Temporal Difference Approach in Linearly Solvable Markov Decision Processes

Mauricio Burdelis

Graduate School of Information Science

Nara Institute of Science and Technology

A thesis submitted for the degree of

PhilosophiæDoctor (PhD)

2013 July

1. Reviewer: Prof. Kazushi Ikeda

2. Reviewer: Prof. Tsukasa Ogasawara

3. Reviewer: Assoc. Prof. Tomohiro Shibata

Day of the defense: 2013/08/25

Abstract

The Reinforcement learning (RL) approach to Machine Learning is a technique to learn how to make decisions in order to achieve a desired goal. The model does not include the presence of a supervisor. The agent must learn by trial and error. This is done by taking actions and observing their consequences in the form of a reward (or cost) signal. Such problems are usually formalized as a Markov decision process (MDP). The mathematical framework of MDPs relies on the Bellman equation and is very general, but finding solutions can be inefficient because of the explosion of possible future states.

The framework of linearly solvable Markov decision processes (LMDP) greatly simplifies reinforcement learning. By attending specific conditions the Bellman equation can be made linear, and it becomes possible to obtain solutions more efficiently. However, it is necessary to previously know the passive dynamics of the system (i.e. the behavior of the system in the absence of controls) which is crucial in the model, but unknown in general.

A method to calculate such passive dynamics distribution (by performing continuous embedding of known traditional MDPs) exists, but requires the previous knowledge of all transition distributions and all immediate costs. Those are usually not known beforehand in temporal difference methods. Such methods require the agent to explore the environment and learn by trial-and-error.

Here we propose a method to estimate the passive dynamics and state costs of a given system. As a consequence, such system can then be modeled as an LMDP. The method can also be combined with a temporal difference algorithm of the LMDP framework (called Z learning). This enables the direct application of Z learning without the need for explicit knowledge of

passive dynamics nor state costs beforehand. The only required knowledge about the passive dynamics distribution of the system is which states can and which cannot be visited starting from each state. And the only remaining limitation for the direct application to real problems (with symbolic actions) is the assumption that the agent can impose any desired transition distribution it wants. Such assumption is an important premise of the LMDPs framework.

During the application of the method, new constraints regarding the passive dynamics and state costs are successively incorporated in the model from observed information of immediate costs. The resulting algorithm properly estimates the desirability and optimal cost-to-go functions, as well as the passive dynamics and state costs, when solving the resulting constrained optimization problem. The convergence speed of the new algorithm is not significantly affected when compared to pure Z learning. This represents an important step for direct application of the framework of LMDPs framework in a real temporal difference approach.

Keywords:

Reinforcement learning, Markov decision process, Linearly solvable Markov decision processes

To God, my parents, teachers and friends.

Acknowledgements

I would like to acknowledge the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) and the Brazilian National Council for Scientific and Technological Development (CNPq) for their scholarship programs which have made this work possible.

Contents

List of Figures

List of Algorithms

1

Introduction

1.1 Background

Reinforcement learning is a technique to learn from interactions with the environment, without the presence of a supervisor (1), (2), (3). It is a very versatile and general technique, which can be applied to many different problems. Some examples of applications include: trajectory optimization (4), (5); robotics and control (6), (7), (8); mobile communication (9); image recognition (10), (11), e-commerce (12); and even medical treatment (13).

The model does not include the presence of a supervisor. A feedback signal from the environment is given in the form of a cost signal. In such applications the objective is to find optimal actions to take, so as to reach a given terminal condition while accumulating the minimum cost possible. The decisions of actions to take cannot be based only on the immediate costs, but must take into account all future possibilities. The optimal cost-to-go function is defined as the expected value of the total cumulative cost starting at a state and optimally choosing actions thereafter. Such function can be used for the greedy computation of optimal actions, and it can be estimated or calculated by using the Bellman equation. However, the solution of the Bellman equation in traditional MDPs can be computationally inefficient, because the number of future possibilities grows exponentially with time (1), (14).

Under specific conditions, the Bellman equation of MDPs is made linear, and solutions can be obtained more efficiently than in traditional MDPs. MDPs within this class are called linearly solvable Markov decision processes (LMDPs). This framework

has many benefits, including faster convergence of temporal difference methods, and easier computation of solutions in closed form (15). Another benefit is the possibility of combining existing control policies (16), which has already been used for character control for animation (17). This framework has been proposed after an insight of the duality between action and perception (18), obtained during theoretical and experimental research on human movement control (19), (20). Other applications of the framework of LMDPs include inverse optimal control (21), and estimating trajectories of optimally-controlled stochastic systems (22).

