

Reinforcement Planning for Effective ε -Optimal Policies in Dense Time with Discontinuities

Léo Henry ✉ 

University College London, UK

Blaise Genest ✉ 

CNRS and CNRS@CREATE, IPAL, France

Institute for Infocomm Research (I2R), Singapore

Alexandre Drewery ✉

ENS Rennes, France

Abstract

Lately, the model of (Decision) Stochastic Timed Automata (DSTA) has been proposed, to model those Cyber Physical Systems displaying dense time (physical part), discrete actions and discontinuities such as timeouts (cyber part). The state of the art results on controlling DSTAs are however not ideal: in the case of infinite horizon, optimal controllers do not exist, while for timed bounded behaviors, we do not know how to build such controllers, even ε -optimal ones.

In this paper, we develop a theory of Reinforcement Planning in the setting of DSTAs, for discounted infinite horizon objectives. We show that optimal controllers do exist in general. Further, for DSTAs with 1 clock (which already generalize Continuous Time MDPs with e.g. timeouts), we provide an effective procedure to compute ε -optimal controllers. It is worth noting that we do *not* rely on the discretization of the time space, but consider symbolic representations instead. Evaluation on a DSTA shows that this method can be more efficient. Last, we show on a counterexample that this is the furthest this construction can go, as it cannot be extended to 2 or more clocks.

2012 ACM Subject Classification Theory of computation → Timed and hybrid models

Keywords and phrases reinforcement planning, timed automata, planning

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.13

Related Version *Long Version:* <http://perso.crans.org/genest/HGD23.pdf> [13]

Funding DesCartes: this research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme, by ANR-20-CE25-0012 MAVeriQ, by PEPR-AI SAIF and by EPSRC Standard Grant CLeVer (EP/S028641/1).

1 Introduction

Controlling *Cyber Physical Systems* (CPSs) is an important task to optimize resources of today's complex systems. It is also challenging, as CPSs often mix discrete control, continuous state space, time and stochasticity, between others. A well-studied model of CPSs is Continuous-Time Markov Decision Processes (CTMDPs). On the theoretical side, optimal controllers exist in the timed-bounded horizon setting [3, 16, 8], with infinite horizon [6, 12], or with discounted infinite horizon [7]. Yet, building such optimal controller might not be as simple, even in the time bounded case [15]. On the practical side, there are many algorithms to produce ε -optimal controllers [16, 8]. CTMDPs are fully continuous models, expanded in the previous decade by *Decision Stochastic Timed Automata* (DSTA, see [5, 4]), able to express hard real-time constraints (e.g. timeouts, deadlines) using real-valued clocks and guards on transitions stemming from the *Timed Automata* [1] and *Stochastic TAs* [2] formalisms. In terms of DSTAs, CTMDPs are DSTAs with a single clock reset on all transitions and no guards.



© Léo Henry, Blaise Genest, and Alexandre Drewery;
licensed under Creative Commons License CC-BY 4.0

43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).

Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 13; pp. 13:1–13:18



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Theoretically, this ability to model hard constraints is very challenging as it produces discontinuities. It breaks the proof for the existence of optimal controller for CTMDPs and to produce ε -optimal policies in practice (except in the special case of value 1 [4]). In [5], an existential theorem for optimal policies for timed bounded horizons in DSTA has been proved in a *non*-constructive manner. However, besides the fact that they can be chosen positional, little is known on the kind of policies necessary to achieve optimality.

In this paper, we develop a *Reinforcement Planning (RP)* framework [18] to prove (Theorem 7) the existence of optimal positional policies for DSTA in general for unbounded horizon equipped with a discount. This is the best possible result, as optimal policies do not exist in general for unbounded horizon without discount [5].

Our main results concern ε -approximations: we prove that for one-clock DSTAs, which are already more powerful than CTMDPs (the unique clock needs not be reset at every transition) and allows to model discontinuous hard constraints such as timeouts, *polytope policies* (defined hereafter) suffices. A polytope policy partitions $[0, \infty)$ (value of the clock) with a finite number of intervals and plays one action on each interval. Technically, we develop a symbolic value iteration algorithm, creating polytopes only when necessary using *symbolic functional analysis* (Lemma 18); and then use *numerical analysis* to effectively compute approximations of these polytopes (Lemma 20). Last, we show that this is the furthest this method can go, as in the presence of 2 clocks or more, the optimal policy after n iterations is not a polytope policy (Theorem 9). A long version is available [13].

Related Work. Reinforcement *Planning* [18, 10] provides foundations to reason about observations for completely specified models, that ground Reinforcement *Learning* (e.g. Monte-Carlo method) which tackle un- or semi-specified systems. RP provides implementable algorithms for fully specified models, which is also the case of our methodology, although the implementation would require a heavy use of both symbolic functional as well as numerical analysis tools which are very technical and tedious to implement. RP has first been theorized for discrete models with finite state space (MDPs) [18]. Some version exist for a totally continuous world (continuous state space, continuous action, continuous behavior) [10]. Some RP frameworks have been developed for some specific model with discrete actions and continuous state space: CTMDPs [7], ODEs [19], but without any discontinuous behaviors such as deadline or timeouts. We did not find any RP methodology which can be applied to a setting mixing continuous state space, discrete action set and non-continuous behaviors (timeouts, deadlines). The closest related work is [9], where the authors design an effective algorithm to produce ε -optimal policies. The main differences are that the method is very different, discretizing time in many modes of ε -size (depending on the ε -precision desired), whereas we compute a minimal number of (larger) modes using symbolic function analysis; and the model considered is different, namely Markov Timed Automata (MTA). Compared with DSTA, MTA cannot model urgency: time can always elapse, and all distributions are exponential, whereas DSTA can enforce leaving a state before some time-out using uniform distributions. Further, uniform distributions are harder to handle than exponential ones. It is thus unclear how to lift similar time-discretization bounds for DSTA. We numerically evaluate on a DSTA the efficiency of our technique in Section 5.3 and compare with the complexity of time-discretization for a MTA of similar size.

2 Decision Stochastic Timed Automata

We start by defining the model of CPS we are considering in the paper, namely Decision Stochastic Timed Automata (*DSTA* for short), extending CTMDPs with hard constraints.

2.1 Notations

We fix finite sets \mathcal{C} of clocks, and \mathcal{V} of *valuations over \mathcal{C}* , i.e. the set of all maps of $\mathcal{C} \rightarrow \mathbb{R}_{\geq 0}$ (clocks always have non-negative values). Given a valuation $v \in \mathcal{V}$, we define two operations:

Time elapse $v + \delta$ with $\delta \in \mathbb{R}_{\geq 0}$ is a valuation such that for all $x \in \mathcal{C}$ $(v + \delta)(x) = v(x) + \delta$;

Reset $\mathcal{C}' \subseteq \mathcal{C}$ $v_{[\mathcal{C}' \leftarrow 0]}$ is the valuation $x \mapsto 0$ if $x \in \mathcal{C}'$ and $x \mapsto v(x)$ otherwise.

We will sometimes consider a valuation v as a vector in $\mathbb{R}_{\geq 0}^{|\mathcal{C}|}$.

We denote \mathcal{G} the set of guards, i.e. of constraints of the form $\bigwedge_{1 \leq i \leq |\mathcal{C}|} (x_i \in I_i)$ and \mathcal{I} the set of *invariants*, i.e. guards featuring intervals with no lower bound. The set of *constraints over a set of clocks \mathcal{C}* is the set of all constraints of the form $(\sum_{1 \leq i \leq |\mathcal{C}|} a_i \times x_i) \bowtie q$ with $a_i \in \mathbb{Q}$, $x_i \in \mathcal{C}$, I_i an interval of \mathbb{Q} , $\bowtie \in \{<, \leq, =, \geq, >\}$ and $q \in \mathbb{Q}$. A *polytope* $z \in \mathcal{Z}$ is a finite set of constraints over \mathcal{C} . Let $z \in \mathcal{Z}$ and $v \in \mathcal{V}$. We write $v \models z$ if and only if v satisfies all constraints in z . A polytope z defines a subset $\mathcal{V}(z)$ of $\mathbb{R}^{|\mathcal{C}|}$ containing the valuations v such that $v \models z$. We define $z_{[\mathcal{C}' \leftarrow 0]}$ as the polytope such that $\mathcal{V}(z_{[\mathcal{C}' \leftarrow 0]}) = \{v_{[\mathcal{C}' \leftarrow 0]} \mid v \in \mathcal{V}(z)\}$.

For a function P from some set to booleans we note $1_P : y \mapsto 1$ iff $P(y)$ and otherwise 0. We write $\mathcal{D}(Y)$ for the set of all probabilities distributions over a set Y .

2.2 Decision Stochastic Timed Automata

We consider DSTAs [5, 4], transition systems defining probabilities on sets of runs by assigning probability distributions to delays, up to a policy dictating the transitions taken.

