Linearly-solvable Markov Decision Processes

Hugues Derogis
hugues.derogis@gmail.com

Yann Labbé
yann.labbe@ens-paris-saclay.fr

Abstract

In this report, we present the basis of the Linearly-solvable Markov Decision Processes (LMDPs), a class of processes for which the optimal Bellman equation can be reduced to a linear system. Firstly, we present LMDPs, their assumptions and the most important equations that can be derived in different settings. Secondly, we present examples of applications in which the LMDP theory can be exploited. We aim at investigating the strength of the LMDP assumptions and illustrate in which settings it is interesting to use these decision processes.

1 Markov Decision Processes and Linearly-solvable Markov Decision Processes

1.1 General definitions

Many situations require to choose the best action to achieve a result starting from a defined state, for example, investing in stocks, defining when to change pieces of a system, which path to choose to reach a goal. These interactions between an agent and an environment are modeled with Markov Decision Processes which can be defined as :

Definition 1. *A Markov Decision Process is a tuple $M = (X, A, p, c)$ where X is the state space, A is the action space, which can both be discrete or continuous, p is the transition probability from a state x to a state y, considering the action a, i.e.*

$$p(y \mid x, a) = \mathbb{P}(x_{t+1} = y \mid x_t = x, a_t = a) \tag{1}$$

and $c(x, a)$ is the cost of choosing action a in state x.

One of the major assumptions is that the process follows a Markov Chain, that is to say, the tuple (x, a, p) is sufficient to characterize the next state y. Our goal is to find the best actions to choose when we are in a particular state. For that, we introduce decision rules and policies.

Definition 2. .
A Decision rule is a mapping between state and action. $\pi_t : X \to A$ or $\pi_t : X \to \Delta(A)$

A Policy is a sequence of decision rules to achieve a goal. $\pi : (\pi_1, \pi_2, ..)$

In our problems, we want to minimize the cost along the path (or maximize the reward along the path depending on the problem). To model that, we introduce the Value function as follows (for coherence with following parts, we will consider an infinite horizon and a total cost with terminal states) :

Definition 3. *We consider an initial state x_0, a policy π and an infinite time horizon with terminal states. The Value function V^π is defined as :*

$$V^\pi(x) = \mathbb{E}[\sum_{t=0}^{\infty} c(x_t, \pi(x_t)) \mid x_0 = x; \pi] \tag{2}$$

which represents the expectation of the sum of costs from the initial state x_0, along the path defined by the policy π.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Using this definition, we want to find the optimal policy, i.e.

$$\pi^* \in \text{argmin}_{\pi \in \Pi}(V^\pi)\,;\, V^* = V^{\pi^*} \tag{3}$$

π^* achieves the largest possible value function in every state x

It can be proved that the optimal Value function is the solution to the optimal Bellman Equation :

$$\forall x \in X \quad V^*(x) = \min_{a \in A}[c(x,a) + \sum_y (p(y \mid x,a)V^*(y)] \tag{4}$$

The optimal Bellman equation is a highly non-linear system of equations due to the min operator. Thus, in the general case, we cannot get further analytically. We then have to use techniques to be able to compute the optimal Value function and optimal policy.

1.2 Linearly-solvable Markov Decision Processes

1.2.1 Introduction to LMDPs

The optimal Bellman equation allows to describe very general problems but in many cases, it is overly general and does not allow us to make use of the particularities of the problem to make the computations easier. In order to make analytical computations easier for some specific problems - that can actually gather many real-life problems -, Todorov introduced in 2006 what he called Linearly-solvable Markov Decision Processes (LMDPs)[3]. LMDPs take their name from the fact that their modelisation leads to linear systems which are much simpler to solve. We sum up in the next paragraph the major assumptions and definitions about this new model.

1.2.2 LMDP : modelisation and assumptions

In this part, we will only consider LMDPs defined on a finite set of states, as it was first introduced by Todorov in [3]. The second part of this report will evaluate the continuous LMDPs and how to link them to discrete LMDPs.
We consider $\mathcal{X}$ a finite space of states, P a uncontrolled transition matrix and c a uncontrolled state cost. LMDPs rely on two major assumptions.