In order to model a given system as an LMDP, it is necessary to know the passive dynamics transition distribution of the system. This distribution corresponds to the behavior of the system in the absence of controls. In many cases this transition distribution is assumed to be "random walk" (equal transition probability to all adjacent states) (14), (23). However, the passive dynamics distribution of an arbitrary system is not known in general.

Z learning is a temporal difference method which can estimate the optimal cost-to-go function, and it takes advantage of the LMDPs framework to achieve faster convergence than traditional methods such as Q learning (1), (14), (23). There exist two ways to apply Z learning: one is by following the passive dynamics as a policy, and the other is by following another policy such as "greedy policy" (the policy which seems optimal given the currently estimated knowledge about the system). In both cases, it is crucial that all states keep being visited for the method to work. In the first case, the passive dynamics needs to be followed but it does not need to be known. However, the passive dynamics distribution might not be a policy which has characteristics of exploration, such as "random walk". In such cases it might be difficult to keep the estimates being updated for all states. In the second case, the passive dynamics needs to be known beforehand for the application of the method. Also, immediate costs in LMDPs are made up of two addends, the state cost and the action cost. If the state costs cannot be measured separately, those need to be known beforehand as well.

The framework of LMDPs has already been criticized for dealing with stochastic optimal control by using discrete action-state spaces, and treating them as MDPs. This would make the framework difficult to apply to high dimensional robotic systems (24). It has also been criticized for imposing strong restrictions on the structure of the cost function, and therefore also restricting the solution to special cases of optimal control

problems, and it is necessary to learn the value function in order to obtain solutions (24). Performance limitations have also been discussed in (25).

However, the fact that the Bellman equation is made linear is a very interesting benefit of LMDPs. Also, the optimal policy can be obtained directly once the desirability function z is obtained (14). Such framework can be especially useful if the Z learning algorithm becomes applicable with less restrictions and in a practical way.

The passive dynamics and state costs can be calculated by using a method for continuous embedding of traditional MDPs (14). Given a traditional MDP, such method determines the passive dynamics and all state costs of an LMDP. This LMDP can then be used to obtain an approximately optimal solution to the traditional MDP. However, in order to use such application all immediate costs of the traditional MDP must be known beforehand. These costs are not necessarily known in temporal difference approaches, where the agent needs to explore the environment in order to observe the immediate costs.

1.2 Motivation

The motivation of the present work is to propose a method to directly apply a temporal difference method (Z learning) in a system modeled as an LMDP, without the previous knowledge of the passive dynamics transition distributions nor state costs. If such method is successfully created, it becomes possible to apply the temporal difference approach in the LMDP framework (Z learning) to a given system without the need for previous knowledge of state costs, immediate costs nor passive dynamics, and without the need to follow the passive dynamics while exploring the environment.

1.3 Outline of the thesis

In this thesis, we propose such method which estimates the passive dynamics and state costs of the system during the execution of Z learning, while also estimating the desirability function (and consequently the optimal cost-to-go function). By exploring the environment, the learning agent observes immediate costs information, which is included in the model as constraints, which are used to update estimates of the passive dynamics distributions and state costs. As long as the correct values are unknown, these

estimates are used in the estimation of the desirability function. As a consequence, there is no need to know all immediate costs beforehand, nor the need to follow the passive dynamics while exploring the environment. This represents an important breakthrough for the application of the framework of LMDPs in a temporal difference approach. In Chapter 2, we review the most important facts about the framework of LMDPs. In Chapter 3 we propose a model of passive dynamics with a simple model of inertia and collisions, for discrete state-space two-dimensional navigation problems. We show experiments in which the Z learning algorithm converges to the correct solution using such passive dynamics distribution in the model. In Chapter 4 we show the main idea of our method to calculate the passive dynamics and state costs by creating and solving systems of linear equations from measurements of immediate costs. In Chapter 5 we show how to integrate such method with Z learning in order to estimate the passive dynamics and state costs while also estimating the desirability and optimal cost-to-go. In Chapter 6 we present concluding remarks.