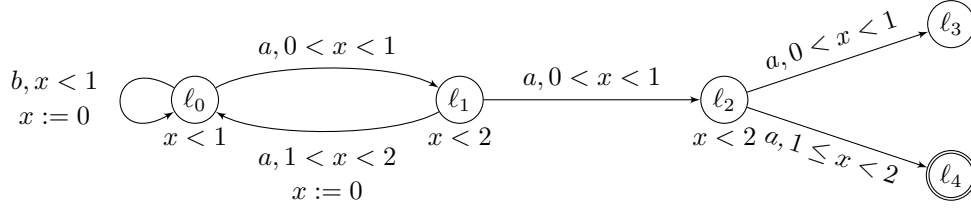
- **Definition 1.** A DSTA over clocks \mathcal{C} and actions A is a tuple $\mathcal{A} = (\mathcal{L}, E, \ell_0, I, \mu)$ with
- \mathcal{L} a set of control state and $\ell_0 \in \mathcal{L}$ is the initial location,
 - E a set of transitions of the form $(\ell_1, a, g, r, \ell_2) \in \mathcal{L} \times A \times \mathcal{G} \times 2^{\mathcal{C}} \times \mathcal{L}$, where g is the guard of the transition and r is the finite set of clocks to reset when taking the transition,
 - $I : \mathcal{L} \rightarrow \mathcal{I}$ defines the invariants the clock values need to satisfy to stay in a control state,
 - $\mu = (\mu_{(\ell, v)})_{(\ell, v) \in \mathcal{L} \times \mathcal{V}}$ defines probability distributions over delays for each control state and clock valuation. For $I(\ell, v) = \{\delta \in \mathbb{R}^+ \mid v + \delta \models I(\ell)\}$, it is of one of the following type:
 - if $I(\ell, v)$ is the singleton $\{0\}$, then μ is the Dirac distribution $\mu(0) = 1$,
 - otherwise, if $I(\ell, v)$ is bounded, then μ is the uniform distribution over $I(\ell, v)$,
 - otherwise, $I(\ell, v) = [0, \infty)$ and μ is an exponential distribution $\mu(\delta) = e^{-\alpha_\ell \delta}$ for some parameter $\alpha_\ell \in \mathbb{R}_{>0}$ independent over the valuation v .

Intuitively, a move in a DSTA \mathcal{A} from a configuration $(\ell, v) \in \mathcal{L} \times \mathcal{V}$ is done in two steps: first, a delay δ is randomly chosen according to the distribution $\mu_{(\ell, v)}$, and then the policy chooses a transition $e = (\ell, a, g, r, \ell')$ enabled from $(\ell, v + \delta)$, reaching $(\ell', (v + \delta)_{[r \leftarrow 0]})$.

Formally, we define a *partial run* of a DSTA as a sequence of transitions and delays $\rho = (e_i, \delta_i)_{1 \leq i \leq n}$ starting from a configuration (ℓ_1, v_1) such that a corresponding sequence of configurations can be defined: $\rho = ((\ell_i, v_i) \xrightarrow{e_i} (\ell_{i+1}, v_{i[r_i \leftarrow 0]}) \xrightarrow{\delta_i} (\ell_{i+1}, v_{i[r_i \leftarrow 0]} + \delta_i))_{1 \leq i \leq n}$ (notably, the transitions e_i are enabled in (ℓ_i, v_i)). A *run* is a partial run starting in the initial configuration $(\ell_0, \mathbf{0})$. We note $Runs(\mathcal{A})$ (resp. $PRuns(\mathcal{A}, (\ell, v))$) the set of runs (resp. partial runs) of a DSTA \mathcal{A} (resp. starting from a configuration (ℓ, v)).

To start with a random passing of time, we can add a dummy state **init** with no incoming transition. Every action from **init** leads to ℓ_0 , resetting all clocks, so that any random delay can be taken in **init**. We will not depict such a state **init** in the different figures.

- **Example 2.** Figure 1 depicts DSTA \mathcal{A} . In \mathcal{A} , the user can play b in ℓ_0 to roll for a new clock value. When they play a , they are sent to ℓ_1 . Depending on the delay randomly chosen in ℓ_1 , the control states either goes back to ℓ_0 ($x > 1$) and resets the clock x , or it moves to



■ **Figure 1** A DSTA with one clock x with initial state ℓ_0 . The goal is to reach ℓ_4 .

ℓ_2 ($x < 1$) from which another delay is randomly drawn and added to x : if this delay makes $x > 1$, then the run goes to winning state, else it goes to a losing state. Intuitively, playing a from ℓ_0 with x very close to 1 is optimal, as then the chance to get stuck in the losing state is very small (the sum of the delays from ℓ_1 and ℓ_2 needs to be smaller than $1 - x$).

To discuss the semantics of DSTAs, we have to define the probabilities some behaviors have to appear. As (almost) every run has a null probability to appear, we discuss at the level of (partial) *paths*, i.e. sequences of transitions. A path $\pi = (e_i)_{1 \leq i \leq n}$ represents the set of runs ρ such that there exist delays $(\delta_i)_{1 \leq i \leq n}$ with a run $\rho = (e_i, \delta_i)_{1 \leq i \leq n}$. We say that a run ρ *models* a path $\pi \in E^*$ when their sequences of transitions match. A partial path is said *feasible* when it has a model. We call *path* a partial path that is modeled by a (initial) run. For paths, we will only consider (initial) runs as models. We write $PPaths(\mathcal{A})$ (resp. $Paths(\mathcal{A})$) for the set of partial paths (resp. paths) of an automaton \mathcal{A} .

2.3 Rewards and policies

We consider a discounted average reward model as the objective function (as usual in Reinforcement Planning), by contrast with winning state and infinite horizon (reward = 1 if winning state ever reached, 0 otherwise) or finite horizon (reward = 1 if winning state reached before the time horizon, 0 otherwise). More precisely, each transition is associated with a reward, and we consider the *discounted sum* of all rewards as our objective function, with immediate rewards of greater interest than later ones. Discount allows to tackle unbounded runs, while optimal policies do not exist for unbounded horizons without discount [5].

► **Definition 3.** Consider a DSTA $\mathcal{A} = (\mathcal{L}, E, \ell_0, I, \mu)$. A reward function is a function $\mathcal{R} : E \rightarrow \mathbb{N}$ associating a reward to each transition. Given discount $0 < \gamma < 1$, the discounted reward of a run $\rho = (e_i, v_i)_i$ or of a path $\pi = (e_i)_i$ is $\mathcal{R}(\rho) = \mathcal{R}(\pi) = \sum_i \gamma^{i-1} \mathcal{R}(e_i)$.

The rewards associated with Figure 1 are 0 on every transition except on from ℓ_2 to ℓ_4 , where the reward is 1. Hence a path reaching ℓ_4 in i steps will get discounted reward γ^i .

The decisions made by the agent are formalized as *policies*. Policies depend on the type of input they can use to take a decision: *generic policies* can use any information about the history of the run to make stochastic choices, while more restrictive *pure positional policies* can only use the current configuration and decide on a given transition deterministically.

► **Definition 4.** A policy σ on a DSTA $\mathcal{A} = (\mathcal{L}, E, \ell_0, I, \mu)$ is a function $\sigma : Runs(\mathcal{A}) \rightarrow \mathcal{D}(E)$. A pure positional policy is a policy σ such that for all pairs of runs ρ, ρ' ending in the same configuration $(\ell, v) \in \mathcal{L} \times \mathcal{V}$, we have $\sigma(\rho) = \sigma(\rho')$ is a Dirac distribution in one action. We can thus represent any pure positional policy as a function $\sigma : (\mathcal{L} \times \mathcal{V}) \rightarrow E$.

Given a policy σ and a history $\rho \in \text{Runs}(\mathcal{A})$ ending in (ℓ, v) , we can define the probability $P_\sigma(\pi \mid \rho)$ for the behavior of the DSTA to model the path $\pi = (e_i)_{1 \leq i \leq n}$ after ρ as follows:

$$P_\sigma(\pi \mid \rho) = \sigma(\rho)(e_1) \int_{\delta \in I(\ell_1, v_{[r_1 \leftarrow 0]})} P_\sigma((e_i)_{2 \leq i \leq n} \mid \rho \cdot (e_1, \delta)) d\mu_{(\ell_1, v_{[r_1 \leftarrow 0]})}$$

with ℓ_1 the target location of e_1 and r_1 the set of clocks it resets. Given a reward function \mathcal{R} this can be used to assign a *value* to a policy.

► **Definition 5.** Fix a reward function \mathcal{R} on a DSTA \mathcal{A} with discount γ and an integer $n \in \mathbb{N}$. The value of a policy σ after a history $\rho \in \text{Runs}(\mathcal{A})$ for n steps is

$$\text{Val}_\sigma^n(\rho) = \sum_{\pi=(e_i)_{1 \leq i \leq n} \in E^n} P_\sigma(\pi \mid \rho) \mathcal{R}(\pi).$$

The value of a policy for history ρ is $\text{Val}_\sigma(\rho) = \lim_{n \rightarrow \infty} \text{Val}_\sigma^n(\rho)$. We write $\text{Val}^*(\rho) = \sup_\sigma \text{Val}_\sigma(\rho)$ for the optimal value. The value $\text{Val}_{(\sigma)}^*(\mathcal{A}) = \text{Val}_{(\sigma)}^*(\lambda)$, denoted $\text{Val}_{(\sigma)}^*$, is the value associated with the empty run. A policy σ^* is said optimal when $\text{Val}^* = \text{Val}_{\sigma^*}$.