1. In LMDPs, instead of only choosing an action, the agent is now capable of directly modifying the transition probabilities from one state to another. That can be written as, the agent chooses Q such as :
$$Q(x' \mid x) = P(x' \mid x, Q) \tag{5}$$
However, a state that could not be reached in the uncontrolled MDP should not be reachable in the controlled LMDP. That is to say, the agent can choose a transition matrix $Q : \mathcal{X} \to \Delta(\mathcal{X})$ but the support of Q has to be included in the support of the uncontrolled transition matrix P. This is the 1^{st} assumption made by LMDPs.
2. The cost incurred is then composed of an uncontrolled/passive state cost c and a controlled cost. The 2^{nd} assumption made by LMDPs is that this controlled cost has to be in the form of the KL divergence between Q and P. We thus have :
$$l(x,Q) = c(x) + KL(Q(. \mid x) \mid\mid P(. \mid x)) \tag{6}$$
with KL defining the Kullback-Leibler divergence. Note that the 1^{st} assumption garanties that the KL divergence is well-defined i.e., $Q(x' \mid x) = 0$ when $P(x' \mid x) = 0$.

These two assumptions - (1) the agent can modify the transition kernel but only in the support of the uncontrolled transition kernel (2) the controlled cost has to be expressed as KL divergence - allow to get simple formulations for the optimal control and optimal value function. This can be seen by two different ways, via Dynamic Programming using the optimal Bellman Equation and via Convex Optimization [2].

Note 1. *A common way to model the assumption (1) is to introduce a control vector u and consider that the controlled transition probabilities are described by the relation $q_{ij}(u) = p_{ij}exp(u_j)$. With these notations, the controls can completely reshape the transition probabilities and it forces the constraint on the kernel of Q. For the sake of generality, we do not use these notations here but they will be used in part 3.*

1.2.3 Obtaining the Linear formulation by Dynamic Programming

The optimal Bellman Equation that we want to solve is :

$$\min_q[c(x) + KL(q(. \mid x) \mid\mid p(. \mid x)) + \sum_y (q(y \mid x)V^*(y)] \tag{7}$$

This minimization can be solved in closed form using Lagrange multipliers. Following computations by Todorov in [4] we introduce :

- $z(x) = \exp(-V(x))$
- $\mathcal{G}[z](x) = \sum_{x'} p(x' \mid x)z(x')$ which is a normalization term, useful to use the properties of the KL divergence

With these new variables, the optimal Bellman equation can be rewritten - after computations - as :

$$\min_{\mathrm{q}}[c(x) - \log(\mathcal{G}[z](x)) + KL(q(. \mid x) \mid\mid \frac{p(. \mid x)z(.)}{\mathcal{G}[z](x)})] \tag{8}$$

The only term which depends on q is the KL divergence, which is minimum (and is 0) when the two densities are equal. We can conclude that this minimum is reached for :

$$q^*(x' \mid x) = \frac{p(x' \mid x)z(x')}{\mathcal{G}[z](x)}) \tag{9}$$

which defines the optimal control in the case of LMDP. Using this closed form solution for q^*, we can replace q^* in the optimal Bellman equation.

$$V^*(x) = c(x) - \log(\mathcal{G}[z](x)) + KL(q^*(. \mid x) \mid\mid \frac{p(. \mid x)z(.)}{\mathcal{G}[z](x)} = c(x) - \log(\mathcal{G}[z](x)) \tag{10}$$

Then :

$$\forall x \in \mathcal{X}, z^*(x) = \exp(-V^*(x)) = \exp(-c(x))\mathcal{G}[z](x)) \tag{11}$$

As $(\mathcal{X} = (x_1, x_2, ..., x_n))$ we can write :

$$G = Diag(\exp(-c(x_i))) \tag{12}$$

$$P = (p(x_i, x_j))_{i,j} \tag{13}$$

Then, we have :

$$z = GPz \tag{14}$$

Thus, our problem can be expressed with a linear equation.

Note 2. *We only considered the case of infinite horizon total-cost with terminate states. One can show that the same results can be obtained - with some modifications in the final formulas - for the other models of MDPs, i.e. infinite horizon with discount, finite horizon and infinite horizon average cost.*

Note 3. *In the case of infinite horizon average cost, [2] proposed another way to get these results and solve it by using a Convex Optimization formulation for a set of feasible stationary transition measures π.*

1.2.4 Computing the optimal value function

In the case of infinite horizon total-cost with terminal states, we have :

$$z = GPz \tag{15}$$

This is an eigenvalue problem. The **Power iteration** method is commonly used to solve these problems. It is defined by the algorithm :

$$z_{k+1} = GPz_k \quad z_0 = 1 \tag{16}$$

This algorithm converges to the unique solution z.