2

The Framework of Linearly Solvable Markov Decision Processes

The framework of linearly solvable Markov decision processes greatly simplifies reinforcement learning. When specific conditions are met, the Bellman equation is made linear and solutions can be obtained more efficiently. In this section we briefly present the framework of traditional Markov decision process (MDPs), and the framework of linearly solvable Markov decision processes (LMDPs), both formulated for discrete time problems. We also explain how the framework of LMDPs differs from traditional MDPs.

2.1 Traditional Markov Decision Processes

Let us denote the state of the environment in a discrete time instant t as x_t. This state signal is said to have the "Markov property" if it retains all relevant information:

$$\Pr(x_{t+1}|x_t) = \Pr(x_{t+1}|x_t, x_{t-1}...) \tag{2.1}$$

In other words, this means that the history of past states has no influence in the future and can be disregarded. A reinforcement learning task that satisfies the Markov property is called a Markov decision process (MDP) (1).

Problems of choosing optimal actions are often formalized as a finite MDP with: a set of possible states X, a set of possible actions U, state transition probabilities

$p(x'|x,u)$ (from current state x to state x' when action u is taken), and immediate cost $\ell(x,u)$ for being at state x and taking action u.

The goal of the agent is to choose actions in order to make state transitions until a terminal or goal state is reached, with minimum incurred total cumulative cost (1), (26), (14). The total cumulative cost is defined as the sum of all immediate costs incurred from the start until reaching a terminal state. The agent knows what is the current perceived state x and the available actions $u \in U$ available at that state, and once an action u is taken, the immediate incurred cost $\ell(x,u)$ is observed, as well as the newly occupied state x'.

The optimal actions cannot be obtained by greedy optimization of the immediate costs. All future possibilities must also be taken into account. An important quantity for achieving this is the optimal cost-to-go function of a state, denoted $v(x)$. It is defined as the statistical expectation of the total cost that the agent will accumulate starting from that state, and following the optimal policy thereafter. Such function can be used for greedy computation of optimal actions.

The task of calculating or estimating the optimal cost-to-go function v is not a trivial one. What makes this task possible is the Bellman equation (1), (14):

$$v\left(x\right) = \min_{u} \left\{ \ell\left(x,u\right) + \mathrm{E}_{x' \sim p(\cdot|x,u)} \left[v\left(x'\right) \right] \right\} \tag{2.2}$$

where

$$\mathrm{E}_{x' \sim p(\cdot|x,u)} \left[v\left(x'\right) \right] \equiv \sum_{x'} p\left(x'|x,u\right) v\left(x'\right) \tag{2.3}$$

is the statistical expectation of the optimal cost-to-go function at the future state x'. The optimal cost-to-go function is the only solution to its Bellman equation.

If all possible immediate costs $\ell(x,u)$ and transition probabilities $p(x'|x,u)$ are known for all possible current and future states x and x' and actions u, the solution to the Bellman equation can be obtained by Dynamic Programming (DP). However, this can be time consuming due to the explosion of unknown variables. Indeed, the number of future states grows exponentially with time. When all possible immediate costs $\ell(x,u)$ and transition probabilities $p(x'|x,u)$ are not known, temporal difference methods (such as TD learning or Q learning (1), (14), (23)) can be used to estimate the value function by exploring the environment and successively updating estimates. Such

estimates are updated based on the observed immediate costs and previous estimates. In a temporal difference approach, the immediate costs $\ell(x, u)$ and state transition distributions $p(x'|x, u)$ are not known by the agent beforehand, and the agent needs to learn by trial-and-error.

2.2 Linearly Solvable Markov Decision Processes

By attending specific conditions, the Bellman equation of an MDP becomes linear, resulting in a linearly-solvable Markov decision process (LMDP). Its solution can be obtained analytically as a solution to an eigenvector problem. The LMDPs framework also has a temporal difference algorithm (called Z learning), which is more efficient than traditional temporal difference methods (such as Q learning and SARSA). We review these facts according to (14) (23).

2.2.1 Sufficient Conditions for Obtaining an LMDP

In the original MDP framework, the agent takes symbolic actions $u \epsilon U$. The probabilities of state transitions from the current state $x \epsilon X$ to a future state $x'\epsilon X$ depend on the action u taken at state x (denoted as $p(x'|x, u)$).