Rewards and values are bounded with known bounds. We write MaxReward (resp. MaxValue) for the supremal reward (resp. value). Similarly, we adopt the notation $\text{Val}_\sigma(\rho, e)$ (resp. $\text{Val}^*(\rho, e)$) to express the value of first playing e and then following σ (resp. having an optimal value). Finally, for (pure) positional policy, we replace ρ by a configuration (ℓ, v) (e.g. $\text{Val}_\sigma(\ell, v)$). We say that a policy σ is ε -optimal whenever $\text{Val}_\sigma(\mathcal{A}) \geq \text{Val}^*(\mathcal{A}) - \varepsilon$.

In general, one cannot represent finitely a measurable policy, nor even a pure positional policy even for a DSTA with a unique clock. We now define a class of policies that can be finitely described, called *polytopes policies* [5]. Given a control state ℓ , it partitions \mathcal{V} with a finite number of polytopes and assigns a unique action to each polytope.

► **Definition 6.** Let \mathcal{A} be a DSTA. A pure positional policy σ is a polytope policy if for each location ℓ of \mathcal{A} there exists a finite set of polytopes Z_1, \dots, Z_n that partition \mathcal{V} such that $\forall 1 \leq i \leq n, \forall v, v' \in Z_i, \sigma(\ell, v) = \sigma(\ell, v')$.

3 Main Results

Our first contribution is to show, by adapting the RP framework to DSTA (see Section 4, which serves as proof of Theorem 7), that *pure positional* optimal policies exist:

► **Theorem 7.** Let \mathcal{A} be a DSTA. Then there exists a pure positional policy σ^* such that for every policy σ and every run ρ , $\text{Val}_{\sigma^*}(\rho) \geq \text{Val}_\sigma(\rho)$.

Notice that discounted infinite horizon is the most generic reward scheme possible enjoying the existence of optimal policies, as it has been shown in [5] that optimal policies do not exist in general for undiscounted infinite horizon. This matches the existence of optimal policies for finite horizon properties of [5], with a vastly different proof technique.

Concerning ε -optimal policies, we show that the class of *polytope policies*, which are finitely describable pure positional policies, are sufficient for DSTA with a unique clock $\mathcal{C} = \{x\}$. The main difficulty is that there are *uncountably many* configurations, and from each a (pure positional) policy needs to provide which action to play. We do so by developing a *symbolic* value iteration algorithm and prove in Lemma 18 that the policies generated are polytope policies, using symbolic functional analysis. Even harder is to actually

compute effectively a ε -approximation from the *uncountably many* configurations, and not only dealing with existence: Using numerical functional analysis, we explain how to compute an approximation of the finitely many positions of the changes of sign of the value function, without approximating the value function itself. This is our second and main contribution:

► **Theorem 8.** *Let \mathcal{A} be a DSTA with a unique clock and $\varepsilon > 0$. Then there exists an effectively constructible positional polytope policy σ_ε that is ε -optimal.*

Section 5 serves as proof of Theorem 8: Lemma 18 shows that there exist an *optimal* polytope policy σ_n for n steps of value iteration. This is complex as we do not know how to provide a close-form solution for the value function itself, and we conjecture that it is a hard problem. Eventually, after n iterations, σ_n will be $\frac{\varepsilon}{2}$ -optimal. Using numerical analysis, we show in Lemma 20 in Section 5 how to compute effectively a polytope policy σ whose value is at most $\frac{\varepsilon}{2}$ away from the value of σ_n , providing us with a polytope policy that is ε -optimal. This is in stark contrast with the non-constructive proof of [5], which needs uncountable time to produce ε -optimal policies.

According to Lemma 18, polytope policies are optimal for DSTA with 1 clock and without loops, as the value is met after a finite number of steps of value iteration. Our third contribution is that polytope policies are not optimal with 2 clocks and without loops.

► **Theorem 9.** *There is a DSTA \mathcal{A} with 2 clocks and without loops such that for all polytope policies σ , there exists a policy σ' with $\text{Val}_{\mathcal{R},\gamma}(\mathcal{A}, \sigma) < \text{Val}_{\mathcal{R},\gamma}(\mathcal{A}, \sigma')$.*

The proof can be found in Section 5.7. It is inspired by [5], where it is shown polytope policies are not optimal for finite horizon properties, even for DSTA with a *unique* clock. We use the second clock to simulate the bound on the time horizon. Theorem 8 shows that for one clock, the setting of discounted infinite horizon has much more desirable properties than the one of bounded time. Notice that polytope policies could still be ε -optimal in general.

4 Bellman equations in dense time

We extend Reinforcement Planning [18] to DSTA models. We first focus on the introduction of Bellman Equations for this continuous setting. The Bellman Equations are the theoretical basis of RP, as they allow to incrementally compute or approximate the (optimal) values.

4.1 General policies

We first reason on general policies in order to prove the optimality of the *pure positional policies*. Once their optimality is proved, the discussion is restricted to this simpler class. We first define the Bellman Equations in their most general form for DSTA.

► **Proposition 10.** *Given a policy σ and an history ρ ending in configuration (ℓ, v) , $\text{Val}_\sigma(\rho)$ is the only bounded solution of the **Bellman Equation in dense time**:*

$$\text{Val}_\sigma(\rho) = \sum_{e=(\ell, a, g, r, \ell') \in E} \sigma(\rho)(e) \times \left(\mathcal{R}(e) + \gamma \times \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \text{Val}_\sigma(\rho \cdot (e, \delta)) d\mu_{(\ell', v_{[r \leftarrow 0]})} \right).$$

In a similar fashion, $\text{Val}_\sigma(\rho, e)$ for $e = (\ell, a, g, r, \ell')$ is the unique solution of:

$$\text{Val}_\sigma(\rho, e) = \mathcal{R}(e) + \gamma \times \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \sum_{e'=(\ell', a, g, r, \ell'') \in E} \sigma(\rho)(e') \times \text{Val}_\sigma(\rho \cdot (e, \delta), e') d\mu_{(\ell', v_{[r \leftarrow 0]})}.$$

Proof sketch. We prove that the value is a solution by using the definition of Val as the limit of Val^n and unfolding it (taking the first action out of the limit). The uniqueness of the bounded solution is proved by comparing two potentially different one: $F(\rho) - F'(\rho)$. Using the equation allows to reduce their difference step by step as the first reward does not depend on F / F' . After n steps, the difference is bounded by γ^n times a probability distribution of differences. As the functions are bounded, taking the limit proves the uniqueness. ◀

From Bellman's equations, the following relations can be inferred:

► **Corollary 11.** *Given a policy σ and a run ρ ending in (ℓ, v) :*

$$\text{Val}_\sigma(\rho) = \sum_{e \in E} \sigma(\rho)(e) \cdot \text{Val}_\sigma(\rho, e) \quad \text{and} \quad \text{Val}_\sigma(\rho, e) = \mathcal{R}(e) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \text{Val}_\sigma(\rho(e, \delta)) d\mu_{(\ell', v_{[r \leftarrow 0]})}$$

with r the reset of e and ℓ' its target.

For any policy, we can construct a pure positional policy at least as efficient:

► **Proposition 12.** *For any policy σ , there exists a pure positional policy σ' such that $\text{Val}_{\sigma'} \geq \text{Val}_\sigma$. In particular one such policy can be defined from σ in the following way: consider a history ρ ending in configuration (ℓ, v) , then*

$$\sigma'(\ell, v) = \arg \max_{e \in E} \sup_{\rho' \text{ ending in } (\ell, v)} \left(\mathcal{R}(e) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \text{Val}_\sigma(\rho' \cdot (e, \delta)) d\mu_{(\ell, v_{[r \leftarrow 0]})} \right).$$

Proof sketch. Using Bellman Equation on Val_σ allows to see that it is lower than what is obtained by playing optimally at the next step (on all executions) and then following σ . Applying the same reasoning n times allows to show that it is lower than Val_σ^n , plus a bounded term depending of Val_σ discounted by γ^n . Taking the limit then allows to conclude. ◀

4.2 Positional policies

As we now know from Proposition 12 that positional policies are as efficient as general policies, the rest of this paper mentions only pure positional policies. We will use the Bellman Equation in their restricted form for pure positional policies, provided now:

$$\text{Val}_\sigma(\ell, v) = \mathcal{R}(\sigma(\ell, v)) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \text{Val}_\sigma(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})}$$

with ℓ' the target location of $\sigma(\ell, v)$.

$$\text{Val}_\sigma(\ell, v, e) = \mathcal{R}(e) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \text{Val}_\sigma(\ell', v_{[r \leftarrow 0]} + \delta, \sigma(\ell', v_{[r \leftarrow 0]} + \delta)) d\mu_{(\ell', v_{[r \leftarrow 0]})}$$

with $e(l, x, a) = (l, a, g, r, \ell')$. The Bellman Equations rely on the integral of the value on the interval of possible delays. We call this the *interval value* function, and write $\text{IVal}_\sigma(\ell, v) = \int_{\delta \in I(\ell', v)} \text{Val}_\sigma(\ell', v + \delta) d\mu_{(\ell', v)}$. We have $\text{Val}_\sigma = \mathcal{R}(\sigma(\ell, v)) + \gamma \cdot \text{IVal}_\sigma(\ell', v_{[r \leftarrow 0]})$.

