Non-Stationary Markov Decision Processes a Worst-Case Approach using Model-Based Reinforcement Learning

Erwan Lecarpentier, Emmanuel Rachelson

Non-Stationary Markov Decision Processes a Worst-Case Approach using Model-Based Reinforcement Learning

Erwan Lecarpentier
Université de Toulouse
ONERA - The French Aerospace Lab
erwan.lecarpentier@isae-supaero.fr

Emmanuel Rachelson
Université de Toulouse
ISAE-SUPAERO
emmanuel.rachelson@isae-supaero.fr

Abstract

This work tackles the problem of robust zero-shot planning in non-stationary stochastic environments. We study Markov Decision Processes (MDPs) evolving over time and consider Model-Based Reinforcement Learning algorithms in this setting. We make two hypotheses: 1) the environment evolves continuously with a bounded evolution rate; 2) a current model is known at each decision epoch but not its evolution. Our contribution can be presented in four points. 1) we define a specific class of MDPs that we call Non-Stationary MDPs (NSMDPs). We introduce the notion of regular evolution by making an hypothesis of Lipschitz-Continuity on the transition and reward functions w.r.t. time; 2) we consider a planning agent using the current model of the environment but unaware of its future evolution. This leads us to consider a worst-case method where the environment is seen as an adversarial agent; 3) following this approach, we propose the Risk-Averse Tree-Search (RATS) algorithm, a zero-shot Model-Based method similar to Minimax search; 4) we illustrate the benefits brought by RATS empirically and compare its performance with reference Model-Based algorithms.

1 Introduction

One of the hot topics of modern Artificial Intelligence (AI) is the ability for an agent to adapt its behaviour to changing tasks. In the literature, this problem is often linked to the setting of Lifelong Reinforcement Learning (LRL) [Silver et al., 2013, Abel et al., 2018a,b] and learning in non-stationary environments [Choi et al., 1999, Jaulmes et al., 2005, Hadoux, 2015]. In LRL, the tasks presented to the agent change sequentially at discrete transition epochs [Silver et al., 2013]. Similarly, the non-stationary environments considered in the literature often evolve abruptly [Hadoux, 2015, Hadoux et al., 2014, Doya et al., 2002, Da Silva et al., 2006, Choi et al., 1999, 2000, 2001, Campo et al., 1991, Wiering, 2001]. In this paper, we investigate environments continuously changing over time that we call Non-Stationary Markov Decision Processes (NSMDPs). In this setting, it is realistic to bound the evolution rate of the environment using a Lipschitz Continuity (LC) assumption.

Model-based Reinforcement Learning approaches [Sutton et al., 1998] benefit from the knowledge of a model allowing them to reach impressive performances, as demonstrated by the Monte Carlo Tree Search (MCTS) algorithm [Silver et al., 2016]. In this matter, the necessity to have access to a model is a great concern of AI [Asadi et al., 2018, Jaulmes et al., 2005, Doya et al., 2002, Da Silva et al., 2006]. In the context of NSMDPs, we assume that an agent is provided with a *snapshot* model when its action is computed. By this, we mean that it only has access to the current model of the environment but not its future evolution, as if it took a photograph but would be unable to predict how it is going to evolve. This hypothesis is realistic, because many environments have a tractable state while their future evolution is hard to predict [Da Silva et al., 2006, Wiering, 2001]. In order to solve

LC-NSMDPs, we propose a method that considers the worst-case possible evolution of the model and performs planning w.r.t. this model. This is equivalent to considering Nature as an adversarial agent. The paper is organized as follows: first we describe the NSMDP setting and the regularity assumption (Section 2); then we outline related works (Section 3); follows the explanation of the worst-case approach proposed in this paper (Section 4); then we describe an algorithm reflecting this approach (Section 5); finally we illustrate its behaviour empirically (Section 6).

2 Non-Stationary Markov Decision Processes

To define a Non-Stationary Markov Decision Process (NSMDP), we revert to the initial MDP model introduced by Puterman [2014], where the transition and reward functions depend on time.

Definition 1. ***NSMDP.*** *An NSMDP is an MDP whose transition and reward functions depend on the decision epoch. It is defined by a 5-tuple* $\{\mathcal{S}, \mathcal{T}, \mathcal{A}, (p_t)_{t\in\mathcal{T}}, (r_t)_{t\in\mathcal{T}}\}$ *where* $\mathcal{S}$ *is a state space;* $\mathcal{T} \equiv \{1, 2, \dots, N\}$ *is the set of decision epochs with* $N \leq +\infty$*;* $\mathcal{A}$ *is an action space;* $p_t(s' \mid s, a)$ *is the probability of reaching state* s' *while performing action* a *at decision epoch* t *in state* s*;* $r_t(s, a, s')$ *is the scalar reward associated to the transition from* s *to* s' *with action* a *at decision epoch* t*.*

This definition can be viewed as that of a stationary MDP whose state space has been enhanced with time. While this addition is trivial in episodic tasks where an agent is given the opportunity to interact several times with the same MDP, it is different when the experience is unique. Indeed, no exploration is allowed along the temporal axis. Within a stationary, infinite-horizon MDP with a discounted criterion, it is proven that there exists a Markovian deterministic stationary policy [Puterman, 2014]. It is not the case within NSMDPs where the optimal policy is non-stationary in the most general case. Additionally, we define the expected reward received when taking action a at state s and decision epoch t as $R_t(s, a) = \mathbb{E}_{s'\sim p_t(\cdot|s,a)}\left[r_t(s, a, s')\right]$. Without loss of generality, we assume the reward function to be bounded between -1 and 1. In this paper, we consider discrete time decision processes with constant transition durations, which imply deterministic decision times in Definition 1. This assumption is mild since many discrete time sequential decision problems follow that assumption. A non-stationary policy π is a sequence of *decision rules* π_t which map states to actions (or distributions over actions). For a stochastic non-stationary policy $\pi_t(a \mid s)$, the value of a state s at decision epoch t within an infinite horizon NSMDP is defined, with $\gamma \in [0, 1)$ a discount factor, by:

$$V_t^\pi(s) = \mathbb{E}\left[\sum_{i=t}^{\infty} \gamma^{i-t} R_i(s_i, a_i) \,\middle|\, s_t = s,\ a_i \sim \pi_i(\cdot \mid s_i),\ s_{i+1} \sim p_i(\cdot \mid s_i, a_i) \right],$$

The definition of the state-action value function Q_t^π for π at decision epoch t is straightforward:

$$Q_t^\pi(s, a) = R_t(s, a) + \gamma \mathop{\mathbb{E}}_{s'\sim p_t(\cdot|s,a)}\left[V_{t+1}^\pi(s')\right].$$

Overall, we defined an NSMDP as an MDP where we stress out the distinction between state, time, and decision epoch due to the inability for an agent to explore the temporal axis at will. This distinction is particularly relevant for non-episodic tasks, i.e. when there is no possibility to re-experience the same MDP starting from a prior date.

The regularity hypothesis. Many real-world problems can be modelled as an NSMDP. For instance, the problem of path planning for a glider immersed in a non-stationary atmosphere [Chung et al., 2015, Lecarpentier et al., 2017], or that of vehicle routing in dynamic traffic congestion. Realistically, we consider that the expected reward and transition functions do not evolve arbitrarily fast over time. Conversely, if such an assumption was not made, a chaotic evolution of the NSMDP would be allowed which is both unrealistic and hard to solve. Hence, we assume that changes occur slowly over time. Mathematically, we formalize this hypothesis by bounding the evolution rate of the transition and expected reward functions, using the notion of Lipschitz Continuity (LC).

Definition 2. ***Lipschitz Continuity.*** *Let* (X, d_X) *and* (Y, d_Y) *be two metric spaces and* $f : X \to Y$*,* f *is L-Lipschitz Continuous (L-LC) with* $L \in \mathbb{R}^+$ *iff* $d_Y(f(x), f(\hat{x})) \leq L\, d_X(x, \hat{x}), \forall (x, \hat{x}) \in X^2$*.* L *is called a Lipschitz constant of the function* f*.*

We apply this hypothesis to the transition and reward functions of an NSMDP so that those functions are LC w.r.t. time. For the transition function, this leads to the consideration of a metric between probability density functions. For that purpose, we use the 1-Wasserstein distance [Villani, 2008].

Definition 3. ***1-Wasserstein distance.*** *Let (X, d_X) be a Polish metric space, μ, ν any probability measures on X, $\Pi(\mu, \nu)$ the set of joint distributions on $X \times X$ with marginals μ and ν. The 1-Wasserstein distance between μ and ν is $W_1(\mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{X \times X} d_X(x, y) d\pi(x, y)$.*

The choice of the Wasserstein distance is motivated by the fact that it quantifies the distance between two distributions in a physical manner, respectful of the topology of the measured space [Dabney et al., 2018, Asadi et al., 2018]. First, it is sensitive to the difference between the supports of the distributions. Comparatively, the Kullback-Leibler divergence between distributions with disjoint supports is infinite. Secondly, if one consider two regions of the support where two distributions differ, the Wasserstein distance is sensitive to the distance between the elements of those regions. Comparatively, the total-variation metric is the same regardless of this distance.