Linear Algebra equation

However, for this simple case, we can have an even simpler analytical solution (but which can need more computations) and compute z with a matrix inversion. Noting N and T the non-terminal and terminal states, if we write G, P and z in the order (N, T), we have :

$$P = \begin{bmatrix} P_{NN} & P_{NT} \\ 0 & I \end{bmatrix}; G = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}; z = \begin{bmatrix} z_N \\ z_T \end{bmatrix} \tag{17}$$

with $D_1 = Diag(\exp(c_{NN}))$ and $D_2 = Diag(\exp(c_{TT}))$

Following the computations in Appendix A, we have :

$$z_T = \exp(-c_{TT}) \tag{18}$$

$$(Diag(\exp(c_{NN})) - P_{NN})z_N = P_{NT}Diag(\exp(-c_{TT})) \tag{19}$$

Thus, we can compute the optimal value directly from the equations (18) and (19). However, this closed-form solution may not be tractable in reality and the power iteration method is often more efficient. We will implement the power iteration in Part 3.

2 Continuous LMDPs

In the first part, we focused on MDPs with discrete state space and time. However, many problems have continuous state spaces and continuous time. Many techniques using the theory of optimal control have been developed to solve these continuous optimal control problems. We will not develop these techniques.

However, we would like to see if we can apply the LMDPs defined above to solve efficiently some continuous problems. We will show that problems with a Brownian continuous dynamics and a quadratic cost can be well approximated by LMDPs, under some extra assumptions. The global method discretizes time with timestep Δt, consider continuous-state discrete-time problems and show that we recover the continuous time problem by taking the limit $\Delta t \to 0$.

2.1 Definition

We consider the special case of continuous state and time controlled MDPs with a Brownian dynamics and a quadratic cost. In order to get the same property in continuous state settings as in discrete state LMDPs, we have to add another constraint, which is that **the control q and the noise are in the same subspace**. Thus, we will consider the following dynamics :

$$dx = A(x)dt + B(x)(q(x)dt + \sigma dw) \tag{20}$$

and

$$l(x, q) = c(x) + \frac{|| \, q \, ||^2}{2\sigma^2} \tag{21}$$

Even if this seems to be a quite particular problem, many mechanical problems are actually in that form. For example, the car-on-the-hill problem mentioned in section 3.4 has the same dynamics and cost function.

Approximation with Linearly-solvable MDPs

As said above, in order to apply the LMDP formalism to this problem, we discretize the dynamics with Δt and consider the continuous-state discrete-time MDP defined by the following dynamics :

$$x_{k+1} = x_k + A(x_k)\Delta t + B(x_k)[q_k(x_k)\Delta t + \sigma \Delta w] \tag{22}$$

with x_k the (continuous) state value at time step k, $A(x_k)$ the uncontrolled dynamics, $B(x_k)q_k(x_k)$ the controlled dynamics and $\sigma \Delta w$ the noise which follows a Gaussian $\sigma \mathcal{N}(0, I\Delta t)$.

(22) can be rewritten :

$$x_{k+1} = x_k + [A(x_k) + B(x_k)q_k(x_k)]\Delta t + B(x_k)\sigma \Delta w \tag{23}$$

The uncontrolled dynamics is defined by :

$$x_{k+1} = x_k + A(x_k)\Delta t + B(x_k)\sigma\Delta w \tag{24}$$

The controlled and uncontrolled transition probabilities are then :

$$p(x_{k+1} \mid x_k, q_k) \sim \mathcal{N}(x_k + (A(x_k) + B(x_k)q_k(x_k))\Delta t, \sigma^2 BB^T) \tag{25}$$

$$p(x_{k+1} \mid x_k) \sim \mathcal{N}(x_k + A(x_k)\Delta t, \sigma^2 BB^T) \tag{26}$$

We can see that unlike the Discrete LMDPs where the agent could completely reshape the transition probabilities, in the hypothesis of continuous LMDPs, the agent is only able to shift the transition probabilities $(A(x_k) + B(x_k)q_k(x_k))$. In the LMDP formalism, as in (6), we want to write the total cost as :

$$l(x_k, q_k) = c(x_k)\Delta t + KL(p(x_{k+1} \mid x_k, q_k) \mid\mid p(x_{k+1} \mid x_k)) \tag{27}$$

Todorov proved in [5] that by taking the limit $\Delta t \to 0$, the solution of the discrete-time problem defined in (22) converges to the solution of continuous-time problem defined in (20). We will not develop the proof here. However, one can have the intuition by showing that the KL cost generalizes the quadratic cost of the problem (21).