We can show that if locally deviating from a value function according to a policy always gives a better result, then globally changing to the new policy enhances the expected return.

► **Proposition 13.** Consider a function F such that for all configurations (ℓ, v) $F(\ell, v, e) = \mathcal{R}(e) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} F(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})}$ with ℓ' the target of e . For a policy σ , if for all configurations (ℓ, v) , $F(\ell, v, \sigma(\ell, v)) \geq F(\ell, v)$ then $\text{Val}_\sigma \geq F$. Furthermore $\text{Val}_\sigma > F$ if and only if there exists a configuration where the inequality is strict.

Proof sketch. The result is obtained as previously by unfolding the equation for n step before using the discount and the boundedness of functions to conclude. ◀

Corollary 16 apply this to obtain a greedy improvement from the current policy. We now state the *Bellman optimality equations in dense time*, characterizing the optimal values.

► **Proposition 14.** Val^* is the only bounded solution of the **Bellman Optimality Equation**:

$$\text{Val}^*(\ell, v) = \max_{e \in E} \left(\mathcal{R}(e) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \text{Val}^*(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})} \right)$$

with r the resets of e and ℓ' its target location. Similarly:

$$\text{Val}^*(\ell, v, e) = \mathcal{R}(e) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \max_{e \in E} \text{Val}^*(\ell, v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})} .$$

Proof sketch. The proof goes as for Bellman Equations: unfolding the definition of Val^* on one step allows to prove that it is a solution, while uniqueness is showed by comparing two solutions and using γ to limit their difference to 0 in the limit. ◀

We will again write $\text{IVal}^*(\ell, \text{val}) = \int_{\delta \in I(\ell, v)} \text{Val}^*(\ell', v + \delta) d\mu_{(\ell', v)}$. As for values of policies, we use it to deduce relations between optimal values for configurations and transitions.

► **Corollary 15.** For any configuration (ℓ, v) and transition $e \in E$ we have the following equalities: $\text{Val}^*(\ell, v) = \max_{e \in E} \text{Val}^*(\ell, v, e)$ and $\text{Val}^*(\ell, v, e) = \mathcal{R}(e) + \text{IVal}^*(\ell', v_{[r \leftarrow 0]})$

Using these equations and Proposition 13, we obtain two important results: the policy improvement theory works as for MDPs (it has no local minima) and the pure positional policies are optimal.

► **Corollary 16.** Given a policy σ , we can deduce a positional pure policy σ' such that $\forall (\ell, v) \in \mathcal{L} \times \mathcal{V}$, $\sigma'(\ell, v) = \arg \max_{e \in E} (\mathcal{R}(e) + \gamma \cdot \text{IVal}_\sigma(\ell', v_{[r \leftarrow 0]}))$ with r the reset of e and ℓ' its target. Then $\text{Val}_{\sigma'} \geq \text{Val}_\sigma$. Furthermore, if $\text{Val}_{\sigma'} = \text{Val}_\sigma$ then σ is optimal.

We now restate Theorem 7 in this context and prove it:

► **Theorem 17.** There exists a pure positional policy σ^* such that $\text{Val}_{\sigma^*} = \text{Val}^*$.

Proof sketch. For each configuration, we consider the action which is associated with the maximal expected reward when being played from this configuration (chosen arbitrarily from the actions with the same expected maximal reward). We call such an action an optimal action. We consider the positional policy which plays these optimal actions from each configuration. Using Proposition 13, we can show that such a policy is indeed optimal. ◀

Notice that the above proofs crucially rely on the fact that the discount $\gamma < 1$. Indeed, for $\gamma = 1$ (infinite horizon reachability *without* discount), playing positionally the optimal action may lead to very bad outcome. E.g., in Figure 1, the optimal action from $(\ell_0, x < 1)$ is to play action b , looping forever in location ℓ_0 , as the expected reward associated with playing b is higher (the supremal value is 1) than the reward from playing a which has some risk associated with it (of value $1 - \varepsilon$). When there is a discount $\gamma < 1$, playing a becomes optimal from $(\ell_0, x < 1)$ for x close to 1, as waiting for a better value of x is not worth it.

5 Computing ε -optimal Policies

5.1 Mode Policies for DSTA with 1 clock

We now restrict ourselves to DSTA with a single clock x . 1-clock polytopes are intervals (called modes), i.e. they can be represented by two bounds in $\mathbb{R}_{\geq 0} \cup \{+\infty\}$. We call *mode policies* such polytope policies. A mode policy σ of DSTA \mathcal{A} with control state \mathcal{L} is represented in the following finite way: $\sigma = (\mathcal{M}_\sigma^\ell, \sigma^\ell)_{\ell \in \mathcal{L}}$ with:

- \mathcal{M}_σ^ℓ is a partition of $\mathbb{R}_{\geq 0}$ in a finite set of modes $M_1 \dots M_k \in \mathcal{M}_\sigma^\ell$ (the modes of ℓ),
- For all $\ell \in \mathcal{L}$, we have $\sigma^\ell : \mathcal{M}_\sigma^\ell \rightarrow E$ gives the action associated with each mode of ℓ .

We will represent the bounds of the interval with rational numbers in \mathbb{Q} , wlog as we only need ε -approximations. Mode-policies allow for a finite number of points of discontinuity.

Closeness or openness of intervals is not a concern as there is probability 0 to reach the exact boundary *between two modes* in a control state. What happens at b for the maximal mode $[a, b]$ of a control state might be important though (e.g. with $b = a$ caused by an invariant), so we have one special mode at the end of the interval for $[b, b]$ whenever $b < +\infty$.

5.2 Symbolic Value Iteration for mode policies

We proceed in two steps: First, we provide our symbolic value iteration Algorithm 1 to define the exact mode policy σ_i^* obtained after i steps of the value iteration. Proposition 19 shows that the value Val_i^* of σ_i^* converges towards the optimal value, and given an $\varepsilon > 0$, it provides an effectively computable i such that $\text{Val}_i^* > \text{Val}^* - \frac{\varepsilon}{2}$. Lemma 18 proves that σ_i^* has a finite number of modes, for all $i \in \mathbb{N}$. We never compute the value function itself (we do not know how to represent it). Further, modes, e.g. the exact value of the (possibly non-rational) boundaries, cannot be computed perfectly.

The second step handles these issues, by computing effectively a mode policy $\bar{\sigma}_1$ approximating σ_1^* , such that its value is at least $\bar{\text{Val}}_1 > \text{Val}_1^* - \varepsilon_1$. We set ε_1 sufficiently small such that inductively, we obtain a policy $\bar{\sigma}_i$ with value $\bar{\text{Val}}_i > \text{Val}_i^* - \frac{\varepsilon}{2}$. Overall, $\bar{\text{Val}}_i > \text{Val}^* - \varepsilon$.

To ease the presentation, we assume that only two actions $\{\alpha, \beta\}$ are available. It is easy to extend to more actions by comparing actions pairwise. We focus on modes where both α, β are allowed, as the policy is trivial if only one action is allowed. On these modes, Val is continuous. Recall that for $I(\ell) = [a, b]$, we denote $\text{IVal}(\ell, x) = \int_x^b \text{Val}(\ell, t) d\mu(\ell, t)$.

Step 1: Symbolic Function Analysis to define the exact mode policy

The structure of Algorithm 1 is as follows: from initial value $\text{Val}_0 = 0$, we inductively define the next action value using Bellman Optimality (line 7). Then, the modes of this new value function are defined (line 9) and for each mode the best action is chosen (line 13). Finally the new value function is *defined* for each mode (line 14). Notice that we only need to *determine* the boundaries where the sign of $\text{Val}_i(\ell, x, \alpha) - \text{Val}_i(\ell, x, \beta)$ changes, we do not need to *compute* or *represent* the value function (*symbolic* nature). The algorithm relies on the crucial fact that the number of such boundaries at each step is finite (Lemma 18).

► **Lemma 18.** *Let $\ell \in \mathcal{L}$ a control state. For all $i \in \mathbb{N}$, defining $f : x \mapsto \text{Val}_i(\ell, x, \alpha) - \text{Val}_i(\ell, x, \beta)$, there is a $m \in \mathbb{N}$ such that for all $j \leq m$, there exists a sequence $(\tilde{f}^{(j)})_{j=0}^m$ of derivable functions, called the pseudo-derivatives of f , with:*

- *the sign of $\tilde{f}^{(0)}$ and of f coincide in all points of M ,*
- *for $1 \leq j \leq m$, the sign of $\tilde{f}^{(j)}$ and of $\frac{d\tilde{f}^{(j-1)}}{dt}$ coincide in all points of M , and*
- *function $\tilde{f}^{(m)}$ has a constant sign over M .*

In particular, functions $f, \tilde{f}^{(0)}$ change sign at most m times.