Definition 4. ***(L_p, L_r)-LC-NSMDP.*** *An (L_p, L_r)-LC-NSMDP is an NSMDP whose transition and reward functions are respectively L_p-LC and L_r-LC w.r.t. time, i.e., $\forall (t, \hat{t}, s, s', a) \in \mathcal{T}^2 \times \mathcal{S}^2 \times \mathcal{A}$,*

$$W_1(p_t(\cdot \mid s, a), p_{\hat{t}}(\cdot \mid s, a)) \leq L_p |t - \hat{t}| \quad \textit{and} \quad |r_t(s, a, s') - r_{\hat{t}}(s, a, s')| \leq L_r |t - \hat{t}|.$$

One should remark that the LC property should be defined with respect to actual decision times and not decision epoch indexes for the sake of realism. In the present case, both have the same value, and we choose to keep this convention for clarity. Our results however extend easily to the case where indexes and times do not coincide. From now on, we consider (L_p, L_r)-LC-NSMDPs, making Lipschitz Continuity our regularity property. Notice that R is defined as a convex combination of r by the probability measure p. As a result, the notion of Lipschitz Continuity of R is strongly related to that of r and p as showed by Property 1. All the proofs of the paper can be found in the Appendix.

Property 1. *Given an (L_p, L_r)-LC-NSMDP, the expected reward function $R_t : s, a \mapsto \mathbb{E}_{s' \sim p_t(\cdot|s,a)} \{r_t(s, a, s')\}$ is L_R-LC with $L_R = L_r + L_p$.*

This result shows R's evolution rate is conditioned by the evolution rates of r and p. It allows to work either with the reward function r or its expectation R, benefiting from the same LC property.

3 Related work

A close work to our approach was done by Iyengar [2005], extending Dynamic Programming (DP, Bellman [1957]) to the search of an optimal robust policy given sets of possible transition functions. It differs from our work in two fundamental aspects: 1) we consider uncertainty in the reward model as well; 2) we use a stronger Lipschitz formulation on the set of possible transition and reward functions, this last point being motivated by its relevance to the non-stationary setting. Further, we propose an online tree-search algorithm, differing from DP in terms of applicability. Szita et al. [2002] proposed a setting where the transition function of an MDP is allowed to change between decision epochs. Similarly to our Lipschitz hypothesis in the Wasserstein metric, they control the total variation distance of subsequent transition functions by a scalar value. Those slowly changing environments allow model-free RL algorithms such as Q-Learning to find near optimal policies. Conversely, Even-Dar et al. [2009] studied the case of non-stationary reward functions with fixed transition models. No assumption is made on the possible reward functions and they propose an algorithm achieving sub-linear regret with respect to the best stationary policy. Dick et al. [2014] viewed a similar setting from the perspective of online linear optimization. Finally, Csáji and Monostori [2008] studied the case of both varying reward and transition functions within a neighbourhood of a reward-transition function pair. They study the convergence of general stochastic iterative algorithms classical RL algorithms such as asynchronous DP, Q-learning and temporal difference learning.

Non-stationary MDPs have been extensively studied. A very common framework is probably that of HM-MDPs (Hidden Mode MDPs) introduced in [Choi et al., 1999]. This is a special class of Partially Observable MDPs (POMDPs [Kaelbling et al., 1998]) where a hidden mode indexes a latent stationary MDP within which the agent evolves. This way, similarly to the context of LRL, the agent experiences a series of different MDPs over time. In this setting, Choi et al. [1999, 2000] proposed methods to learn the different models of the latent stationary MDPs. Doya et al. [2002] built a modular architecture switching between models and policies when a change is detected. Similarly, Wiering [2001], Da Silva et al. [2006], Hadoux et al. [2014] proposed a method tracking the switching occurrence and re-planning if needed. Overall, as in LRL, the HM-MDP setting considers abrupt evolution of the transition and reward functions whereas we consider a continuous one. Other

settings have been considered, as by Jaulmes et al. [2005], who do not make particular hypothesis on the evolution of the NSMDP.They build a learning algorithm for POMDPs solving where time dependency is taken into account by weighting recently experienced transitions more than older ones.

To plan efficiently within an NSMDP, our approach consists in taking advantage of the *slow* LC evolution of the environment in order to plan according to the worst-case. Generally, taking advantage of Lipschitz continuity to infer bounds on the value of a function within a certain neighbourhood is a widely used tool in the RL, bandit and optimization communities [Kleinberg et al., 2008, Rachelson and Lagoudakis, 2010, Pirotta et al., 2015, Pazis and Parr, 2013, Munos, 2014]. We implement this approach with a Minimax-like algorithm [Fudenberg and Tirole, 1991], where the environment is seen as an adversarial agent, which, to the best of our knowledge, is a novel perspective.

4 Worst-case approach

We consider finding an optimal policy within an LC-NSMDP under the non-episodic task hypothesis. The latter prevents us from learning from previous experience data since they become outdated with time and no information samples have been collected yet for future time steps. An alternative is to use model-based RL algorithms such as MCTS. For a current state s_0, such algorithms focus on finding the optimal action a_0^* by using a generative model. This action is then undertaken and the operation repeated at the next state. However, using the true NSMDP model for this purpose is an unrealistic hypothesis, since this model is generally unknown. We assume the agent does not have access to the true NSMDP model; instead, we introduce the notion of *snapshot model*.Intuitively, the snapshot associated to time t_0 is a temporal slice of the NSMDP at t_0.

Definition 5. ***Snapshot of an NSMDP.*** *The snapshot of an NSMDP $\{\mathcal{S}, \mathcal{T}, \mathcal{A}, (p_t)_{t\in\mathcal{T}}, (r_t)_{t\in\mathcal{T}}\}$ at decision epoch t_0, denoted by MDP$_{t_0}$, is the stationary MDP defined by the 4-tuple $\{\mathcal{S}, \mathcal{A}, p_{t_0}, r_{t_0}\}$ where $p_{t_0}(s' \mid s, a)$ and $r_{t_0}(s, a, s')$ are the transition and reward functions of the NSMDP at t_0.*

Similarly to the NSMDP, this definition induces the existence of the snapshot expected reward R_{t_0} defined by $R_{t_0} : s, a \mapsto \mathbb{E}_{s'\sim p_{t_0}(\cdot|s,a)}\{r_{t_0}(s, a, s')\}$. Notice that the snapshot MDP$_{t_0}$ is stationary and coincides with the NSMDP *only* at t_0. Particularly, one can generate a trajectory $\{s_0, r_0, \cdots, s_k\}$ within an NSMDP using the sequence of snapshots $\{\text{MDP}_{t_0}, \cdots, \text{MDP}_{t_0+k-1}\}$ as a model. Overall, the hypothesis of using snapshot models amounts to considering a planning agent only able to get the current stationary model of the environment. In real-world problems, predictions often are uncertain or hard to perform e.g. in the thermal soaring problem of a glider.

We consider a generic *planning* agent at s_0, t_0, using MDP$_{t_0}$ as a model of the NSMDP. By planning, we mean conducting a look-ahead search within the possible trajectories starting from s_0, t_0 given a model of the environment. The search allows in turn to identify an optimal action w.r.t. the model. This action is then undertaken and the agent jumps to the next state where the operation is repeated. The consequence of planning with MDP$_{t_0}$ is that the estimated value of an s, t pair is the value of the optimal policy of MDP$_{t_0}$, written $V^*_{\text{MDP}_{t_0}}(s)$. The true optimal value of s at t within the NSMDP does not match this estimate because of the non-stationarity. The intuition we develop is that, given the *slow* evolution rate of the environment, for a state s seen at a future decision epoch during the search, we can predict a scope into which the transition and reward functions at s lie.

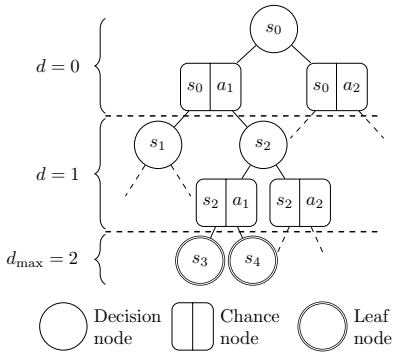
Property 2. ***Set of admissible snapshot models.*** *Consider an (L_p, L_r)-LC-NSMDP, $s, t, a \in \mathcal{S} \times \mathcal{T} \times \mathcal{A}$. The transition and expected reward functions (p_t, R_t) of the snapshot MDP$_t$ respect*

$$(p_t, R_t) \in \Delta_t \coloneqq \mathcal{B}_{W_1}\left(p_{t-1}(\cdot \mid s, a), L_p\right) \times \mathcal{B}_{|\cdot|}\left(R_{t-1}(s, a), L_R\right)$$

where $L_R = L_p + L_r$ and $\mathcal{B}_d(c, r)$ denotes the ball of centre c, defined with metric d and radius r.