One condition to obtain an LMDP is that the agent needs to be able to specify the transition probabilities directly. In other words, there are no symbolic actions u nor the set of actions U in the model. Instead, the agent can directly specify the probability of transition from the current state $x \epsilon X$ to any possible future state $x' \epsilon X$. These probabilities will be represented by using the letter u. Controlled transition distributions will simply be represented as $u(x'|x)$.

Another condition is that the immediate cost incurred in a state transition must be of the form:

$$\ell(x, u) = q(x) + \mathrm{KL}(u(.|x) \| p_d(.|x)) \tag{2.4}$$

where $q(x)$ is called "state cost" (which depends only on the argument state, and indicates how undesirable a given state is), and

$$\mathrm{KL}(u(\cdot|x) \| p_d(\cdot|x)) \equiv \mathrm{E}_{x' \sim u(\cdot|x)} \left[\log \frac{u(x'|x)}{p_d(x'|x)} \right] \tag{2.5}$$

is the Kullback Leibler (KL) divergence between the controlled state transition distribution $u(x'|x)$ and the "passive dynamics" $p_d(x'|x)$ of the system. The latter is a transition distribution that corresponds to the behavior of the system in the absence of controls. The KL divergence measures how "different" these distributions are from each another, and is called "action cost". It represents the fact that the more different the desired behavior is from the default behavior (or the more the controlled dynamics are "pushing" the system away from its default behavior), the higher the cost will be. As a consequence of the definition of the action cost, $u(x'|x) = 0$ is required whenever $p_d(x'|x) = 0$ in order to keep the KL divergence well-defined. This constraint also adds the benefit of avoiding impossible state transitions (such as a sudden "jump" from any current state to a goal state).

2.2.2 The Linear Bellman Equation

When the previously mentioned conditions are met, it can be shown that the Bellman equation (2.2) can be reduced to:

$$z(x) = \exp\left(-q\left(x\right)\right) \mathcal{G}\left[z\right]\left(x\right) \tag{2.6}$$

where

$$z\left(x\right) \equiv \exp\left(-v\left(x\right)\right) \tag{2.7}$$

is called the "desirability function" of a state, and

$$\mathcal{G}\left[z\right]\left(x\right) \equiv \sum_{x'} p_d\left(x'|x\right) z\left(x'\right) \tag{2.8}$$

is a linear operator, which corresponds to the expected desirability of the future state when following the passive dynamics. It is important to notice that this new equation (2.6) is linear in z. It can then be shown that the optimal controlled transition distribution u^* is then given by:

$$u^*\left(x\right) = \frac{p_d\left(x'|x\right) z\left(x'\right)}{\sum_{x'}\left[p_d\left(x'|x\right) z\left(x'\right)\right]} \tag{2.9}$$

When the passive dynamics distribution $p_d(x'|x)$ and state costs $q(x)$ for all possible states are known, then the optimal cost-to-go function for all states can be obtained by solving the Bellman equation. In traditional MDPs this can be done with dynamic

programming. In the framework of LMDPs this problem becomes an eigenvetor problem, and therefore is simpler to solve. Equation (2.6) can be written in vector notation. This is done by enumerating the states from 1 to n, representing $z(x)$ and $q(x)$ as column vectors $\mathbf{z}$ and $\mathbf{q}$, and representing $p(x'|x)$ as a matrix P (where the row-index corresponds to the current state x and the column-index corresponds to future state x'). Then the linear Bellman equation (2.6) becomes:

$$\mathbf{z} = M\mathbf{z} \tag{2.10}$$

where

$$M = \text{diag}\left(\exp\left(-\mathbf{q}\right)\right) P \tag{2.11}$$

where "diag" transforms vectors into diagonal matrices. By solving this eigenvector problem the desirability function $z(x)$ can be obtained for all states, and consequently the optimal cost-to-go function v (equation 2.7). This is an efficient method, but its application depends on the previous knowledge of all state costs, as well as the complete knowledge of the passive dynamics of the system. The LMDPs class has also been extended to work with Rényi divergences, a more general class of divergences of which the KL divergences are a special case (27). By changing the value of the continuous parameter α the behavior of the system can also be tuned to be more risk averse ($\alpha > 0$) or risk seeking ($\alpha < 0$). In the present work we will only use KL divergences, which are the case of Rényi divergences in which $\alpha \to 0$.

