal{M}^{\max}$ and $||s'|| = n - 1$. An action $\sigma(s) \in A(s)$ is chosen arbitrarely for states s that are not connected to T in the underlying graph of $\mathcal{M}$.

We build σ in this way: let $s \in S$ be a state of $\mathcal{M}$ and $\mathbb{A}(s) = \{\alpha \in A^{\max}(s) \mid \exists s' \in Succ(s, \alpha), ||s'|| = ||s|| - 1\}$,

$$\sigma(s) = \alpha \text{ such that } \alpha \in \mathbb{A}(s)$$

A.3.2 Stochastic shortest path expectation problem

Let $\mathcal{M} = (S, A, \Delta, w, AP, L)$ be an MDP and $T \subseteq S$ be a set of target states of $\mathcal{M}$. We build an optimal strategy σ that minimises the expected cost of paths to reach T from all states $s \in S$ of $\mathcal{M}$. We thus first compute $\min_\sigma \mathbb{E}_s^{\sigma}(\Diamond T) = \mathbb{E}_s^{\min}(\Diamond T)$, for all $s \in S$. Let $\ell \in \mathbb{N}$ be a cost threshold. If $\mathbb{E}_s^{\sigma}(\Diamond T) \leq \ell$, σ satisfies the SSP-E problem [RRS15] for the state s. Otherwise, no strategy satisfying the SSP-E problem exist. Let $S_{=1} = \{s \in S \mid \mathbb{P}_s^{\max}(\Diamond T) = 1\}$ be the set of states that reach T with a maximum probability one and $(x_s)_{s \in S}$ be a vector for the following LP:

$$\max \sum_{s \in S_{=1}} x_s$$

subject to the constraints

$$\begin{aligned} x_s &= 0 && \forall s \in T, \\ x_s &= \infty && \forall s \in S \text{ such that } \mathbb{P}_s^{\max}(\Diamond T) < 1, \\ x_s &\leq w(\alpha) + \sum_{s' \in Succ(s,\alpha) \setminus T} \Delta(s, \alpha, s') \cdot x_{s'} && \forall s \in S_{=1} \setminus T, \forall \alpha \in A(s). \end{aligned}$$

The optimal solution $(v_s)_{s \in S}$ of this LP is unique and gives the following result:

$$v_s = \mathbb{E}_s^{\min}(\Diamond T) \quad \forall s \in S$$

We can now build an optimal pure memoryless strategy σ that minimises the expected cost of paths of $\mathcal{M}$ to reach T :

$$\sigma(s) = \arg \min_{\alpha \in A(s)} (w(\alpha) + \sum_{s' \in Succ(s,\alpha) \setminus T} \Delta(s, \alpha, s') \cdot v_{s'})$$

A.4 Stochastic shortest path percentile problem

We solve an SSP-P problem [RRS15] with the following algorithm:

Algorithm 8 Solving the SSP-P problem

Input : $\mathcal{M}$, an MDP with state space S, $s \in S$, a state of $\mathcal{M}$, $T \subseteq S$, a subset of target states, $\ell \in \mathbb{N}$, a cost threshold, and $\alpha \in [0, 1] \cap \mathbb{Q}$, a probability threshold.

Output : The optimal pure finite-memory strategy for the SSP-P problem for the state s, the set of target states T, the cost threshold ℓ, and the probability threshold α if such a strategy exists, $False$ else.

1: $\mathcal{M}_\ell \leftarrow$ unfold($\mathcal{M}$, s, ℓ) *(* build the unfolding of $\mathcal{M}$ from s up to ℓ *)*
2: $S_\ell \leftarrow$ state space of $\mathcal{M}_\ell$
3: $T_\ell \leftarrow \{(s, v) \in S_\ell \mid s \in T \ \wedge \ v \leq \ell\}$
4: solve the SR problem for $(s, 0)$, T_ℓ, and α in $\mathcal{M}_\ell$ *(* cf. Appendix A.3.1 *)*
5: **if** there exists a strategy satisfying this SR problem **then**
6: $\quad \sigma \leftarrow$ build the optimal strategy for this SR problem
7: $\quad$ **return** σ
8: **else**
9: $\quad$ **return** $False$

B.1 Bellman equation system for minimal expected cost-to-target

Let $\mathcal{M}$ be a finite MDP with state space S and with a probability transition function Δ, $s \in S$ be a state of $\mathcal{M}$, $T \subseteq S$ be a subset of target states, and $S_{=1} = \{s \in S \mid \mathbb{P}_s^{\max}(\Diamond T) = 1\}$. The vector $(x_s)_{s \in S}$ with $x_s = \mathbb{E}_s^{\min}(\mathsf{TS}^T)$ yields the unique solution of the following equation system:

- if $s \in T$, then $x_s = 0$,
- else if $s \notin S_{=1}$, then $x_s = \infty$,
- else,

$$x_s = \min_{\alpha \in A(s)} \Big(w(\alpha) + \sum_{s' \in Succ(s,\alpha)} \Delta(s, \alpha, s') \cdot x_{s'} \Big).$$

The linear program defined in Appendix A.3.2 is actually derived from this equation system.