For a future prediction at s, t, we consider the question of using a better model than p_{t_0}, R_{t_0}. The underlying evolution of the NSMDP being unknown, a desirable feature would be to use a model leading to a policy that is *robust* to every possible evolution. To that end, we propose to use the snapshots corresponding to the worst possible evolution scenario under the constraints of Property 2. We claim that such a practice is an efficient way to 1) ensure robust performance to all possible evolutions of the NSMDP and 2) avoid catastrophic terminal states. Practically, this boils down to using a different value estimate for s at t than $V^*_{\text{MDP}_{t_0}}(s)$ which provided no robustness guarantees.

Given a policy $\pi = (\pi_t)_{t\in\mathcal{T}}$ and a decision epoch t, a worst-case NSMDP corresponds to a sequence of transition and reward models minimizing the expected value of applying π in any pair (s, t), while



(a) Tree structure, $d_{\max} = 2$, $\mathcal{A} = \{a_1, a_2\}$.

ϵ		RATS	DP-snapshot	DP-NSMDP
0	$\mathbb{E}\left[\sum r\right]$	-0.026	**0.48**	**0.47**
	CVaR	**-0.81**	-0.90	-0.9
0.5	$\mathbb{E}\left[\sum r\right]$	**-0.032**	-0.46	**-0.077**
	CVaR	**-0.81**	-0.90	**-0.81**
1	$\mathbb{E}\left[\sum r\right]$	**0.67**	-0.78	**0.66**
	CVaR	**0.095**	-0.90	**-0.033**

(b) Expected return $\mathbb{E}\left[\sum r\right]$ and CVaR at 5%.

Figure 1: Tree structure and results from the Non-Stationary bridge experiment.

remaining within the bounds of Property 2. We write $\overline{V}_t^{\pi}(s)$ this value for s at decision epoch t.

$$\overline{V}_t^{\pi}(s) \coloneqq \min_{(p_i,R_i)\in\Delta_i,\forall i\in\mathcal{T}} \mathbb{E}\left[\sum_{i=t}^{\infty} \gamma^{i-t} R_i(s_i,a_i) \middle| \begin{array}{l} s_t = s \\ a_i \sim \pi_i(\cdot \mid s_i), s_{i+1} \sim p_i(\cdot \mid s_i, a_i) \end{array}\right] \tag{1}$$

Intuitively, the worst-case NSMDP is a model of a non-stationary environment leading to the poorest possible performance for π, while being an admissible evolution of MDP_t. Let us define $\overline{Q}_t^{\pi}(s,a)$ as the worst-case Q-value for the pair (s,a) at decision epoch t:

$$\overline{Q}_t^{\pi}(s,a) \coloneqq \min_{(p,R)\in\Delta_t} \mathop{\mathbb{E}}_{s'\sim p}\left[R(s,a) + \gamma \overline{V}_{t+1}^{\pi}(s')\right]. \tag{2}$$

5 Risk-Averse Tree-Search algorithm

The algorithm. Tree search algorithms within MDPs have been well studied and cover two classes of search trees, namely closed loop [Keller and Helmert, 2013, Kocsis and Szepesvári, 2006, Browne et al., 2012] and open loop [Bubeck and Munos, 2010, Lecarpentier et al., 2018]. Following [Keller and Helmert, 2013], we consider closed loop search trees, composed of *decision nodes* alternating with *chance nodes*. We adapt their formulation to take time into account, resulting in the following definitions. A decision node at depth t, denoted by $\nu^{s,t}$, is labelled by a unique state / decision epoch pair (s,t). The edges leading to its children chance nodes correspond to the available actions at (s,t). A chance node, denoted by $\nu^{s,t,a}$, is labelled by a state / decision epoch / action triplet (s,t,a). The edges leading to its children decision nodes correspond to the reachable state / decision epoch pairs (s',t') after performing a in (s,t) as illustrated by Figure 1a. We consider the problem of estimating the optimal action a_0^* at s_0, t_0 within a worst-case NSMDP, knowing MDP_{t_0}. This problem is twofold. It requires 1) to estimate the worst-case NSMDP given MDP_{t_0} and 2) to explore the latter in order to identify a_0^*. We propose to tackle both problems with an algorithm inspired by the minimax algorithm [Fudenberg and Tirole, 1991] where the *max* operator corresponds to the agent's policy, seeking to maximize the return; and the *min* operator corresponds to the worst-case model, seeking to minimize the return. Estimating the worst-case NSMDP requires to estimate the sequence of subsequent snapshots minimizing Equation 2. The inter-dependence of those snapshots (Equation 1) makes the problem hard to solve [Iyengar, 2005], particularly because of the combinatorial nature of the opponent's action space. Instead, we propose to solve a relaxation of this problem, by considering snapshots only constrained by MDP_{t_0}. Making this approximation leaves a possibility to violate property 2 but allows for an efficient search within the developed tree and (as will be shown experimentally) leads to robust policies. For that purpose, we define the set of admissible snapshot models w.r.t. MDP_{t_0} by $\Delta_{t_0}^t \coloneqq \mathcal{B}_{W_1}\left(p_{t_0}(\cdot \mid s,a), L_p|t-t_0|\right) \times \mathcal{B}_{|\cdot|}\left(R_{t_0}(s,a), L_R|t-t_0|\right)$. The relaxed analogues of Equations 1 and 2 for $s,t,a \in \mathcal{S}\times\mathcal{T}\times\mathcal{A}$ are defined as follows:

$$\hat{V}_{t_0,t}^{\pi}(s) \coloneqq \min_{(p_i,R_i)\in\Delta_{t_0}^i,\forall i\in\mathcal{T}} \mathbb{E}\left[\sum_{i=t}^{\infty} \gamma^{i-t} R_i(s_i,a_i) \middle| \begin{array}{l} s_t = s \\ a_i \sim \pi_i(\cdot \mid s_i), s_{i+1} \sim p_i(\cdot \mid s_i, a_i) \end{array}\right],$$

$$\hat{Q}_{t_0,t}^{\pi}(s,a) \coloneqq \min_{(p,R)\in\Delta_t} \mathop{\mathbb{E}}_{s'\sim p}\left[R(s,a) + \gamma \overline{V}_{t+1}^{\pi}(s')\right].$$

Algorithm 1: RATS algorithm

RATS (s_0, t_0, maxDepth)
ν_0 = rootNode(s_0, t_0)
Minimax(ν_0)
$\nu^* = \arg\max_{\nu' \textbf{ in } \nu.\text{children}} \nu'.\text{value}$
return ν^*.action

Minimax (ν, maxDepth)
if ν *is DecisionNode* **then**
  if ν.state *is terminal* **or** ν.depth = maxDepth **then**
    return ν.value = heuristicValue(ν.state)
  else
    return ν.value = $\max_{\nu' \in \nu.\text{children}}$Minimax($\nu'$, maxDepth)
else
  return $\nu.\text{value} = \min_{(p,R)\in\Delta^t_{t_0}} R(\nu) + \gamma \sum_{\nu' \in \nu.\text{children}} p(\nu' \mid \nu)\text{Minimax}(\nu', \text{maxDepth})$

Their optimal counterparts, while seeking to find the optimal policy, verify the following equations:

$$\hat{V}^*_{t_0,t}(s) = \max_{a\in\mathcal{A}} \hat{Q}^*_{t_0,t}(s,a), \tag{3}$$

$$\hat{Q}^*_{t_0,t}(s,a) = \min_{(p,R)\in\Delta^t_{t_0}} \mathop{\mathbb{E}}_{s'\sim p} \left[R(s,a) + \gamma \hat{V}^*_{t_0,t+1}(s') \right]. \tag{4}$$

We now provide a method to calculate those quantities within the nodes of the tree search algorithm.
Max nodes. A decision node $\nu^{s,t}$ corresponds to a max node due to the greediness of the agent w.r.t. the subsequent values of the children. We aim at maximizing the return while retaining a risk-averse behaviour. As a result, the value of $\nu^{s,t}$ follows Equation 3 and is defined as:

$$V(\nu^{s,t}) = \max_{a\in\mathcal{A}} V(\nu^{s,t,a}). \tag{5}$$

Min nodes. A chance node $\nu^{s,t,a}$ corresponds to a min node due to the use of a worst-case NSMDP as a model which minimizes the value of $\nu^{s,t,a}$ w.r.t. the reward and the subsequent values of its children. Writing the value of $\nu^{s,t,a}$ as the value of s, t, a, within the worst-case snapshot minimizing Equation 4, and using the children's values as values for the next reachable states, leads to Equation 6.