Indeed, $p(x_{k+1} \mid x_k)$ and $p(x_{k+1} \mid x_k, q_k)$ follow Gaussian distributions, so we can apply the following formula to compute the KL in the case of Gaussian distributions :

$$KL(\mathcal{N}_0 \mid\mid \mathcal{N}_1) = \frac{1}{2}(tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T\Sigma_1^{-1}(\mu_1 - \mu_0) - n - \ln(\frac{det\Sigma_1}{det\Sigma_0}) \tag{28}$$

with n the dimension of the states space, $\Sigma_{0,1}$ the covariance matrices and $\mu_{0,1}$ the means. Here, $\Sigma_0 = \Sigma_1$ and we thus have :

$$KL(p(x_{k+1} \mid x_k, q_k) \mid\mid p(x_{k+1} \mid x_k)) = \frac{\Delta t}{2\sigma^2} \mid\mid q_k \mid\mid^2 \tag{29}$$

Thus, we can write :

$$l(x_k, q_k) = (c(x_k) + \frac{1}{2\sigma^2} \mid\mid q_k \mid\mid^2)\Delta t \tag{30}$$

When we take the limit $\Delta t \to 0$, we recover the quadratic cost function in (21). Thus, continuous problems following the equations (20) and (21) can be well approximated by continuous-state discrete-time LMDPs (for a precise proof, see [5]). We will see how to use this property to get efficient algorithms to compute the optimal value function in continuous problems.

2.2 Embedding of continuous-state LMDPs with discrete-state LMDPs

We have seen that we can approximate continuous problems by continuous-state discrete-time LMDPs. To solve these continuous-state LMDPs it is quite natural to consider an embedding between the continuous-state LMDP and our discrete-state LMDP defined in Part 1. Different embeddings were considered in the last few years : the most natural embedding is grid discretization, but more complex embeddings such as the Soft State Aggregation Scheme [1] were proposed. We will describe quickly the latter method, while describing more extensively the grid embedding. An example of grid embedding is mentioned in section 3.4.

Grid Embedding

In the case of the grid embedding, we define a set of points x_n such that the mean and variance of our discrete model match the mean and variance of the continuous model in the case of uncontrolled MDPs. In the continuous-state model we had :

$$p(x_{k+1} \mid x_k) \sim \mathcal{N}(x_k + A(x_k)\Delta t, \sigma^2 BB^T) \tag{31}$$

Then, writing $\bar{y_n}$ the expectation of the transition from a state x_n in the discrete model and P_{nk} the probability to go from x_n to x_k in the discrete model, we have : $\bar{y_n} = \sum_k P_{nk}x_k$ and the following matching has to be respected :

$$\bar{y_n} = x_n + A(x_n)\Delta t \tag{32}$$

$$\sum_k P_{nk}(x_k - \bar{y_n}) = \sigma^2 BB^T \tag{33}$$

Note that the grid points x_n and the values of P_{nk} are the parameters of the grid embedding that we have to choose.

Using this embedding, we can thus write :

$$\mathbf{z} = [\exp(-v(x_1)), \exp(-v(x_2)), ..., \exp(-v(x_n))] \tag{34}$$

and $G = diag(exp(-c(x_n)\Delta t))$

With these notations, we retrieve the previous linear equation :

$$\mathbf{z} = GP\mathbf{z} \tag{35}$$

As seen previously, this linear equation can be solved efficiently using the power iteration method.

Soft State Aggregation Scheme and Gaussian RBF

In the case of **Soft State Aggregation Scheme** defined by Zhong in [1], instead of considering a grid, we consider clusters and introduce two probabilities : the Aggregation probability $\Phi_i(x) = p(i \mid x)$ which represents the degree of membership of the continuous state x in the discrete cluster i and the deaggregation probability $d_i(x) = p(x \mid i)$ which represents the degree to which the discrete cluster i is represented by the continuous state x. By choosing good aggregation and deaggregation probabilities - as described in [1] - , it is possible to get the same linear equation as before over the set of clusters, compute the value function over the clusters and then retrieve the continuous value function.

3 Applications of LMDPs

The previous parts have introduced the notion of LMDPs. It has been shown that the optimal control and optimal value function for this type of decision process can be computed efficiently. However, this formalism may appear very theoretical and only convenient in order to derive specific equations.

In this part, we use the LMDP formalism in order to solve real problems, some of which have been introduced in the lectures and TPs of the class. We show that in some problems who might not appear to be linked to LMDPs - or Reinforcement Learning at all - at first sight, we can benefit from the analytical developments in order to produce efficient solving methods.