■ **Algorithm 1** Symbolic Value Iteration Algorithm to define the exact mode policy σ_i^* .

```

1 Init:  $i := 0$  and  $\forall \ell \in \mathcal{L}, x \in \mathbb{R}_{\geq 0}, \iota \in \{\alpha, \beta\}$ , let  $\text{Val}_0^*(\ell, x) := \text{Val}_0^*(\ell, x, \iota) = 0$ ;
2  $\sigma_0^*$  has only one mode  $M = [0, \infty)$  for each  $\ell \in \mathcal{L}$  playing  $\sigma_0^{*\ell}(M) = \alpha$ ;
3 repeat
4   for each location  $\ell \in \mathcal{L}$  do
5     for each mode  $M \in \mathcal{M}_{\sigma_i}^\ell$  do
6       for each action  $\iota \in \{\alpha, \beta\}$ , let  $\ell'$  the destination of the transition  $\iota$  and
           $r = 1$  if the clock is reset,  $r = 0$  otherwise do
7         Define  $\text{Val}_{i+1}(\ell, x, \iota) := \mathcal{R}(\iota) + \gamma \text{IVal}_i^*(\ell', (1-r)x)$  for any  $x$ ;
8       end
9       Compute a minimal finite set  $\mathcal{M}_M^\ell$  of modes such that on each mode
           $\text{Val}_{i+1}(\ell, v, \alpha) - \text{Val}_{i+1}(\ell, v, \beta)$  does not change sign;
10      end
11      Set  $\mathcal{M}_{i+1}^\ell := \bigcup_{M \in \mathcal{M}_{\sigma_i}^\ell} \mathcal{M}_M^\ell$ 
12    end
13    Assign  $\sigma_{i+1}^* := (\mathcal{M}_{i+1}^\ell, \sigma_{i+1}^{\ell})_{\ell \in \mathcal{L}}$  where for each control state  $\ell \in \mathcal{L}$  and mode
           $M \in \mathcal{M}_{i+1}$ , the policy is  $\sigma_{i+1}^{*\ell}(M) = \alpha$  if  $\text{Val}_{i+1}(\ell, x, \alpha) - \text{Val}_{i+1}(\ell, x, \beta) \geq 0$ 
          for  $x \in M$ , and  $\sigma_{i+1}^{*\ell}(M) = \beta$  otherwise;
14    Define  $\text{Val}_{i+1}^*(\ell, v) := \text{Val}_{i+1}(\ell, v, \sigma_{i+1}^{\ell}(M))$  for  $v \in M$ ;
15    Let  $i := i + 1$ ;
16 end

```

Line 13 computes the sign of $\text{Val}_{i+1}(\ell, x, \alpha) - \text{Val}_{i+1}(\ell, x, \beta) = f(x)$. As f changes sign at most m times (Lemma 18), we have at most $m + 1$ modes. The proof of Lemma 18 can be found in Section 5.4 and relies on the symbolic derivation of $\tilde{f}^{(j)}$ depending on the probability distributions $d\mu(\ell, t)$. We can now prove that Algorithm 1 produces an ε -policy:

► **Proposition 19.** *For all $\varepsilon > 0$, Algorithm 1 produces a mode-policy σ_ε^* that is ε -optimal. Further, one can effectively compute a step i after which $(\text{Val}^* - \varepsilon) < \text{Val}_{\sigma_i^*} \leq \text{Val}^*$.*

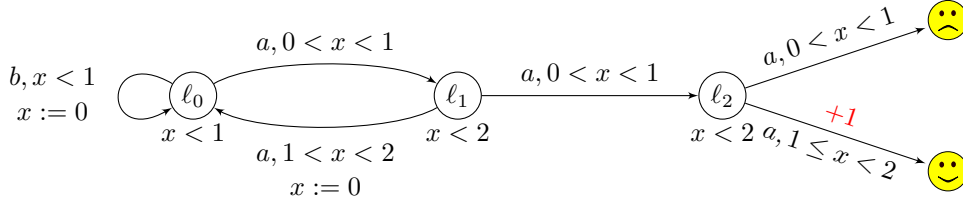
The proof of Proposition 19 can be found in Section 5.5. It is done in three steps: first show that Val_i^* converges, towards a value that is solution of the Bellman Optimality Equation. The third part of the proof provides the number of steps i to ensure a given ε .

To prove Theorem 8, we make symbolic Algorithm 1 *effective*, computing a mode policy $\overline{\sigma}_i^*$ approximating σ_i^* . In particular, we effectively approximate constants coming from integral computations in $(\tilde{f}^{(j)})_{j=0}^m$, then approximate their roots, to obtain approximated roots of f , giving \mathbb{Q} -boundaries arbitrarily close to the boundaries of modes of the symbolic σ_i^* .

Step 2: Numerical Analysis to effectively compute an approximated mode policy $\overline{\sigma}_i^*$

Line 13 of Algorithm 1 asks to find the change of sign (or equivalently the roots) for the function f defined in Lemma 18. While the values of the roots may not be easily computable symbolically, we show in the following Lemma that we can approximate them with arbitrarily high precision using numerical analysis (e.g. using Ridder's method [17]):

► **Lemma 20.** *Let f and m be as defined in Lemma 18. Then for all $\varepsilon_0 > 0$, one can compute $b_0 \leq a_1 \leq b_1 \leq \dots \leq a_m$ such that for all $j \leq m$, f does not change sign on $[b_j, a_{j+1})$, and $|f(t)| \leq \varepsilon_1 = 5\varepsilon_0 \text{MaxValue}$ over all $t \in [a_j, b_j]$ and $|b_j - a_j| \leq \varepsilon_0$.*



The proof of Lemma 20 can be found in [13]. It is crucial that the number of intervals m does not depend upon the bounds ε_0 to obtain an effectively computable mode policy. From a value function Val , associated function f , and $b_0 \leq a_1 \leq b_1 \leq \dots \leq a_m$ from Lemma 20 using a small enough $\varepsilon_0 > 0$, we define the approximated mode policy $\bar{\sigma}$ as follows:

- Between (b_i, b_{i+1}) , the policy $\bar{\sigma}$ plays α if the sign of $f(\frac{b_i+a_{i+1}}{2})$ is > 0 . Indeed, between (b_i, a_{i+1}) , we know that playing α brings a higher Val_{i+1} than playing β . Between (a_{i+1}, b_{i+1}) , the difference is small so it does not matter much the choice α or β .
- Otherwise, $\bar{\sigma}$ plays β .

We denote $\bar{\text{Val}}(\ell, v) = \mathcal{R}(\bar{\sigma}(\ell, v)) + \gamma \text{IVal}(\ell', v_{r \leftarrow 0})$ the value function associated with $\bar{\sigma}$, for ℓ' reached by playing $\bar{\sigma}$ resetting r . Let Val_{+1} be the value function we would obtain by applying one iteration of Algorithm 1 from Val . As corollary of Lemma 20, we obtain:

► **Corollary 21.** $|\bar{\text{Val}}(\ell, x) - \text{Val}_{+1}(\ell, x)| < \epsilon_1 = 5\varepsilon_0 \text{MaxValue}$.


We inductively compute $\bar{\text{Val}}_i, \bar{\sigma}_i$ from the previously computed $\bar{\text{Val}}_{i-1}, \bar{\sigma}_{i-1}$.

The proof of Theorem 8, provided in Section 5.6, shows that we can limit the effect of those successive approximations on the value of the resulting policy. We obtain $\bar{\text{Val}}_i > \text{Val}_i - \frac{\varepsilon}{2}$ by applying Corollary 21 repetitively for ε_0 chosen small enough. We conclude by combining this error for the i provided by Proposition 19, and using the fact that $\text{Val}(\bar{\sigma}_i) \geq \bar{\text{Val}}_i$.

5.3 Running example: methodology and numerical evaluation

We now roll out our procedure on an example, and compare with discretization [9].

Let us consider the DSTA \mathcal{A} from Figure 1. For convenience, we copy the picture hereafter. At initialization, there is a unique mode $[0, \infty)$ in all control state, choosing action a .

After one step, only ℓ_2 has a non-null value. The value of ℓ_2 is $\text{Val}_1(\ell_2, x) = \text{Val}^*(\ell_2, x) = 1$ if $x > 1$ and 0 if $x < 1$, as the only transition with non-null reward is the one from ℓ_2 to .

The value remains unchanged by later iterations. As there is no choice of action (only a available), ℓ_2 has a unique mode choosing a . It is not always possible to compute a closed-form solution for the value, and often one needs to keep the integral expression.

After two steps, the value in ℓ_1 is updated as $\text{Val}_1(\ell_2, x) > 0$ for some x . The value is $\text{Val}_2(\ell_1, x) = \text{Val}_3(\ell_1, x) = 0$ for $x > 1$ and $\frac{\gamma}{2-x} \int_{t \in [x, 1)} 1 dt = \gamma \frac{1-x}{2-x}$ for $x < 1$, where again we can give a closed form solution. This will be updated at later steps (≥ 4) after ℓ_0 gets a non-zero value and transition from ℓ_1 to ℓ_0 can bring some contribution to the reward as well. As there is no choice of action in ℓ_1 , there is a unique mode, choosing action a .