$$V(\nu^{s,t,a}) = \min_{(p,R)\in\Delta^t_{t_0}} R(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim p} V(\nu^{s',t+1}) \tag{6}$$

Our approach considers the environment as an adversarial agent, as in an *asymmetric* two-player game, in order to search for a robust plan. The resulting algorithm, RATS for Risk-Averse Tree-Search, is described in Algorithm 1. Given an initial state / decision epoch pair, a minimax tree is built using the snapshot MDP_{t_0} and the operators corresponding to Equations 5 and 6 in order to estimate the worst-case snapshots at each depth. The tree is built, the action leading to the best possible value from the root node is selected and a real transition is performed. The next state is then reached, the new snapshot model MDP_{t_0+1} is acquired and the process re-starts. Notice the use of $R(\nu)$ and $p(\nu' \mid \nu)$ in the pseudo-code: they are light notations respectively standing for $R_t(s,a)$ corresponding to a chance node $\nu \equiv \nu^{s,t,a}$ and the probability $p_t(s'|s,a)$ to jump to a decision node $\nu' \equiv \nu^{s',t+1}$ given a chance node $\nu \equiv \nu^{s,t,a}$. The tree built by RATS is entirely developed until the maximum depth $d_{\max}$. A heuristic function is used to evaluate the leaf nodes of the tree.

Analysis of RATS. We are interested in characterizing Algorithm 1 without function approximation and therefore will consider finite, countable, $\mathcal{S} \times \mathcal{A}$ sets. We now detail the computation of the min operator (Property 3), the computational complexity of RATS (Property 4) and the heuristic function.

Property 3. ***Closed-form expression of the worst case snapshot of a chance node.*** *Following Algorithm 1, a solution to Equation 6 is given by:*

$$\hat{R}(s,a) = R_{t_0}(s,a) - L_R|t-t_0| \quad \textit{and} \quad \hat{p}(\cdot \mid s,a) = (1-\lambda)p_{t_0}(\cdot \mid s,a) + \lambda p_{sat}(\cdot \mid s,a)$$

with $p_{sat}(\cdot \mid s,a) = (0,\cdots,0,1,0,\cdots,0)$ *with* 1 *at position* $\arg\min_{s'} V(\nu^{s',t+1})$, $\lambda = 1$ *if* $W_1(p_{sat}, p_0) \leq L_p|t-t_0|$ *and* $\lambda = L_p|t-t_0|\, W_1(p_{sat}, p_0)$ *otherwise.*

Property 4. ***Computational complexity.*** *The total computation complexity of Algorithm 1 is* $\mathcal{O}(B(|\mathcal{S}|^{3.5}|\mathcal{A}|^2)^{d_{\max}})$ *with* B *the number of time steps and* $d_{\max}$ *the maximum depth of the tree.*

Heuristic function. As in vanilla minimax algorithms, Algorithm 1 bootstraps the values of the leaf nodes with a heuristic function if these leaves do not correspond to terminal states. Given such a leaf node $\nu^{s,t}$, a heuristic aims at estimating the value of the optimal policy at (s,t) within the worst-case NSMDP, i.e. $\hat{V}^*_{t_0,t}(s)$. Let $H(s,t)$ be such a heuristic function, we call *heuristic error* in (s,t) the difference between $H(s,t)$ and $\hat{V}^*_{t_0,t}(s)$. Assuming that the heuristic error is uniformly bounded, the following property provides an upper bound on the propagated error due to the choice of H.

Property 5. ***Upper bound on the propagated heuristic error within RATS.*** *Consider an agent executing Algorithm 1 at* s_0, t_0 *with a heuristic function* H*. We note* $\mathcal{L}$ *the set of all leaf nodes. Suppose that the heuristic error is uniformly bounded, i.e.* $\exists \delta > 0$, $\forall \nu^{s,t} \in \mathcal{L}$, $|H(s) - \hat{V}^*_{t_0,t}(s)| \leq \delta$*. Then we have for every decision and chance nodes* $\nu^{s,t}$ *and* $\nu^{s,t,a}$*, at any depth* $d \in [0, d_{\max}]$*:*

$$|V(\nu^{s,t}) - \hat{V}^*_{t_0,t}(s)| \leq \gamma^{(d_{\max}-d)}\delta \quad \textit{and} \quad |V(\nu^{s,t,a}) - \hat{Q}^*_{t_0,t}(s,a)| \leq \gamma^{(d_{\max}-d)}\delta.$$

This last result implies that with *any* heuristic function H inducing a uniform heuristic error, the propagated error at the root of the tree is guaranteed to be upper bounded by $\gamma^{d_{\max}}\delta$. In particular, since the reward function is bounded by hypothesis, we have $\hat{V}^*_{t_0,t}(s) \leq 1/(1-\gamma)$. Thus, selecting for instance the zero function ensures a root node heuristic error of at most $\gamma^{d_{\max}}/(1-\gamma)$. In order to improve the precision of the algorithm, we propose to guide the heuristic by using a function reflecting better the value of state s at leaf node $\nu^{s,t}$. The ideal function would of course be $H(s) = \hat{V}^*_{t_0,t}(s)$, reducing the heuristic error to zero, but this is intractable. Instead, we suggest to use the value of s within the snapshot MDP_t using an evaluation policy π, i.e. $H(s) = V^{\pi}_{\text{MDP}_t}(s)$. This snapshot is also not available, but Property 6 provides a range wherein this value lies.

Property 6. ***Bounds on the snapshots values.*** *Let* $s \in \mathcal{S}$*,* π *a stationary policy,* MDP_{t_0} *and* MDP_t *two snapshot MDPs,* $t, t_0 \in \mathcal{T}^2$ *be. We note* $V^{\pi}_{MDP_i}(s)$ *the value of* s *within* MDP_i *following* π*. Then,*

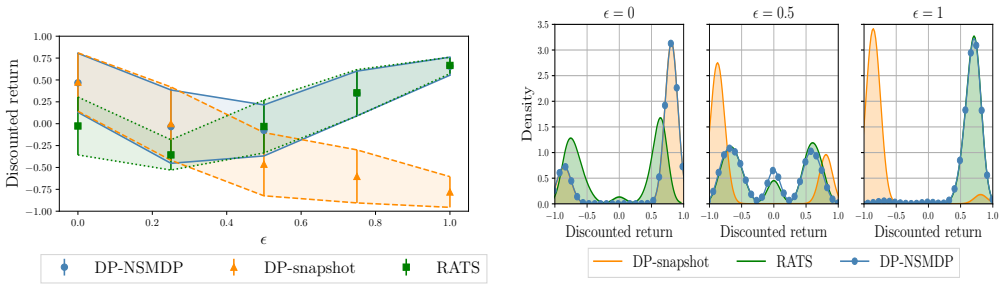
$$|V^{\pi}_{MDP_{t_0}}(s) - V^{\pi}_{MDP_t}(s)| \leq |t - t_0| L_R/(1-\gamma).$$

Since MDP_{t_0} is available, $V^{\pi}_{\text{MDP}_{t_0}}(s)$ can be estimated, e.g. via Monte-Carlo roll-outs. Let $\widehat{V}^{\pi}_{\text{MDP}_{t_0}}(s)$ denote such an estimate. Following Property 6, $V^{\pi}_{\text{MDP}_{t_0}}(s) - |t-t_0|L_R/(1-\gamma) \leq V^{\pi}_{\text{MDP}_t}(s)$. Hence, a worst-case heuristic on $V^{\pi}_{\text{MDP}_t}(s)$ is $H(s) = \widehat{V}^{\pi}_{\text{MDP}_{t_0}}(s) - |t-t_0|L_R/(1-\gamma)$. The bounds provided by Property 5 decrease quickly with $d_{\max}$, and given that $d_{\max}$ is *large enough*, RATS provides the optimal risk-averse maximizing the worst-case value for any evolution of the NSMDP.