C.1 Prism to Jani format using Storm

Let $\mathcal{M}$ be the MDP defined as follows in the Prism language:

```
mdp

module simple_mdp

s: [0..2] init 0;

[beta] s=0 -> 0.5: (s'=1) + 0.5: (s'=2);
[gamma] (s=1 | s=2) -> 1: (s'=0);
[alpha] s=2 -> 1: (s'=2);

endmodule

label "a" = s=0;
label "b" = s=1;

rewards "weights"
  [alpha] true: 5;
  [beta] true: 3;
  [gamma] true: 2;
endrewards
```

$\mathcal{M}$ is actually the MDP of Figure 4.1. Storm allows to convert this model to Jani with the following command :

```
storm-pars --prism2jani --prism simple_mdp.prism --exportJani:jani-output simple_mdp.jani
```

The Jani ouput is the following :


```
{
    "actions": [
        {
            "name": "alpha"
        },
        {
            "name": "beta"
        },
        {
            "name": "gamma"
        }
    ],
    "automata": [
        {
            "edges": [
                {
                    "action": "alpha",
                    "assignments": [
                        {
                            "ref": "weights",
                            "value": 5
                        }
                    ],
                    "destinations": [
                        {
                            "assignments": [
                                {
                                    "ref": "s",
                                    "value": 2
```

```
                        }
                    ],
                    "location": "l"
                }
            ],
            "guard": {
                "exp": {
                    "left": "s",
                    "op": "=",
                    "right": 2
                }
            },
            "location": "l"
        },
        {
            "action": "beta",
            "assignments": [
                {
                    "ref": "weights",
                    "value": 3
                }
            ],
            "destinations": [
                {
                    "assignments": [
                        {
                            "ref": "s",
                            "value": 1
                        }
                    ],
                    "location": "l",
                    "probability": {
                        "exp": 0.5
                    }
                },
                {
                    "assignments": [
                        {
                            "ref": "s",
                            "value": 2
                        }
                    ],
                    "location": "l",
                    "probability": {
                        "exp": 0.5
                    }
                }
            ],
            "guard": {
                "exp": {
                    "left": "s",
                    "op": "=",
                    "right": 0
                }
            },
            "location": "l"
        },
        {
            "action": "gamma",
            "assignments": [
                {
                    "ref": "weights",
                    "value": 2
                }
            ],
            "destinations": [
                {
                    "assignments": [
                        {
                            "ref": "s",
                            "value": 0
                        }
                    ],
                    "location": "l"
                }
            ],
            "guard": {
                "exp": {
                    "left": "s",
                    "op": "=",
                    "right": 1
                }
            },
            "location": "l"
        },
        {
            "action": "gamma",
            "assignments": [
                {
                    "ref": "weights",
                    "value": 2
                }
            ],
            "destinations": [
                {
```

```
                        "assignments": [
                            {
                                "ref": "s",
                                "value": 0
                            }
                        ],
                        "location": "l"
                    }
                ],
                "guard": {
                    "exp": {
                        "left": "s",
                        "op": "=",
                        "right": 2
                    }
                },
                "location": "l"
            }
        ],
        "initial-locations": [
            "l"
        ],
        "locations": [
            {
                "name": "l",
                "transient-values": [
                    {
                        "ref": "a",
                        "value": {
                            "left": "s",
                            "op": "=",
                            "right": 0
                        }
                    },
                    {
                        "ref": "b",
                        "value": {
                            "left": "s",
                            "op": "=",
                            "right": 1
                        }
                    }
                ]
            }
        ],
        "name": "simple_mdp",
        "variables": []
    }
],
"constants": [],
"features": [
    "derived-operators"
],
"jani-version": 1,
"name": "jani_from_prism",
"properties": [],
"restrict-initial": {
    "exp": true
},
"system": {
    "elements": [
        {
            "automaton": "simple_mdp"
        }
    ]
},
"type": "mdp",
"variables": [
    {
        "initial-value": false,
        "name": "a",
        "transient": true,
        "type": "bool"
    },
    {
        "initial-value": false,
        "name": "b",
        "transient": true,
        "type": "bool"
    },
    {
        "initial-value": 0.0,
        "name": "weights",
        "transient": true,
        "type": "real"
    },
    {
        "initial-value": 0,
        "name": "s",
        "type": {
            "base": "int",
            "kind": "bounded",
            "lower-bound": 0,
            "upper-bound": 2
        }
```

```
        }
    ]
}
```