Note 4. *The following sections are based on the experiments presented in [3] and [4]. This section is not meant to re-develop the analytical modelisations and computations of these papers. We will use some of their derivations in order to focus on the critical analysis of the experimental results and the conclusions that can be drawn.*

Note 5. *With the exception of the section 3.4, all the figures and results that we present are the outputs of our own MATLAB implementations. We re-used some of the parts of Matteo Pirotta's code that was provided as materials for the TP1.*

Note 6. *In the following parts we consider LMDPs with a discrete state space and we use the notations explicited in the Note 1.*

3.1 Shortest path problem

Consider a graph with a set of vertex S and edges E. We note D the adjacency matrix whose elements indicate the presence ($d_{ij} = 0$) or absence ($d_{ij} = 1$) of an edge between vertex i and j.

In the section 3 of [3], Todorov shows that the problem of finding the shortest path $s(i)$ from any vertex i to some vertex in a set $A \subseteq S$ can be reduced to finding the optimal value function of a well-chosen Linear Solvable Markov Decision Process. As we saw in part 1.2.2, a LMDP is

(a) (b)

Figure 1: Shortest path problem using LMDPs. The power iteration method is used to compute V_ρ with $\rho = 50$. The numbers displayed are the $s(i), i \in S$ rounded to the nearest integer.

characterized by a set of states $\mathcal{X}$, an uncontrolled transition matrix P and an uncontrolled state cost vector c.

The LMDP considered has a set of discrete states S, the underlying transition matrix corresponds to a random walk on a graph $p_{ij} = \frac{d_{ij}}{\sum_k d_{ik}}$ and we choose the uncontrolled state costs $c_\rho(i) = \rho\delta_{i\in A}$ where $\rho > 0$ is arbitrary.

By exploiting an upper bound on the controlled cost $l(i, u)$, the following relation can be proved:

$$\forall i \in S \quad s(i) = \lim_{\rho\to\infty} \frac{V_\rho(i)}{\rho} \tag{36}$$

where V_ρ is the optimal value function associated with our LMDP. This equation shows that finding the shortest paths in any graph can be reduced to solving an eigenvalue problem or a linear system as it was shown in 1.2.4. Solving the linear system has in general a complexity of around $O(n^3)$ even though sparsity could be exploited to reduce computations. In [3], Todorov gives the result that in general the power iteration method for solving the linear equation (47) has an average running time that scales linearly with the number of non-zero elements in P. Therefore, this method is the one to be used when scalability is a concern. This last result makes this approach through LMDPs very interesting as the Djikstra's algorithm which is widely used for finding shortest paths has a running time of $O(n\log(n))$.

While the equation (36) involves a limit, we get a good approximation of $s(i)$ by taking ρ large enough, but not too large in order to prevent the terms of $G = \mathrm{Diag}(exp(-c(i))$ from falling under machine precision.

In practice, $\rho = 50$ yields good results at is it shown in our results Figure 1 for two graphs built from grids where we only consider edges between cells in a 4-neighborhood. Here, the LMDP theory for discrete states was used directly without any assumptions.

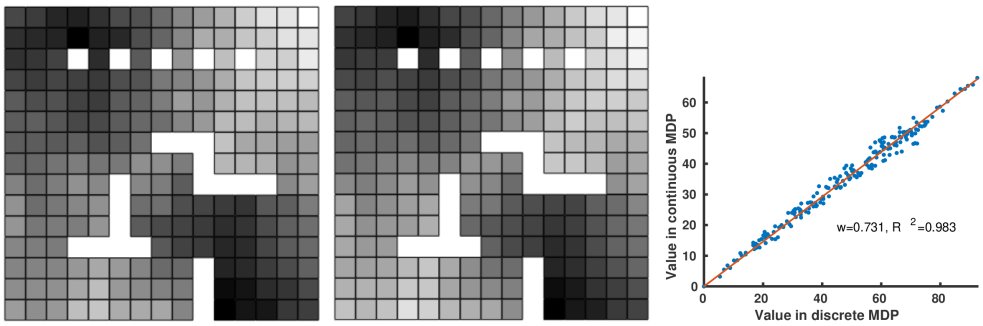
At first sight, the shortest problem did not appear to be related to LMDPs or even Reinforcement Learning because of the lack of agent-environment interaction. Thus, it shows the importance of a good modelisation and the impact it may have on the solving method. Furthermore, it illustrates the presence of a link between a more general type of problem for which we are seeking "optimality".