6 Experiments

We compare the RATS algorithm with two policies [1]. The first one, named DP-snapshot, uses Dynamic Programming to compute the optimal actions w.r.t. the snapshot models at each decision epoch. The second one, named DP-NSMDP, uses the real NSMDP as a model to provide its optimal action. The latter behaves as an omniscient agent and should be seen as an upper bound on the performance. We choose a particular grid-world domain coined "Non-Stationary bridge" illustrated in Appendix, Section 7. An agent starts at the state labelled S in the centre and the goal is to reach one of the two terminal states labelled G where a reward of +1 is received. The grey cells represent holes that are terminal states where a reward of -1 is received. Reaching the goal on the right leads to the highest payoff since it is closest to the initial state and a discount factor $\gamma = 0.9$ is applied. The actions are $\mathcal{A} = \{\text{Up, Right, Down, Left}\}$. The transition function is stochastic and non-stationary. At decision epoch $t = 0$, any action deterministically yields the intuitive outcome. With time, when applying Left or Right, the probability to reach the positions usually stemming from Up and Down increases symmetrically until reaching 0.45. We set the Lipschitz constant $L_p = 1$. Aside, we introduce a parameter $\epsilon \in [0,1]$ controlling the behaviour of the environment. If $\epsilon = 0$, only the left-hand side bridge becomes slippery with time. It reflects a close to worst-case evolution for a policy aiming to the left-hand side goal. If $\epsilon = 1$, only the right-hand side bridge becomes slippery

[1] For ML reproducibility checklist informations, see Appendix Section 8.



(a) Discounted return vs ϵ, 50% of standard deviation. (b) Discounted return distributions $\epsilon \in \{0, 0.5, 1\}$.

Figure 2: Discounted return of the three algorithms for various values of ϵ.

with time. It reflects a close to worst-case evolution for a policy aiming to the right-hand side goal. In between, the misstep probability is proportionally balanced between left and right. One should note that changing ϵ from 0 to 1 does not cover all the possible evolutions from MDP_{t_0} but provides a concrete, graphical illustration of RATS's behaviour for various possible evolutions of the NSMDP.

We tested RATS with $d_{\max} = 6$ so that leaf nodes in the search tree are terminal states. Hence, the optimal risk-averse policy is applied and no heuristic approximation is made. Our goal is to demonstrate that planning in this worst-case NSMDP allows to minimize the loss given any possible evolution of the environment. To illustrate this, we report results reflecting different evolutions of the same NSMDP using the ϵ factor. It should be noted that, at $t = 0$, RATS always moves to the left, even if the goal is further, since going to the right may be risky if the probabilities to go Up and Down increase. This corresponds to the careful, risk-averse, behaviour. Conversely, DP-snapshot always moves to the right since MDP_0 does not capture this risk. As a result, the $\epsilon = 0$ case reflects a favorable evolution for DP-snapshot and a bad one for RATS. The opposite occurs with $\epsilon = 1$ where the cautious behavior dominates over the risky one, and the in-between cases mitigate this effect.

In Figure 2a, we display the achieved expected return for each algorithm as a function of ϵ, i.e. as a function of the possible evolutions of the NSMDP. As expected, the performance of DP-snapshot strongly depends on this evolution. It achieves high return for $\epsilon = 0$ and low return for $\epsilon = 1$. Conversely, the performance of RATS varies less across the different values of ϵ. The effect illustrated here is that RATS maximizes the minimal possible return given *any* evolution of the NSMDP. It provides the guarantee to achieve the best return in the worst-case. This behaviour is highly desirable when one requires robust performance guarantees as, for instance, in critical certification processes. Figure 2b displays the return distributions of the three algorithms for $\epsilon \in \{0, 0.5, 1\}$. The effect seen here is the tendency for RATS to diminish the left tail of the distribution corresponding to low returns for each evolution. It corresponds to the optimized criteria, i.e. robustly maximizing the worst-case value. A common risk measure is the Conditional Value at Risk (CVaR) defined as the expected return in the worst $q\%$ cases. We illustrate the CVaR at 5% achieved by each algorithm in Table 1b. Notice that RATS always maximizes the CVaR compared to both DP-snapshot and DP-NSMDP. Indeed, even if the latter uses the true model, the optimized criteria in DP is the expected return.

7 Conclusion

We proposed an approach for robust zero-shot planning in non-stationary stochastic environments. We introduced the framework of Lipchitz-Continuous Non-Stationary MDPs (NSMDPs) and derived the Risk-Averse Tree-Search (RATS) algorithm, to predict the worst-case evolution and to plan optimally w.r.t. this worst-case NSMDP. We analyzed RATS theoretically and showed that it approximates a worst-case NSMDP with a control parameter that is the depth of the search tree. We showed empirically the benefit of the approach that searches for the highest lower bound on the worst achievable score. RATS is robust to every possible evolution of the environment, i.e. maximizing the expected worst-case outcome on the whole set of possible NSMDPs. Our method was applied to the uncertainty on the evolution of a model. Generally, it could be extended to any uncertainty on the model used for planning, given bounds on the set of the feasible models. The purpose of this contribution is to lay a basis of worst-case analysis for robust solutions to NSMDPs. As is, RATS is computationally intensive and scaling the algorithm to larger problems is an exciting future challenge.

References

David Abel, Dilip Arumugam, Lucas Lehnert, and Michael Littman. State Abstractions for Lifelong Reinforcement Learning. In *International Conference on Machine Learning*, pages 10–19, 2018a.

David Abel, Yuu Jinnai, Sophie Yue Guo, George Konidaris, and Michael Littman. Policy and Value Transfer in Lifelong Reinforcement Learning. In *International Conference on Machine Learning*, pages 20–29, 2018b.

Kavosh Asadi, Dipendra Misra, and Michael L Littman. Lipschitz continuity in model-based reinforcement learning. *arXiv preprint arXiv:1804.07193*, 2018.

Richard Bellman. *Dynamic programming*. Princeton, USA: Princeton University Press, 1957.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Sébastien Bubeck and Rémi Munos. Open loop optimistic planning. In *10th Conference on Learning Theory*, 2010.

L. Campo, P. Mookerjee, and Y. Bar-Shalom. State estimation for systems with sojourn-time-dependent Markov model switching. *IEEE Transactions on Automatic Control*, 36(2):238–243, 1991.

Samuel P.M. Choi, Dit-yan Yeung, and Nevin L. Zhang. Hidden-mode Markov decision processes. In *IJCAI Workshop on Neural, Symbolic, and Reinforcement Methods for Sequence Learning*. Citeseer, 1999.

Samuel P.M. Choi, Dit-Yan Yeung, and Nevin L. Zhang. Hidden-mode Markov decision processes for nonstationary sequential decision making. In *Sequence Learning*, pages 264–287. Springer, 2000.

Samuel P.M. Choi, Nevin L. Zhang, and Dit-Yan Yeung. Solving hidden-mode Markov decision problems. In *Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics, Key West, Florida, USA*, 2001.

Jen Jen Chung, Nicholas R.J. Lawrance, and Salah Sukkarieh. Learning to soar: Resource-constrained exploration in reinforcement learning. *The International Journal of Robotics Research*, 34(2): 158–172, 2015.

Balázs Csanád Csáji and László Monostori. Value function based reinforcement learning in changing Markovian environments. *Journal of Machine Learning Research*, 9(Aug):1679–1709, 2008.

Bruno C. Da Silva, Eduardo W. Basso, Ana L.C. Bazzan, and Paulo M. Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 217–224. ACM, 2006.

Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Travis Dick, Andras Gyorgy, and Csaba Szepesvari. Online learning in Markov decision processes with changing cost sequences. In *International Conference on Machine Learning*, pages 512–520, 2014.

Kenji Doya, Kazuyuki Samejima, Ken-ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369, 2002.

Eyal Even-Dar, Sham M. Kakade, and Yishay Mansour. Online Markov Decision Processes. *Mathematics of Operations Research*, 34(3):726–736, 2009.

Drew Fudenberg and Jean Tirole. Game theory. *Cambridge, Massachusetts*, 393(12):80, 1991.

Emmanuel Hadoux. *Markovian sequential decision-making in non-stationary environments: application to argumentative debates*. PhD thesis, UPMC, Sorbonne Universités CNRS, 2015.

Emmanuel Hadoux, Aurélie Beynier, and Paul Weng. Sequential decision-making under non-stationary environments via sequential change-point detection. In *Learning over Multiple Contexts (LMCE)*, 2014.

Garud N. Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2): 257–280, 2005.

Robin Jaulmes, Joelle Pineau, and Doina Precup. Learning in non-stationary partially observable Markov decision processes. In *ECML Workshop on Reinforcement Learning in non-stationary environments*, volume 25, pages 26–32, 2005.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Thomas Keller and Malte Helmert. Trial-based heuristic tree search for finite horizon MDPs. In *ICAPS*, 2013.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2008.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

Erwan Lecarpentier, Sebastian Rapp, Marc Melo, and Emmanuel Rachelson. Empirical evaluation of a Q-Learning Algorithm for Model-free Autonomous Soaring. *arXiv preprint arXiv:1707.05668*, 2017.

Erwan Lecarpentier, Guillaume Infantes, Charles Lesire, and Emmanuel Rachelson. Open loop execution of tree-search algorithms. *IJCAI*, 2018.

Rémi Munos. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends® in Machine Learning*, 7(1):1–129, 2014.