After three steps, the values of actions in ℓ_0 are updated for the first time. We have $\text{Val}_3(\ell_0, x, b) = 0$ and $\text{Val}_3(\ell_0, x, a) = \frac{\gamma}{2-x} \int_{t \in [x, 1)} \text{Val}_2(\ell_1, t) dt = \frac{\gamma^2}{2-x} \int_{t \in [x, 1)} \frac{1-t}{2-t} dt > 0$. We can notice that $\frac{1-t}{2-t} = 1 - \frac{1}{2-t}$. As $\int_{t \in [x, 1)} \frac{1}{2-t} dt = \int_{t \in (1, 2-x]} \frac{1}{t} dt = \log(2-x)$, this gives $\text{Val}_3(\ell_0, x, a) = \gamma^2 \frac{1-x-\log(2-x)}{2-x}$. So there is a unique mode, $[0, \infty)$ choosing action a .

13:12 Reinforcement Planning for ε -Optimal Policies in Dense Time

The fourth steps for ℓ_0 is the first interesting case. To take its decision, value iteration considers $\text{Val}_3(\ell_0, x) = \gamma^2 \frac{1-x-\log(2-x)}{2-x}$ and $\text{Val}_3(\ell_1, x) = \gamma \frac{1-x}{2-x}$. We have:

$$\text{Val}_4(\ell_0, x, b) = \gamma^3 \int_{t \in [0,1)} \frac{1-t-\log(2-t)}{2-t} dt; \quad \text{Val}_4(\ell_0, x, a) = \frac{\gamma^2}{1-x} \int_{t \in [x,1)} \frac{1-t}{2-t} dt.$$

To know whether one needs to choose action a or b in ℓ_0 at the 4th step of value iteration, one has to study the sign of $f = \text{Val}_4(\ell_0, x, b) - \text{Val}_4(\ell_0, x, a)$. When it is positive, the policy should play b , and a otherwise. The value $\text{Val}_4(\ell_0, x, b)$ is a constant C_1 not depending upon x , whose value can be approximated to $0.0666\gamma^3$. We *do not compute* a closed form solution for $\text{Val}_4(\ell_0, x, a)$ - this is not necessary and often not possible anyway.

We have $f(x) = C_1 - \frac{\gamma^2}{1-x} \int_{t \in [x,1)} \frac{1-t}{2-t} dt$. We multiply f by $(1-x)$ which does not change its sign (as $x < 1$), defining $\tilde{f}^{(0)}(x) = f(x)(1-x)$. We compute $\frac{d\tilde{f}^{(0)}(x)}{dx} = \gamma^2 \frac{1-x}{2-x} - C_1$, getting rid of the integral. Multiplying by $(2-x)$, which is always positive, we define $\tilde{f}^{(1)}(x) = \gamma^2(1-x) - C_1(2-x)$. This is an affine function, which changes sign at most once, and whose root can be easily decided, depending on the fix value of γ . The function $\tilde{f}^{(1)}$ cancels at the same root, hence $\tilde{f}^{(0)}$, changes at most twice its sign, that is $f(x)$ changes at most twice its sign (3 modes maximum) at the same (at most 2) roots. Actually, for $\gamma = 0.95$, it has only 1 root r in $[0,1)$, around 0.49863, so 2 modes:

$$M_4^{\ell_0}: \quad \begin{array}{ccc} & b & a \\ & \text{---} & \text{---} \\ 0 & r \approx 0.49863 & 1 \end{array} \rightarrow$$

We finish the example by explaining how modes are taken into account inductively:

$$\text{Val}_5(\ell_0, x, b) = \gamma \int_{t \in (0,r)} \text{Val}_4(\ell_0, t, b) dt + \gamma \int_{t \in (r,1)} \text{Val}_4(\ell_0, t, a) dt,$$

while $\text{Val}_5(\ell_0, x, a) = \text{Val}_4(\ell_0, x, a)$ as $\text{Val}_4(\ell_1, x, a) = 0$ for $x > 1$. By definition, $\text{Val}_5(\ell_0, x, a)$ does not depend upon x , it is a constant C_2 larger than $C_1 = \text{Val}_4(\ell_0, x, a)$ for $\gamma = 0.95$, which gives still 2 modes for $M_5^{\ell_0}$ with a switch at $r^5 > r$. For all i , $\text{Val}_i(\ell_1, x, a)$ is a constant for $x > 1$, while $\text{Val}_i(\ell_1, x, a) = \text{Val}_2(\ell_1, x, a)$ for $x < 1$. Hence, all subsequent value iteration for ℓ_0 have two modes, playing b on $(0, r^i)$ and a on $(r^i, 1)$, with r^i approaching r^* for the optimal strategy playing b on $(0, r^*)$ and a on $(r^*, 1)$.

Lemma 18 generalizes this process proving that there is only a finite number of sign changes for f , while Lemma 20 shows that the bounds of the intervals of the different roots/modes of $\tilde{f}^{(j)}$ can be approximated arbitrarily close (using Ridder's method [17]).

Evaluation on the running example and comparison with time-discretization

We now evaluate the different constants needed on the running example. Consider a standard discount $\gamma = 0.95$. MaxReward=1 gives (proposition 19) a number of value iterations of $i < 200$ for a precision of $\varepsilon = 10^{-5}$, and of $i < 300$ for $\varepsilon = 10^{-7}$. For each iteration i , the two constants $C_1^i = \text{Val}_i(\ell_0, x, b)$ independent of the choice of $x < 1$ and $C_2^i = \text{Val}_i(\ell_1, x, a)$ independent of the choice of $x > 1$ are evaluated, i.e. 2 evaluations of function at one given point x , so overall i , 400 or 600 evaluations of function at one point. Then, the roots root_i^1 of $\tilde{f}^{(1)}$ are computing using C_1^i, C_2^i , then the root of $\tilde{f}^{(0)}$ in $(0, \text{root}_i^1)$ if any, and the root in $(\text{root}_i^1, 1)$ if any, using Ridder's method. This yields 600, 900 call to Ridder's altogether.

Each call needs respectively $\log(2 \cdot 10^5) < 13$ and $\log(2 \cdot 10^7) < 17$ iterations to ensure the $\varepsilon = \frac{1}{2}10^{-5}$ and $\varepsilon = \frac{1}{2}10^{-7}$ precision respectively. Each iteration of Ridder's method evaluates $f(x)$ at one given point x to narrow down where the root is (which side of $f(x)$ it is). Applying Corollary 21, one obtains a precision of 10^{-4} with 8200 evaluations of functions at one point, and 10^{-6} after 15900 evaluations of function at one point respectively.

Consider the time-discretization method of [9]. It cannot handle uniform distribution as used in ℓ_0, ℓ_1 , but for the sake of comparison, consider a 3 states system using only exponential distributions, and a single clock. Quoting the long technical report p22 http://www.cs.ox.ac.uk/people/taolue.chen/pub-papers/cdc_full.pdf of [9], the authors need to solve a >60000 states MDP for 10^{-4} precision, and > 1 million states MDP for 10^{-6} precision. Using Linear Programming, this would amount to the same number of variables, with a process far from linear. Using value iteration would be possible, but incur another error which would need to be taken into account, amounting to a similar 200 iterations times 60000 states for the 10^{-4} precision. An advantage is that the time-discretization also holds for more than one clock, although the complexity would be prohibitive, with an MDP with 1.2 billion states already for 2 clocks and the precision 10^{-4} , making it a moot advantage, on top of the bounds not applying for DSTAs.

5.4 Proof of Lemma 18

Notice that if we could compute a closed-form solution of the value functions after n steps, it would be easy to deduce the values of x where the sign of $f(x)$ changes. The problem is that we do not know closed-form solutions of integrals of more and more complicated functions.

Instead, we cleverly replace the integrals by a simpler functions $\tilde{f}^{(j)}$ which changes sign at the same value, derive, until we reach a polynomial where the lemma is trivially true, and analyze the roots of function $\tilde{f}^{(j)}$ by a backward induction. This does not work with ≥ 2 clocks as for functions of ≥ 2 variables, sign changes are not related with roots of derivatives.