2.2.3 Z Learning

When the state costs and passive dynamics are not all necessarily known, the agent must explore the environment in order to observe unknown costs, and learn by trial-and-error. The optimal cost-to-go is then estimated by a temporal difference method. Z learning is a temporal difference method which takes advantage of the class of LMDPs to achieve faster convergence than traditional reinforcement learning methods (such as Q learning (1)). As in traditional temporal difference methods, initial estimates are successively updated until convergence is achieved. By estimating the desirability function z, the optimal cost-to-go v can be obtained (equation 2.7).

If the agent is simply following the passive dynamics transition distribution during the execution of the algorithm, the Z learning update formula is as follows:

$$z_{\text{new}}(x_t) \leftarrow (1-\eta_t)\, z_{\text{cur}}(x_t) + \eta_t \exp(-q_t)\, z_{\text{cur}}(x_{t+1}) \tag{2.12}$$

where $z_{\text{new}}(x_t)$ is the new estimate of the desirability z at the current state x_t, $z_{\text{cur}}(x_t)$ is the current estimate of $z(x_t)$, $z_{\text{cur}}(x_{t+1})$ is the current estimate of the desirability z at the future state x_{t+1}, q_t is the state cost of the current state x_t, and η_t is a learning rate which decreases over time.

For the application of this kind of Z learning the passive dynamics transition probabilities do not need to be known, but the agent needs to follow the passive dynamics during the process. The KL divergence will always be null, and all observed costs will correspond to the state costs q. Therefore, there is no need to know these costs beforehand. For such a method to work, all states must keep being visited, and desirability $z(x)$ estimates must keep being updated for all x. If the passive dynamics distribution is "random walk" (i.e. equal transition probabilities to all adjacent states) or another distribution with characteristic of exploration, then the application of such a method should not be a problem. However, this method might be difficult to apply in practice if the passive dynamics distribution does not explore the environment.

It is possible to apply Z learning following a controlled transition distribution $\hat{u}$ (i.e. different than passive dynamics), but then importance sampling is required: the last term in equation (2.12) needs to be multiplied by $p_d(x_{t+1}|x_t)/\hat{u}(x_{t+1}|x_t)$. In the case of "greedy Z learning", $\hat{u}$ is the policy which appears optimal given the current estimates of the desirability function $\hat{z}$ (according to equation (3.2)). It is important to notice that, in order to apply this method, it is necessary to know the passive dynamics distribution beforehand. Also, the measured immediate costs ℓ in the process will no longer correspond to the state costs q (because the action costs will no longer be null), so the state costs $q(x)$ must be known for all states x beforehand, or those must be measured or estimated separately.

2.3 Final Considerations of This Chapter

The framework of LMDPs has many advantages to traditional MDPs because the Bellman equation becomes linear. However, its direct application in realistic problems faces

some limitations. This is especially true for temporal difference applications. The assumption that the agent can impose any desired transition distribution is very strong, as well as the assumption that the passive dynamics transition distributions are known.

If all immediate costs and transition distributions of a traditional MDP are known, it can be modelled as an LMDP. A method for continuous embedding of traditional MDPs exists, and the approximation has been found to be very accurate in practice (14). However, in temporal difference methods the immediate costs and transition distribution are not all previously known in general.

The proposal of methods to deal with the assumption that the agent can impose any desired transition distribution remains to be developed. The main contribution of this thesis is the proposal of a method to estimate the passive dynamics and state costs during the execution of Z learning (the temporal difference method of the framework of LMDPs). As a consequence Z learning can be applied by following a controlled policy without the need for knowing all immediate costs nor passive dynamics transition probabilities beforehand.

In the next chapter we propose a simple model of passive dynamics distributions which is different than random walk, and supports inertia and collisions. Such passive dynamics distributions are modelled in a two-dimensional grid world. We also show results of experiments in which Z learning converges to the correct solutions using this passive dynamics model instead of random walk.

3

Modeling Inertia and Collisions in the Passive Dynamics Distribution

Previous work has shown results of temporal difference learning experiments, using passive dynamics distributions which corresponded to "random walk" (23) (14). This distribution corresponds to equal probability of transition to all possible future states. However, in the LMDPs model there are no theoretical limitations to the passive dynamics distributions. Therefore, in this section we present a different passive dynamics model, which includes a simple model of inertia and collisions. Such model is more realistic than "random walk", and it also has less characteristic of exploration. We then perform experiments with the proposed environment, and observe that the temporal difference learning algorithm of the LMDPs framework (Z learning) still converges when using the proposed passive dynamics model. The main concept and related computational experiments can be found in (5).