Jason Pazis and Ronald Parr. PAC Optimal Exploration in Continuous Space Markov Decision Processes. In *AAAI*, 2013.

Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Policy gradient in lipschitz Markov Decision Processes. *Machine Learning*, 100(2-3):255–283, 2015.

Martin L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Emmanuel Rachelson and Michail G. Lagoudakis. On the locality of action domination in sequential decision making. 2010.

Daniel L. Silver, Qiang Yang, and Lianghao Li. Lifelong Machine Learning Systems: Beyond Learning Algorithms. In *AAAI Spring Symposium: Lifelong Machine Learning*, volume 13, page 05, 2013.

David Silver, Aja Huang, Chris .J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

Richard S. Sutton, Andrew G. Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

István Szita, Bálint Takács, and András Lörincz. ε-mdps: Learning in varying environments. *Journal of Machine Learning Research*, 3(Aug):145–174, 2002.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Marco A. Wiering. Reinforcement learning in dynamic environments using instantiated information. In *Machine Learning: Proceedings of the Eighteenth International Conference (ICML2001)*, pages 585–592, 2001.

Non-Stationary Markov Decision Processes a Worst-Case Approach using Model-Based Reinforcement Learning

Appendix

Erwan Lecarpentier
Université de Toulouse
ONERA - The French Aerospace Lab
erwan.lecarpentier@isae-supaero.fr

Emmanuel Rachelson
Université de Toulouse
ISAE-SUPAERO
emmanuel.rachelson@isae-supaero.fr

In the following proofs, the dual formulation of the 1-Wasserstein distance is used several times. We include the definition here for reference purpose.

Definition 1. ***Dual formulation of the 1-Wasserstein distance.*** *Let (X, d_X) be a Polish metric space and μ, ν any two probability measures on X. The dual formulation of the 1-Wasserstein distance between μ and ν is defined by*

$$W_1(\mu, \nu) = \sup_{f \in Lip_1} \int_X f(x) d(\mu - \nu)(x) \tag{1}$$

where Lip_1 denotes the set of the continuous mappings $X \to \mathbb{R}$ with a minimal Lipschitz constant bounded by 1*.*

1 Proof of Property 1

Consider an (L_p, L_r)-LC-NSMDP. Let $s, t, a, \hat{t} \in \mathcal{S} \times \mathcal{T} \times \mathcal{A} \times \mathcal{T}$ be. By definition of the expected reward function, the following holds:

$$\begin{aligned}
R_t(s,a) - R_{\hat{t}}(s,a) &= \int_{\mathcal{S}} \Big(p_t(s' \mid s,a) r_t(s,a,s') - p_{\hat{t}}(s' \mid s,a) r_{\hat{t}}(s,a,s') \Big) ds' \\
&= \int_{\mathcal{S}} \Big(r_t(s,a,s') \Big[p_t(s' \mid s,a) - p_{\hat{t}}(s' \mid s,a) \Big] \\
&\quad + p_{\hat{t}}(s' \mid s,a) \Big[r_t(s,a,s') - r_{\hat{t}}(s,a,s') \Big] \Big) ds' \\
&= \int_{\mathcal{S}} r_t(s,a,s') \Big[p_t(s' \mid s,a) - p_{\hat{t}}(s' \mid s,a) \Big] ds' \\
&\quad + \int_{\mathcal{S}} p_{\hat{t}}(s' \mid s,a) \Big[r_t(s,a,s') - r_{\hat{t}}(s,a,s') \Big] ds' \\
&\leq \sup_{\|f\|_L \leq 1} \int_{\mathcal{S}} f(s',t') \Big[p_t(s' \mid s,a) - p_{\hat{t}}(s' \mid s,a) \Big] ds' \\
&\quad + \int_{\mathcal{S}} p_{\hat{t}}(s' \mid s,a) L_r |t - \hat{t}| ds' \\
&\leq W_1(p(\cdot \mid s,t,a), p(\cdot \mid s,\hat{t},a)) + L_r |t - \hat{t}| \\
&\leq (L_p + L_r) |t - \hat{t}|
\end{aligned}$$

Where we used the triangle inequality, the fact that r is a bounded function and the dual formulation of the 1-Wasserstein distance (see Definition 1). The same inequality can be derived with the opposite terms which concludes the proof by taking the absolute value.

2 Proof of Property 2

Proof. The proof is straightforward using the Lipschitz property of Definition 4 and Property 1. □

3 Proof of Property 4

From Property 3, the development of a chance node is equivalent to computing a 1-Wasserstein distance which is a linear program. Following Vaidya's algorithm [Vaidya, 1989], the cost in the worst case is $\mathcal{O}(|\mathcal{S}|^{2.5})$ where $|\mathcal{S}|$ is the dimension of the problem in our case. From Equation 5, solving a decision node is equivalent to finding a maximizing action that can be done in $\mathcal{O}(|\mathcal{A}|)$ operations. One tree computation takes at most $|\mathcal{S}|^{d_{\max}}$ development of decision nodes and $|A|^{d_{\max}}$ development of chance nodes which is thus in total $\mathcal{O}((|\mathcal{S}|^{3.5}|\mathcal{A}|^2)^{d_{\max}})$. The RATS algorithm builds a tree at each time step thus B times in the total process which concludes the proof.

4 Proof of Property 3

We are looking for a closed-form expression of the value of a chance node $\nu^{s,t,a}$ as defined in Equation 6 recalled below.

$$(\bar{p}, \bar{R}) = \underset{(p,R)\in\Delta_{t_0,t}}{\arg\min} \; R(s,a) + \gamma \mathbb{E}_{s'\sim p(\cdot|s,a)} V(\nu^{s',t+1})$$

Obviously, we have that $\bar{R} = R_{t_0}(s,a) - L_R|t-t_0|$ and $\bar{p}$ is given by:

$$\bar{p} = \underset{p\in\mathcal{B}_{W_1}(p_{t_0}(\cdot|s,a), L_p|t-t_0|)}{\arg\min} \sum_{s'} p(s' \mid s,a) V(\nu^{s',t+1})$$

where $\mathcal{B}_d(c,r)$ denotes the ball of center c, defined with metric d and radius r. Since we are in the discrete case, we enumerate through the elements of $\mathcal{S}$ and write the vectors $p \equiv (p(s' \mid s,a))_{s'}$, $p_0 \equiv (p_{t_0}(s' \mid s,a))_{s'}$ and $v \equiv (V(\nu^{s',t+1}))_{s'}$. The problem can then be re-written as follows:

$$\bar{p} = \underset{p}{\arg\min} \quad p^\top v \tag{2}$$

$$\text{s.t. } p^\top \mathbf{1} = 1 \tag{3}$$

$$p \geq 0 \tag{4}$$

$$W_1(p, p_0) \leq C \tag{5}$$

Where we have $\mathbf{1} \in \mathbb{R}^{|\mathcal{S}|}$ a vector of ones, $C = L_p|t-t_0|$ and the 1-Wasserstein metric between two discrete distributions written in dual form following Lemma 1 as:

$$W_1(u,v) = \max_f f^\top (u-v) \tag{6}$$

$$\text{s.t. } Af \leq b$$

Where the matrix A and vector b are defined such that for any indexes i, j we have $|f_i - f_j| \leq d_{i,j}$ with $d_{i,j}$ the metric defined over the measured space, in our case the state space $\mathcal{S}$. Hence we propose to solve the program 2 under constraints 3 to 5. Let us first show that this problem is convex. Clearly, the objective function in Equation 2 is linear, hence convex, and the constraints 3 and 4 define a convex set. We prove that the 1-Wasserstein distance is convex in Lemma 1.

Lemma 1. ***Convexity of the 1-Wasserstein distance.*** *The 1-Wasserstein distance is convex i.e. for $\lambda \in [0,1]$, (X, d_X) a Polish space and any three probability measures w_0, w_1, w_2 on X, the following holds:*

$$W_1(w_0, \lambda w_1 + (1-\lambda)w_2) \leq \lambda W_1(w_0, w_1) + (1-\lambda) W_1(w_0, w_2)$$

Proof. We use the dual representation of the 1-Wasserstein distance of Definition 1.

$$
\begin{aligned}
&W_1(w_0,\lambda w_1 + (1-\lambda)w_2) \\
&\quad = \sup_{f\in \text{Lip}_1} \int_X f(x)(w_0(x) - \lambda w_1(x) - (1-\lambda)w_2(x))dx \\
&\quad = \sup_{f\in \text{Lip}_1} \int_X \left(\lambda f(x)(w_0(x) - w_1(x)) + (1-\lambda) f(x)(w_0(x) - w_2(x))\right) dx \\
&\quad \leq \lambda \sup_{f\in \text{Lip}_1} \int_X f(x)(w_0(x) - w_1(x))dx + (1-\lambda) \sup_{f\in \text{Lip}_1} \int_X f(x)(w_0(x) - w_2(x))dx \\
&\quad \leq \lambda W_1(w_0, w_1) + (1-\lambda) W_1(w_0, w_2)
\end{aligned}
$$

Where we used the linearity of the integral and the triangle inequality on the $\sup$ operator. □

The program 2 is thus convex. One can also observe that the gradient of the objective function is constant, equal to $+v$. Furthermore, p_0 is an admissible initial point that we could use for a gradient descent method. However, given p_0, following the descent direction $-v$ may break the constraints 3 and 4. One would have to project this gradient onto a certain, unknown, set of hyperplanes in order to apply the gradient method descent. Let us note $\text{proj}(v)$ the resulting projected gradient, that is unknown.