3.2 Approximation of discrete MDPs

During the course, we have mostly studied the Reinforcement Learning for MDPs with discrete state and actions. We have seen algorithms to find the optimal value function and optimal policy of an MDP such as policy iteration and value iteration. After looking at the LMDP theory, the question that we can raise is: can we benefit from this theory in order to build a more efficient algorithm which is applicable to these MDPs ?

This question is not easy as the agent-environment interaction are fundamentally different in MDPs and LMDPs. In an MDP, the agent only choses an action but in an LMDP, the agent can 'nearly' completely choose the transition probability through a real-valued vector u of costs.

In the section 4 of [3], a method is suggested to approximate discrete MDPs with LMDPs. Under some assumptions that we can fall under by modifying the formulation of our problem, Todorov



(a) Continuous LMDP (b) Discrete MDP (c) Comparison of value functions

Figure 2: Result of the continuous embedding in an infinite-horizon total-cost setting.

suggests to build a LMDP whose controlled transition probabilities and costs match the ones (eq. (37),(38)) of the discrete MDP for each discrete actions, each of which is represented by a control vector $u^a, a \in A(i)$.

$$p_{ij}\exp(u^a_j) = \tilde{p}_{ij}(a) \quad \forall i \in X, j \in N(i), a \in A(i) \tag{37}$$

$$c(i) + \mathrm{KL}\left(p_i \exp(u^a) || p_i\right) = \tilde{l}(i,a) \quad \forall i \in X, a \in A(i) \tag{38}$$

Our traditional discrete MDP has transition probabilities $\tilde{p}(j|i,a)$ and costs $\tilde{l}(i,a)$ and $N(i)$ is the set of next possible states. By exploiting the previous constraints, we can obtain the parameters P and c of an LMDP in closed form and we are able to compute the optimal value function for this LMDP using efficient techniques such as power iteration.

We consider a discrete MDP for which the states correspond to the cells of a grid and in each state, at most 4 actions are available: up, down, left, right. The state-action costs $\tilde{l}(x,a)$ are integers generated randomly between 1 and 10 and zero for two absorbing states. Firstly, we use value-iteration in order to compute the optimal value-function which will serve as ground truth, presented in Figure 2b: the highest value is displayed in white.

Secondly, we compute the parameters P and c of our approximation LMDP using the equations from [3]. The optimal value function is computed from these parameters using power iteration and displayed in Figure 2a. We compare the value functions for each state in the Figure 2c.

As in the paper, the correlation is excellent. The value-function has lower values in the continuous setting because the control space is larger but the order among the states is preserved. The optimal policy for the discrete setting can be computed by finding the vector u^a which is the closest in terms of transition probabilities to the optimal control u^*.

Such results suggests that this approximation is to be considered when dealing with any discrete MDP: computing the optimal value function for an LMDP is equivalent to half a step of a policy iteration. Policy iteration requires to solve a linear system and improve the policy at each iteration whereas only one power-iteration is required in the LMDP formulation.

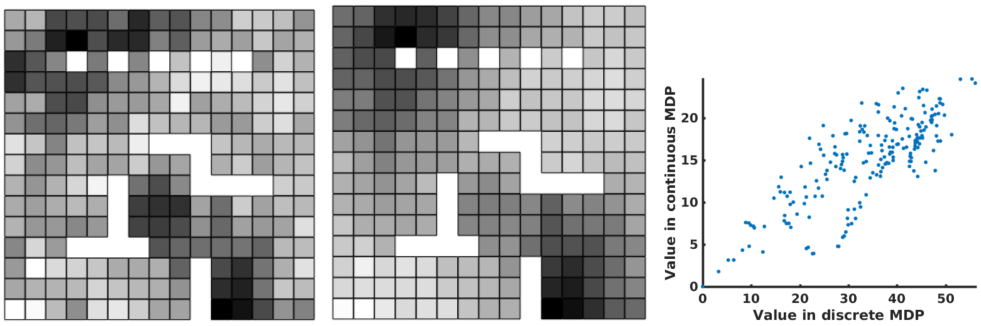
So far, we have only considered the value function which was defined as in the equation (7); in an infinite-horizon total-cost setting with terminal states. In Todorov's paper [3], this is the only case for which results are presented but the equivalent of the equation (14) is presented for the other formulations. We decided to try the previous approximation in the infinite-horizon discounted-cost setting as it was the one we encountered in the TP1. In this setting, the analog of (14) with a discount factor α is:

$$z = GPz^\alpha \tag{39}$$

This equation is non-linear but can be solved via the analog of power-iteration:

$$z_{k+1} = GPz_k^\alpha \quad z_1 = \mathbf{1} \tag{40}$$

We implemented this method and tested it in the same problem as the previous one and the results are presented in the Figure 3.