We fix a location ℓ at iteration i . We present the proof in the case where neither playing α nor β resets x (other cases are simpler) and all distributions on delays are uniform (which is more complex than exponentials). Other cases can be found in [13]. For $\iota \in \{\alpha, \beta\}$, we denote $(M_j = (x_j^\iota, x_{j+1}^\iota))_{0 \leq j \leq J_\iota}$ the modes of σ_i in ℓ_ι that can be reached from x . We have:

$$\text{Val}_{i+1}(\ell, x, \iota) = \mathcal{R}(\iota) + \gamma \frac{1}{x_{J_\iota+1}^\iota - x} \left(\int_{x \leq t \leq x_1^\iota} \text{Val}_i(\ell_\iota, t, \sigma_i^{\ell_\iota}(M_0)) dt + \sum_{j=1}^{J_\iota} \int_{x_j^\iota \leq t \leq x_{j+1}^\iota} \text{Val}_i(\ell_\iota, t, \sigma_i^{\ell_\iota}(M_j)) dt \right).$$

Let $x_m = \min(x_0^\alpha, x_0^\beta)$. There exists constants C_ι , C'_ι and C''_ι such that:

$$\text{Val}_{i+1}(\ell, x, \iota) = C_\iota + C'_\iota \frac{1}{x_{J_\iota+1}^\iota - x} \int_{x \leq t < x_m} \text{Val}_i(\ell_\iota, t, \sigma_i^{\ell_\iota}(M_0)) dt + C''_\iota \frac{1}{x_{J_\iota+1}^\iota - x}.$$

Lemma 18 only needs symbolic functional analysis (value of m and existence of functions $(\tilde{f}^{(j)})_{j \leq m}$), so we do not need in Lemma 18 the actual value of C_ι 's, only their existence. Later, in Lemma 20, we will however need to approximate these constants. This is done through numerical integration (Newton-Cotes and Generalized Gauss-Laguerre quadrature). To study the sign of $f_{i+1}(x) = \text{Val}_{i+1}(\ell, x, \alpha) - \text{Val}_{i+1}(\ell, x, \beta)$ we study the sign of $\tilde{f}^{(0)} = (x_{\alpha+1}^\alpha - x)(x_{\beta+1}^\beta - x)f_{i+1}(x)$, which is the same. For C a constant, we write $\tilde{f}^{(0)}$ as:

$$\begin{aligned}\tilde{f}^{(0)}(x) = & C(x_{J_\alpha+1}^\alpha - x)(x_{J_\beta+1}^\beta - x) + C'_\alpha(x_{J_\beta+1}^\beta - x) \int_{x \leq t \leq x_m} \text{Val}_i(\ell_\alpha, t) dt + C_\alpha(x_{J_\beta+1}^\beta - x) \\ & + C'_\beta(x_{J_\alpha+1}^\alpha - x) \int_{x \leq t \leq x_m} \text{Val}_i(\ell_\beta, t) dt + C_\beta(x_{J_\alpha+1}^\alpha - x).\end{aligned}$$

We derive $\tilde{f}^{(0)}$ once to obtain $\tilde{f}^{(1)}$, then once more obtaining for some C_1, C_2, C_3, C_4, C_5 :

$$g(x) = C_1 + C_2 \text{Val}_i(\ell_\alpha, x) + C_3(x_{J_\beta+1}^\beta - x) \text{Val}'_i(\ell_\alpha, x) + C_4 \text{Val}_i(\ell_\beta, x) + C_5(x_{J_\alpha+1}^\alpha - x) \text{Val}'_i(\ell_\beta, x)$$

There is no more integral terms in $g^{(2)}(x)$. Function $g^{(2)}(x)$ is of the generic form:

$$G_i(x) = P(x) + \sum_{k \leq K} P_k^\alpha(x) \text{Val}_i^{(k)}(\ell_\alpha, x) + \sum_{k \leq K} P_k^\beta(x) \text{Val}_i^{(k)}(\ell_\beta, x),$$

for some polynomials $P, P_0^\alpha, \dots, P_K^\alpha, P_0^\beta, \dots, P_K^\beta$. Generalizing the method above, we can show by induction over $i \in \mathbb{N}$ [13], that there exists a $m' \in \mathbb{N}$ such that for all $j \leq m'$, there exists a sequence $(\tilde{g}^{(j)})_{j=0}^{m'}$ of derivable functions with:

- the sign of $\tilde{g}^{(0)}$ and of $G_i(x)$ coincide in all points of M ,
- for $3 \leq j \leq m'$, the sign of $\tilde{g}^{(j)}$ and of $\frac{d\tilde{g}^{(j-1)}}{dt}$ coincide in all points of M , and
- function $\tilde{g}^{(m')}$ has a constant sign over M .

Applying this result, it suffices to choose $m = m' + 2$ and $\tilde{f}^{(j)} = \tilde{g}^{(j-2)}$ for all $2 \leq j \leq m' + 2$.

5.5 Proof of Proposition 19

First, we define $\text{MaxReward} = \max_{(\ell, \iota) \in \mathcal{L} \times E} (\mathcal{R}(\iota))$, the maximal reward over all the transitions, as well as $\text{MaxValue} = \frac{\text{MaxReward}}{1-\gamma}$ an upper bound on values in all the states.

We now show that the value iteration process indeed converges towards the optimal value $\text{Val}^*(\mathcal{A})$. For this, we first demonstrate that $(\text{Val}_i)_{i \in \mathbb{N}}$ converges as it is non-decreasing and upper bounded. Then we prove that the value it converges to is a (bounded) solution to Bellman Optimality Equations, which allows us to stop the algorithm after a computable number i of steps, with the guarantee that the policy is ε -optimal.

Proof.

Value Iteration converges. Recall that $\text{Val}_0(\ell, v) = 0$ for all ℓ, v , and that all rewards are non negative. By an induction on i , we show that the Val_i have an upper bound:

$$\forall i \in \mathbb{N}, \forall (\ell, v) \in \mathcal{L} \times \mathbb{R}_{\geq 0}, \text{Val}_i(\ell, v) \leq \frac{\text{MaxReward}}{1-\gamma} = \text{MaxValue}$$

Another induction on i shows that the v_i are increasing: for all $i \geq 0$, $\text{Val}_{i+1} \geq \text{Val}_i$.

Initialization: $\text{Val}_0 \leq \text{Val}_1$ is trivial as $\text{Val}_0 = 0$ and no reward is negative.

Now, assume $\text{Val}_{J+1} \geq \text{Val}_J$. Let $(\ell, v) \in \mathcal{L} \times \mathbb{R}_{\geq 0}$.

$$\begin{aligned}\text{Val}_{J+2}(\ell, v) &= \max_{\iota \in E} \mathcal{R}(\iota) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \text{Val}_{J+1}(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})} \\ &\quad \text{applying the induction hypothesis, we get} \\ &\geq \max_{\iota \in E} \mathcal{R}(\iota) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \text{Val}_J(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})} \\ &= \text{Val}_{J+1}(\ell, v)\end{aligned}$$

Thus (Val_i) is increasing with i , bounded, hence converges towards some Val_∞ .

Val_∞ is a solution to Bellman Optimality Equation. For all $(\ell, v) \in \mathcal{L} \times \mathbb{R}_{\geq 0}$ and i :

$$\text{Val}_{i+1}(\ell, v) = \max_{\iota \in \Sigma} \mathcal{R}(\iota) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \text{Val}_i(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})}$$

At the limit, we obtain:

$$\begin{aligned} \text{Val}_{\infty}(\ell, v) &= \lim_{i \rightarrow \infty} \max_{\iota \in \Sigma} \mathcal{R}(\iota) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \text{Val}_i(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})} \\ &= \max_{\iota \in \Sigma} \mathcal{R}(\iota) + \gamma \lim_{i \rightarrow \infty} \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \text{Val}_i(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})} \\ \text{Val}_{\infty}(\ell, v) &= \max_{\iota \in \Sigma} \mathcal{R}(\iota) + \gamma \int_{\delta \in I(\ell', v_{[r \leftarrow 0]})} \text{Val}_{\infty}(\ell', v_{[r \leftarrow 0]} + \delta) d\mu_{(\ell', v_{[r \leftarrow 0]})} \end{aligned}$$

Val_∞ is indeed a bounded solution of the Bellman Optimality Equation. Under the hypothesis that the optimal value Val* exists, it is the only bounded solution of the Bellman Optimality Equation: we therefore have Val_∞ = Val*.

As the mode-policies produced by Value Iteration are greedy over the successive Val_i that converge toward Val*, their values converge toward Val*.

Expression of the approximation as a function of i . As stated, σ_i^* plays optimally according to Val_i, thus Val_{σ_i^{*} ≥ Val_i. Furthermore, by unraveling the expression of Val_i up to Val₀ = 0 and comparing it to Val_∞ (using the Bellman Optimality Equation), we obtain: Val_∞(ℓ, v) − Val_i(ℓ, v) ≤ γⁱ MaxValue. Given ε > 0, we set $i = \frac{\log(\varepsilon) - \log(1-\gamma) + \log(\text{MaxReward})}{\log(\gamma)}$, to obtain Val_{σ_i^{*} ≥ Val_i ≥ Val_∞ − ε = Val* − ε. ◀}}

5.6 Proof of Theorem 8

Let ε > 0 be a small constant. Using Proposition 19 we know that we can compute i such that Val_i* ≥ Val* − ε/2. In particular, Val_i* (λ) ≥ Val* (λ) − ε/2 (where λ is the empty run). We choose approximation factor ε₀ as follows for Corollary 21 such that Val_i(λ) ≥ Val_i* (λ) − ε/2.