We remark that the vector $p_{\text{sat}} = (0, \cdots, 0, 1, 0, \cdots, 0)$ with 1 at the index $\arg\min_i v_i$ where v_i denotes the ith coefficient of v, is the optimal solution of the program 2 when we remove the Wasserstein constraint 5. One can observe that the optimal solution with the constraint 5 would as well be p_{sat} if the constant C is *big enough*. As a result, the descent direction $\nabla = p_{\text{sat}} - p_0$ is the one to be followed in this setting when applying the gradient descent method to this case. Furthermore, following ∇ from p_0 until p_{sat} never breaks the constraints 3 and 4. Since the gradient of the objective function is constant, there can exist only one $\text{proj}(v)$. ∇ fulfils the requirements, hence we have $\text{proj}(v) = \nabla$.

We can now apply the gradient method descent with the following 1-shot rule since the gradient is constant:

$$
\bar{p} := p_0 + \lambda\nabla \text{ with, } \begin{cases} \lambda = 1 \text{ if } W_1(p_{\text{sat}}, p_0) \leq C \\ \lambda = C\, W_1(p_{\text{sat}}, p_0) \end{cases}
$$

Indeed, in the first case, we can follow ∇ until the extreme distribution p_{sat} without breaking the constraint 5. Going further is trivially infeasible.

In the second case, we have to stop *in between* so that the constraint 5 is saturated. In such a case, we cannot go further without breaking this constraint and we recall that no projected gradient could be found by uniqueness of this gradient in our setting. Hence we have the following equality:

$$
\begin{aligned}
W_1(p_0 + \lambda\nabla, p_0) &= C \\
\max_{Af\leq b} f^\top (p_0 + \lambda\nabla - p_0) &= C \\
\lambda \max_{Af\leq b} f^\top \nabla &= C \\
\lambda &= C\, W_1(p_{\text{sat}}, p_0)
\end{aligned}
$$

Where we used the fact that $\nabla = p_{\text{sat}} - p_0$. The latter result concludes the proof.

5 Proof of Property 5

Let us consider a tree developed with Algorithm 1 with a heuristic function $H : s \mapsto H(s)$ used to estimate the value of a leaf node. The set of the leaves nodes is denoted by $\mathcal{L}$ and we have the following uniform upper bound $\delta > 0$ on the heuristic error:

$$
\forall \nu^{s,t} \in \mathcal{L},\ |H(s) - \overline{V}^*_{t_0,t}(s)| < \delta \tag{7}
$$

We want to prove the following result for a decision and chance nodes $\nu^{s,t}$ and $\nu^{s,t,a}$ at any depth $d \in [0, d_{\max}]$:

$$|V(\nu^{s,t}) - \overline{V}^*_{t_0,t}(s)| \leq \gamma^{(d_{\max}-d)}\delta \tag{8}$$

$$|V(\nu^{s,t,a}) - \overline{Q}^*_{t_0,t}(s,a)| \leq \gamma^{(d_{\max}-d)}\delta \tag{9}$$

The proof is made by induction, starting at depth $d_{\max}$ and reversely ending at depth 0. At $d_{\max}$, the nodes are leaf nodes, their values is estimated with the heuristic function i.e. $V(\nu^{s,t}) = H(s)$. Hence the result is directly proven by hypothesis in Equation 7. We will now start by proving the result for the chance nodes which come as the first parents of the decision node for which we initialized the induction proof. Then we extend it to the parents decision nodes which completes the proof.

Chance nodes case. Consider any chance node $\nu^{s,t,a}$ at depth $d \in [0, d_{\max}]$. We suppose that the property is true for depth $d+1$, thus we have for any decision node at $d+1$ denoted by $\nu^{s',t'}$:

$$|V(\nu^{s',t'}) - \overline{V}^*_{t_0,t'}(s')| \leq \gamma^{(d_{\max}-(d+1))}\delta$$

Following Equation 6 of the paper, we have by construction:

$$V(\nu^{s,t,a}) = \overline{R}_t(s,a) + \gamma \sum_{s'} \overline{p}_t(s' \mid s,a) V(\nu^{s',t'})$$

By definition, the true Q-value function defined by the Bellman Equation 2 gives the true target value:

$$\overline{Q}^*_{t_0,t}(s,a) = \overline{R}_t(s,a) + \gamma \sum_{s'} \overline{p}_t(s' \mid s,a) \overline{V}^*_{t_0,t'}(s')$$

Hence, using the induction hypothesis, we have the following inequalities proving the result of Equation 9:

$$\begin{aligned}
|V(\nu^{s,t,a}) - \overline{Q}^*_{t_0,t}(s,a)| &= \gamma \left| \sum_{s'} \overline{p}_t(s' \mid s,a) V(\nu^{s',t'}) - \sum_{s'} \overline{p}_t(s' \mid s,a) \overline{V}^*_{t_0,t'}(s') \right| \\
&\leq \gamma \sum_{s'} \overline{p}_t(s' \mid s,a) \left| V(\nu^{s'}) - \overline{V}^*_{t_0,t'}(s') \right| \\
&\leq \gamma \sum_{s'} \overline{p}_t(s' \mid s,a) \gamma^{(d_{\max}-(d+1))}\delta \\
&\leq \gamma^{(d_{\max}-d)}\delta
\end{aligned}$$

Decision nodes case. Consider now any decision node $\nu^{s,t}$ at the same depth $d \in [0, d_{\max})$. The value of such a node is given by Equation 5 of the paper and the following holds.

$$V(\nu^{s,t}) = V(\nu^{s,t,\bar{a}}), \text{ with, } \bar{a} = \arg\max_{a \in \mathcal{A}} V(\nu^{s,t,a})$$

Similarly, we define $a^* \in \mathcal{A}$ as follows:

$$\overline{V}^*_{t_0,t}(s) = \overline{Q}^*_{t_0,t}(s,a^*), \text{ with, } a^* = \arg\max_{a \in \mathcal{A}} \overline{Q}^*_{t_0,t}(s,a)$$

We distinguish two cases: 1) if $\bar{a} = a^*$ and 2) if $\bar{a} \neq a^*$. In case 1), the result is trivial by writing the value of the decision node as the value of the chance node with the action a^* and using the – already proven for depth d – result of Equation 9.

$$\begin{aligned}
|V(\nu^{s,t}) - \overline{V}^*_{t_0,t}(s)| &= |V(\nu^{s,t,a^*}) - \overline{Q}^*_{t_0,t}(s,a^*)| \\
&\leq \gamma^{(d_{\max}-d)}\delta
\end{aligned}$$

In case 2), the maximizing actions are different. Still following Equation 9, we have that $V(\nu^{s,t,a^*}) \geq \overline{Q}^*_{t_0,t}(s,a^*) - \gamma^{(d_{\max}-d)}\delta$. Yet, since $\bar{a}$ is the maximizing action in the tree, we have that $V(\nu^{s,t,\bar{a}}) \geq V(\nu^{s,t,a^*})$. By transitivity, we can thus write the following:

$$\begin{aligned}
& V(\nu^{s,t,\bar{a}}) \geq \overline{Q}^*_{t_0,t}(s,a^*) - \gamma^{(d_{\max}-d)}\delta \\
\Rightarrow \quad & \overline{Q}^*_{t_0,t}(s,a^*) - V(\nu^{s,t,\bar{a}}) \leq \gamma^{(d_{\max}-d)}\delta
\end{aligned} \tag{10}$$

Furthermore, still following Equation 9, we have that $\overline{Q}^*_{t_0,t}(s,\bar{a}) \geq V(\nu^{s,t,\bar{a}}) - \gamma^{(d_{\max}-d)}\delta$. Yet, since a^* is the maximizing action in $\widehat{\text{MDP}}$, we have that $\overline{Q}^*_{t_0,t}(s,a^*) \geq \overline{Q}^*_{t_0,t}(s,\bar{a})$. By transitivity, we can thus write the following:

$$\begin{aligned} & \overline{Q}^*_{t_0,t}(s,a^*) \geq V(\nu^{s,t,\bar{a}}) - \gamma^{(d_{\max}-d)}\delta \\ \Rightarrow \quad & V(\nu^{s,t,\bar{a}}) - \overline{Q}^*_{t_0,t}(s,a^*) \leq \gamma^{(d_{\max}-d)}\delta \end{aligned} \tag{11}$$

By assembling equations 10 and 11, we prove equation 8 and the proof by induction is complete.