(a) Continuous LMDP (b) Discrete MDP (c) Comparison of value functions

Figure 3: Result of the continuous embedding in an infinite-horizon discounted-cost setting with discount $\alpha = 0.95$

In this case, the approximation is not satisfying which is probably due to the non-linearities introduced in the equations with this formulation. While we did not test all the alternative formulations, it is likely that the approximation would yield good results in every other formulations because they lead to linear equations.

One has to be careful when using this approximation because no lower bound of the approximation error have been explicited so far, as stated in [4].

3.3 Z-Learning

We have presented applications in which it is required to have the full knowledge of the underlying environment, that is P and c. Z-Learning is an algorithm analog to Q-Learning that can be used to approximate the value function without a model available; using Monte-Carlo simulations.

In the case of Z-Learning, the agent has access to the current state i_k, the next state j_k and the state-cost c_k at each step and the stochastic approximation $\hat{z}$ to the function z is:

$$\hat{z}(i_k) \leftarrow (1 - \alpha_k)\hat{z}(i_k) + \alpha_k \exp(-c_k)\hat{z}(j_k) \tag{41}$$

We constructed two continuous grid-world MDPs with $c(i) = 1\delta_{i \in N}$. Using the optimal controlled transition probabilities computed using the LMDP theory, we constructed two discretes MDPs for which the value function are identical to the continuous ones. We used learning rates $\alpha_k = \frac{1}{N(i_k, a_k)^\beta}$ for Q-Learning and $\alpha_k = \frac{1}{N(i_k)^\beta}$ for Z-Learning. We used an ϵ-greedy policy where the greedy controls are computed using the equation (9) with $\hat{V}$ and the random exploration controls are $u = 0$ for Z-Learning.

The approximation error $\frac{\max_i |V(i) - \hat{V}(i)|}{\max_i V(i)}$ for both algorithms are presented in the Figure 4 for $\beta = 0.7$ and $\epsilon = 0.2$. The results are averaged over 10 runs.

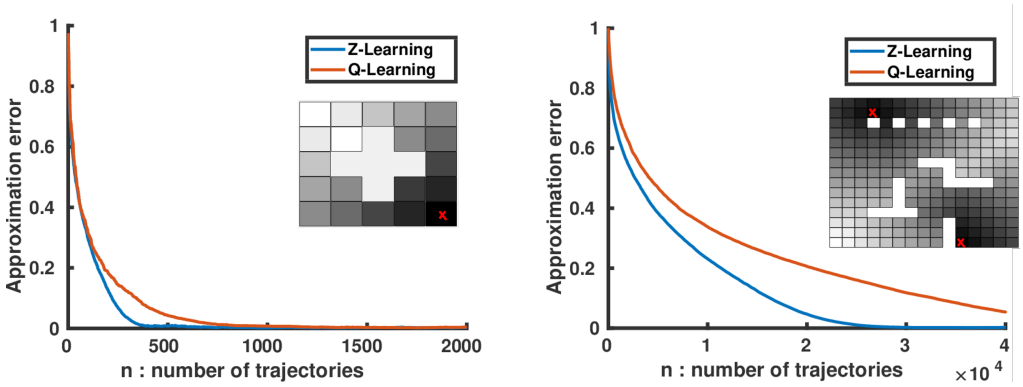
Z-Learning performs better in both cases but the gap is bigger when the dimension increases. Z-Learning is faster to learn its environment, it only has to explore the incurred costs in each states while Q-Learning must also explore the set of all possible actions for each state in order to build a good estimate $Q(x, a)$.

Z-Learning is only suitable for an unknown environment whose interaction with the agent involves a control vector u, we cannot apply it directly to discrete MDPs. The embedding presented in 3.2 cannot be applied here because $\tilde{P}$ and $\tilde{l}$ are unknown.

In the next section, we provide an example where the agent-environment interaction is well suited for LMDPs and where Z-Learning could be used in practice if we had no knowledge of the model.

3.4 Continuous problems - Car-On-The-Hill

In this section, we provide an example of a problem whose dynamics can be modeled with the relation of the equation (23). In this case, the grid embedding technique (2.2) is used to discretize the state space in order to apply the LMDP theory.