We show by induction on j that Val_j ≥ Val_j* − ∑_{k=0}^{j-1} γ^k × (5ε₀ MaxValue) with ε₀ as defined in Lemma 20. For $j = 0$ we have Val₀ = Val₀. Inductive step: suppose the result for a given j . Then, by Corollary 21 and Bellman Optimality Equation:

$$\overline{\text{Val}}_{j+1}(\ell, v) \geq \max_{\iota} (\mathcal{R}(\iota) + \gamma \cdot \overline{\text{IVal}}_j(\text{loc}'_{\iota}, v_{[r_{\iota} \leftarrow 0]})) - 5\varepsilon_0 \text{MaxValue}$$

with r_{ι} the reset associated with ι in $(\ell, v + \delta)$ and ℓ'_{ι} the target location. Using the induction hypothesis we then have

$$\overline{\text{Val}}_{j+1}(\ell, v) \geq \max_{\iota} (\mathcal{R}(\iota) + \gamma \cdot \text{IVal}_j(\text{loc}'_{\iota}, v_{[r_{\iota} \leftarrow 0]})) - 5 \text{MaxValue} \left(\gamma \times \sum_{k=0}^{j-1} \gamma^k \varepsilon_0 - \varepsilon_0 \right).$$

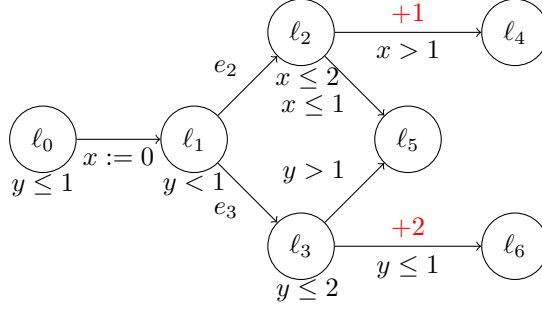
By definition of Val_{j+1} we then have our result which finished the proof by induction.

We thus fix $\frac{\varepsilon}{2} \geq \sum_{k=0}^{i-1} \gamma^k 5\varepsilon_0 \text{MaxValue}$ i.e. $\varepsilon_0 \leq \frac{\varepsilon}{10 \text{MaxValue}} \times \frac{(1-\gamma)}{(1-\gamma^i)}$, which we can effectively compute. This ends the proof of Theorem 8.

5.7 Proofs of Theorem 9 : Counter example with 2 clocks

To prove Theorem 9, we exhibit a simple DSTA with two clocks and without loops (Figure 2) in which the optimal value can not be represented using polytopes.

The only choice is between e_2 and e_3 in ℓ_1 . The optimal values in ℓ_4 , ℓ_5 and ℓ_6 are simply 0 and the optimal values in ℓ_2 and ℓ_3 can be correctly computed in one step as: Val*(ℓ₂, (x, y)) = 1 if $x > 1$ and 0 otherwise and Val*(ℓ₃, (x, y)) = 2 if $y \leq 1$ and 0 otherwise.



■ **Figure 2** A simple DSTA with two clocks without loops.

Thus the values of e_2 and e_3 are correctly computed in two steps as:

$$\text{Val}^*(\ell_1, (x, y), e_2) = \frac{1}{2-x} \int_{\delta \in]1-x, 2-x]} 1 dt = \frac{1}{2-x}$$

$$\text{Val}^*(\ell_1, (x, y), e_3) = \frac{2}{2-y} \int_{\delta \in]0, 1-y]} 1 dt = 2 \times \frac{1-y}{2-y}$$

The optimal value in ℓ_1 is thus maximum of these two values. It comes that e_2 is played when $\frac{1}{2-x} - 2 \times \frac{1-y}{2-y} > 0$ i.e. $3y + 2x - 2xy - 2 > 0$ which cannot be represented by a polytope.

6 Conclusion

Over the span of this paper, we defined a framework for Reinforcement Planning in dense time with discontinuities, modelled by DSTAs. In the presence of a single clock, we developed an algorithm that computes a ϵ -optimal policy over an uncountable number of configurations in bounded time, represented in a finite way as a mode policy. Notice that we do not use a closed-form solution for the value function itself: we believe such closed-form solutions cannot be computed in general even for 1-clock DSTA, as some integrals do not have closed-form solutions. Further, we did not rely on a discretization of time in ϵ -steps, in order to keep the number of mode minimal which we showed to be efficient on our running example w.r.t. time-discretization. This made the proofs particularly involved.

An interesting question is what happens if we use policy iteration instead of value iteration. This technique considers the reward associated with a policy, and thus demands more complex computations than value iteration which only considers bounded paths. We do not know how to evaluate them as we do not have closed-form solution of the value function.

Last, we supposed that we knew all the parameters of our model (rewards, distributions on delays) as it is the case in RP. In untimed settings, Reinforcement Learning developed techniques to work on partially unknown models by exploring them while trying to limit the necessary sample size. A interesting extension of this work would be to study a possible adaptation of exploration algorithms in dense time, which however reveals to be very challenging when dealing with Timed automata and its extensions [11, 14].

References

- 1 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994. doi:10.1007/BFb0031987.
- 2 Christel Baier, Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Marcus Größer. Probabilistic and topological semantics for timed automata. In *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science: 27th International Conference, New Delhi, India, December 12–14, 2007. Proceedings*, pages 179–191, Berlin, Heidelberg, 2023. Springer-Verlag. doi:10.1007/978-3-540-77050-3_15.
- 3 Christel Baier, Holger Hermanns, Joost-Pieter Katoen, and Boudewijn R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time markov decision processes. *Theoretical Computer Science*, 345(1):2–26, 2005. Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004). doi:10.1016/j.tcs.2005.07.022.
- 4 Nathalie Bertrand, Thomas Brihaye, and Blaise Genest. Deciding the value 1 problem for reachability in 1-clock decision stochastic timed automata. In Gethin Norman and William Sanders, editors, *Quantitative Evaluation of Systems*, pages 313–328. Springer International Publishing, 2014. doi:10.1007/978-3-319-10696-0_25.
- 5 Nathalie Bertrand and Sven Schewe. Playing optimally on timed automata with random delays. In Marcin Jurdziński and Dejan Ničković, editors, *Formal Modeling and Analysis of Timed Systems*, pages 43–58, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi:10.1007/978-3-642-33365-1_5.
- 6 Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, 3rd edition, 2007.
- 7 Steven Bradtko and Michael Duff. Reinforcement learning methods for continuous-time markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL: https://proceedings.neurips.cc/paper_files/paper/1994/file/07871915a8107172b3b5dc15a6574ad3-Paper.pdf.
- 8 Tomas Brazdil, Vojtech Forejt, Jan Krcal, Jan Kretinsky, and Antonin Kucera. Continuous-Time Stochastic Games with Time-Bounded Reachability. In Ravi Kannan and K. Narayan Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 4 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 61–72, Dagstuhl, Germany, 2009. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.FSTTCS.2009.2307.
- 9 Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Reachability probabilities in markovian timed automata. In *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC’11)*, pages 7075–7080, 2011. doi:10.1109/CDC.2011.6160992.
- 10 Kenji Doya. Reinforcement Learning in Continuous Time and Space. *Neural Computation*, 12(1):219–245, January 2000. doi:10.1162/089976600300015961.
- 11 Olga Grinchtein. *Learning of Timed Systems*. PhD thesis, Uppsala University, Sweden, 2008. URL: <http://nbn-resolving.de/urn:nbn:se:uu:diva-8763>.
- 12 Xianping Guo and On’esimo Hernandez-Lerma. Continuous-time markov decision processes. *Theory and Applications*, 2009. doi:10.1007/978-3-642-02547-1.
- 13 Léo Henry, Blaise Genest, and Alexandre Drewery. Reinforcement planning for effective ε -optimal policies in dense time with discontinuities. Technical report, CNRS, 2023. URL: <http://perso.crans.org/genest/HGD23.pdf>.
- 14 Léo Henry, Thierry Jéron, and Nicolas Markey. Active learning of timed automata with unobservable resets. In Nathalie Bertrand and Nils Jansen, editors, *18th International Conferences on Formal Modelling and Analysis of Timed Systems (FORMATS’20)*, volume 12288 of *Lecture Notes in Computer Science*, pages 144–160. Springer, September 2020. doi:10.1007/978-3-030-57628-8_9.

- 15 Rupak Majumdar, Mahmoud Salamati, and Sadegh Soudjani. On Decidability of Time-Bounded Reachability in CTMDPs. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 133:1–133:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2020.133.
- 16 Markus N. Rabe and Sven Schewe. Optimal time-abstract schedulers for ctmdps and continuous-time markov games. *Theoretical Computer Science*, 467:53–67, 2013. doi:10.1016/j.tcs.2012.10.001.
- 17 C. Ridders. A new algorithm for computing a single root of a real continuous function. *IEEE Transactions on Circuits and Systems*, 26(11):979–980, 1979. doi:10.1109/TCS.1979.1084580.
- 18 Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.
- 19 Çagatay Yildiz, Markus Heinonen, and Harri Lähdesmäki. Continuous-time model-based reinforcement learning. In *International Conference on Machine Learning*, 2021. URL: <https://api.semanticscholar.org/CorpusID:231855323>.