6 Proof of Property 6

Let $s, t_0, t \in \mathcal{S} \times \mathcal{T} \times \mathcal{T}$ be. We consider the two snapshots MDP_{t_0} and MDP_t and are interested in the values of s within those two snapshots using the random policy π. We note $V^{\pi}_{\text{MDP}_{t_0}}(s)$ and $V^{\pi}_{\text{MDP}_t}(s)$ those values. Let $n \in \mathbb{N}$ be. We note $V^{\pi,n}_{\text{MDP}_{t_0}}(s)$ and $V^{\pi,n}_{\text{MDP}_t}(s)$ the finite horizon values defined as follows:

$$V^{\pi,n}_{\text{MDP}_{t_0}}(s) = \mathbb{E}\left\{ \sum_{i=0}^{n} \gamma^i r_{t_0}(s_i, a_i, s_{i+i}) \middle| \begin{array}{l} s_0 = s, \\ s_{i+1} \sim p_{t_0}(\cdot \mid s_i, a_i),\ i \geq 0 \\ a_i \sim \pi(\cdot),\ i \geq 0 \end{array} \right\}$$

where we replace t_0 by t for the definition of $V^{\pi,n}_{\text{MDP}_t}(s)$. We first prove a result on the finite horizon values in Lemma 2.

Lemma 2. *We consider an (L_p, L_R)-LC-NSMDP. For $s, t, t_0 \in \mathcal{S} \times \mathcal{T} \times \mathcal{T}$ and $n \in \mathbb{N}$, the finite horizon of the values of s within the snapshots MDP_t and MDP_{t_0} verify:*

$$\begin{aligned} |V^{\pi,n}_{MDP_{t_0}}(s) - V^{\pi,n}_{MDP_t}(s)| &\leq L_{V_n}|t - t_0| \\ \text{with, } L_{V_n} &= \sum_{i=0}^{n} \gamma^i L_R \end{aligned}$$

Proof. The proof is made by induction. Let us start with $n = 0$. By definition, we have:

$$\begin{aligned} \left| V^{\pi,0}_{\text{MDP}_{t_0}}(s) - V^{\pi,0}_{\text{MDP}_t}(s) \right| &= \left| \int_{\mathcal{A}} \pi(a \mid s) \left(R_{t_0}(s,a) - R_t(s,a) \right) da \right| \\ &\leq \int_{\mathcal{A}} \pi(a \mid s) L_R |t_0 - t| da \\ &\leq L_R |t_0 - t| \end{aligned}$$

Which verifies the property for $n = 0$ with $L_{V_0} = L_R$. Let us now consider $n \in \mathbb{N}$ and suppose the property true for rank $n-1$. By writing the Bellman equation for the two value functions, we obtain the following calculation:

$$\begin{aligned} V^{\pi,n}_{\text{MDP}_{t_0}}(s) - V^{\pi,n}_{\text{MDP}_t}(s) = \int_{\mathcal{S}\times\mathcal{A}} \pi(a|s) \Big[& p_{t_0}(s' \mid s,a)(r_{t_0}(s,a,s') + \gamma V^{\pi,n-1}_{\text{MDP}_{t_0}}(s')) - \\ & p_t(s' \mid s,a)(r_t(s,a,s') + \gamma V^{\pi,n-1}_{\text{MDP}_t}(s')) \Big] ds' da \\ \text{i.e. } V^{\pi,n}_{\text{MDP}_{t_0}}(s) - V^{\pi,n}_{\text{MDP}_t}(s) = \int_{\mathcal{A}} \pi(a|s) \Big[& A(s,a) + B(s,a) \Big] da \end{aligned} \tag{12}$$

With the following values for $A(s,a)$ and $B(s,a)$:

$$\begin{aligned} A(s,a) &= \int_{\mathcal{S}} (r_{t_0}(s,a,s') + \gamma V^{\pi,n-1}_{\text{MDP}_{t_0}}(s')) \Big[p_{t_0}(s' \mid s,a) - p_t(s' \mid s,a) \Big] ds' \\ B(s,a) &= \int_{\mathcal{S}} p_t(s' \mid s,a) \Big[r_{t_0}(s,a,s') - r_t(s,a,s') + \gamma (V^{\pi,n-1}_{\text{MDP}_{t_0}}(s') - V^{\pi,n-1}_{\text{MDP}_t}(s')) \Big] ds' \end{aligned}$$

Let us first bound $A(s,a)$ by noticing that $s' \mapsto r_{t_0}(s,a,s') + \gamma V^{\pi,n-1}_{\text{MDP}_{t_0}}(s')$ is bounded by $\frac{1}{1-\gamma}$. Since the function $s' \mapsto \frac{1}{1-\gamma}$ belongs to Lip_1, we can write the following:

$$\begin{aligned}
A(s,a) \leq & \sup_{f \in \text{Lip}_1} \int_{\mathcal{S}} f(s') \Big[p_{t_0}(s' \mid s,a) - p_t(s' \mid s,a) \Big] ds' \\
\leq & W_1(p_{t_0}, p_t) \\
\leq & L_p |t - t_0|
\end{aligned}$$

B is straightforwardly bounded using the induction hypothesis:

$$\begin{aligned}
B(s,a) \leq & \int_{\mathcal{S}} p_t(s' \mid s,a) \Big[L_r |t - t_0| + \gamma \sum_{i=0}^{n-1} \gamma^i L_R |t - t_0| \Big] ds' \\
\leq & L_r |t - t_0| + \sum_{i=1}^{n} \gamma^i L_R |t - t_0|
\end{aligned}$$

We inject the result in Equation 12:

$$\begin{aligned}
V^{\pi,n}_{\text{MDP}_{t_0}}(s) - V^{\pi,n}_{\text{MDP}_t}(s) \leq & \int_{\mathcal{A}} \pi(a|s) \Big[L_p |t - t_0| + L_r |t - t_0| + \sum_{i=1}^{n} \gamma^i L_R |t - t_0| \Big] da \\
\leq & (L_p + L_r) |t - t_0| + \sum_{i=1}^{n} \gamma^i L_R |t - t_0| \\
\leq & L_R |t - t_0| + \sum_{i=1}^{n} \gamma^i L_R |t - t_0| \\
\leq & \sum_{i=0}^{n} \gamma^i L_R |t - t_0|
\end{aligned}$$

The same result can be derived with the opposite expression. Hence, taking the absolute value, we prove the property at rank n, i.e.

$$|V^{\pi,n}_{\text{MDP}_{t_0}}(s) - V^{\pi,n}_{\text{MDP}_t}(s)| \leq \sum_{i=0}^{n} \gamma^i L_R |t - t_0| \tag{13}$$

which concludes the proof by induction. □

The proof of Property 6 follows easily by remarking that the sequence L_{V_n} of Lemma 2 is geometric and converges towards $\frac{L_R}{1-\gamma}$ when n goes to infinity.

7 Non-Stationary bridge environment

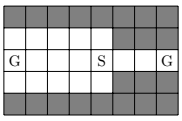


Figure 1: The Non-Stationary bridge environment

8 Informations about the Machine Learning reproducibility checklist

For the experiments run in Section 6, the computing infrastructure used was a laptop using four 64-bit CPU (model: Intel(R) Core(TM) i7-4810MQ CPU @ 2.80GHz). The collected samples sizes and number of evaluation runs for each experiment is summarized in Table 1.

Experiment	Number of experiment repetitions	Number of episodes	Maximum length of episodes	Upper bound on the number of computed transition samples (s, a, r, s')
Non-Stationary Bridge Figure 1	3 (one per agent)	96	10	89,579,520

Table 1: Summary of the number of experiment repetition, number of sampled tasks, number of episodes, maximum length of episodes and upper bounds on the number of collected samples.

The displayed confidence intervals in Figure 2a is 50% of the estimated confidence interval $\bar{\sigma}$ computed w.r.t. the following formula:

$$\bar{\sigma} = \sqrt{\frac{1}{1-N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \quad \text{where,} \quad \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i,$$

with $D = \{x_i\}_{i=1}^{N}$ the set of the collected data (discounted return in this case). No data were excluded neither pre-computed. Hyper-parameters were determined to our appreciation, they may be sub-optimal but we found the results convincing enough to display interesting behaviours.

References

Pravin M. Vaidya. Speeding-up linear programming using fast matrix multiplication. In *30th Annual Symposium on Foundations of Computer Science*, pages 332–337. IEEE, 1989.