(a) (b)

Figure 4: Comparison of Q and Z-Learning

The car-on-the-hill problem is modeled as follows : the state of a car is represented by a vector $x = [x_1, x_2]^T$ with x_1 it's horizontal position and x_2 it's tangential velocity. The tangential velocity of the car can be controlled through a real-valued scalar u. We want the car to reach a state x_T while minimizing the energy injected in the system. The matrices A and B of eq. (23) depend on the terrain.

This problem can be solved by building a discrete MDP obtained by discretizing the state and control spaces and using policy or value-iteration. However, the state and control spaces can be quite large and the computations become very unpractical. Using LMDPs, we can solve the problem while keeping a continuous control space. In [4], Todorov compares the previous approaches, the LMDP approach is ≈ 10 times faster than policy iteration and 100 times faster than value iteration. As in the previous section, it illustrates that in some situations, we can benefit from the formulation with continuous controls in order reduce computations.

Using this grid approximation and the LMDP solving approach, Todorov shows that trajectories simulated with the optimal control u^* that is computed with this discretization method follow very closely the most likely trajectory that can be derived using the optimal control theory.

In this problem, we made two approximations: we used the LMDP theory while the assumption (1) of 1.2.2 is not completely verified, and we discretized the state space. In the context of these approximations, the results are excellent which suggests that there are advantages of using LMDPs even if the assumptions are not completely satisfied.

Note 7. *We did not reproduce the car-on-the-hill experiment of [4]. We tried to implement it and managed to get some results but these were different from those presented in the paper. There is most likely a bug in this part of the code that we did not manage to fix in time.*

4 Conclusion

LMDPs appear as a class of processes for which the theoretical results justify their use in many applications where computationally efficient algorithms are required.

One of the most interesting result is that discrete MDPs can be approximated with LMDPs and solved in a more efficient manner compared to policy and value iteration. In some settings, the results appear as reliable but in some others, this embedding cannot be used which is the main limit that we observed.

Concerning continuous MDPs, due to the continuous nature of the control space of LMDPs, it is not surprising to find many links with the control theory. Using LMDPs for continuous MDPs yields efficient solving algorithms and no drawbacks appear in the experimental results even though the assumptions are only partially respected.

We can say that there are advantages to using LMDPs even when the asumptions are not completely satisfied but the lack of guarantees might be problematic in some conditions.

References

[1] Z. Mingyuan and E. Todorov. Aggregation methods for lineary-solvable markov decision process. 18:11220–11225, 08 2011.

[2] G. Neu and V. Gómez. Fast rates for online learning in linearly solvable markov decision processes. *arXiv preprint arXiv:1702.06341*, 2017.

[3] E. Todorov. Linearly-solvable markov decision problems. In *Advances in neural information processing systems*, pages 1369–1376, 2007.

[4] E. Todorov. Efficient computation of optimal actions. *Proceedings of the national academy of sciences*, 106(28):11478–11483, 2009.

[5] E. Todorov. Eigenfunction approximation methods for linearly-solvable optimal control problems. pages 161–168, March 2009.

A Linear Algebra equation

We note N and T the non-terminal and terminal states. With these notations, if we write G, P and z in the order (N, T), we have :

$$P = \begin{bmatrix} P_{NN} & P_{NT} \\ 0 & I \end{bmatrix}; G = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}; z = \begin{bmatrix} z_N \\ z_T \end{bmatrix} \tag{42}$$

with $D_1 = Diag(\exp(c_{NN}))$ and $D_2 = Diag(\exp(c_{TT}))$

In the terminal states we have $v(x_{TT}) = c(x_{TT})$, i.e. :

$$z_T = \exp(-c_{TT}) \tag{43}$$

We then have :

$$GP = \begin{bmatrix} D_1 P_{NN} & D_1 P_{NT} \\ 0 & I \end{bmatrix} \tag{44}$$

Then, we have the following formula :

$$\begin{bmatrix} z_N \\ z_T \end{bmatrix} = \begin{bmatrix} D_1(P_{NN} z_N + P_{NT} z_T) \\ z_T \end{bmatrix} \tag{45}$$

Then :

$$(D_1^{-1} - P_{NN}) z_N = P_{NT} z_T \tag{46}$$

i.e.

$$(Diag(\exp(c_{NN})) - P_{NN}) z_N = P_{NT} Diag(\exp(-c_{TT})) \tag{47}$$