Let us consider a two-dimensional grid world navigation problem, such as the one in figure 3.1. The goal is to find the goal position while avoiding the obstacles along the way.

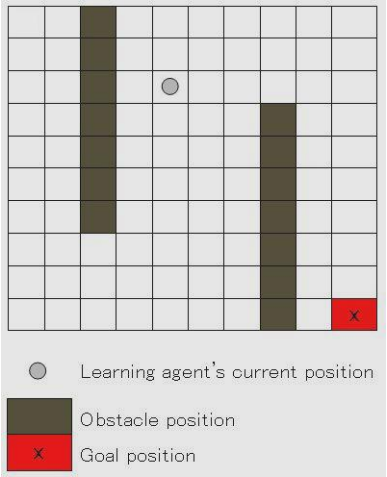


Figure 3.1: A 10×10 two-dimensional grid world.

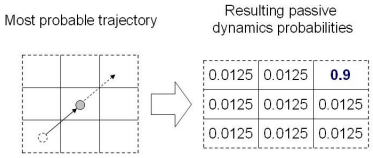


Figure 3.2: A simple inertia model ($hp = 0.9$).

3.1 The Passive Dynamics Model

3.1.1 Modeling States as Position Pairs

Initially, a few difficulties have emerged when modeling the state signal in the grid world. If we the state is modeled simply as the position of the learning agent, then in order to model inertia, it would at least be necessary to know the current and the previous states of the agent. This would violate the "Markov property" . Also, in order to model collisions with obstacles and walls, the agent would originally need to be able to take symbolic actions of movement in the direction of the wall or obstacle, but there are no symbolic actions in the framework of LMDPs.

In order to overcome the previously mentioned difficulties, the state signal was modeled to include information of position pairs: the current and the previous position of the agent.

A simple model of inertia was created by assigning a "highest probability" value (denoted hp) to the state which is in the same trajectory as the last transition (i.e.

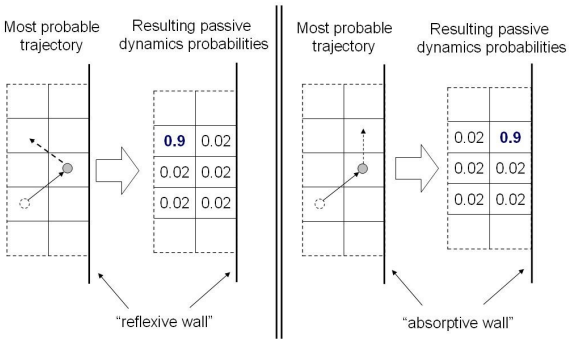


Figure 3.3: A simple model for collisions, with a reflexive wall and with an absorptive wall ($hp = 0.9$).

the transition from the previous position to the current position). The remaining $(1 - hp)$ probability is equally shared by the remaining adjacent positions. This model is illustrated in figure 3.2.

When the position of the agent is adjacent to a wall or obstacle, and the trajectory of the last transition goes in the direction of the wall or obstacle, then the position that receives the highest probability value "hp" is the most likely position after a collision with this wall or obstacle, and the remaining $(1 - hp)$ probability is equally shared by the remaining adjacent states. Two types of collisions were modeled: reflexive collisions (which reflect the component which is normal to the wall or obstacle), and absorptive collisions (which absorb the component which is normal to the wall). This model is illustrated in figure 3.3.

3.2 Experiments with the Environment

Experiments were performed to observe the convergence of Z learning in the grid world environment illustrated in figure 3.1, when the passive dynamics of the system differs from random walk, but instead corresponds to our model (which includes inertia and collisions).

Since the original formulation of Z learning (23), (14) does not require that the passive dynamics distribution be equal to "random walk", it is expected that the approximation errors of the optimal cost-to-go function estimates $\hat{v}$ consistently decrease as the simulation steps increase.

3. MODELING INERTIA AND COLLISIONS IN THE PASSIVE DYNAMICS DISTRIBUTION

In the experiments, at each time step the agent can only move to an adjacent position to the current position. The agent is not allowed to move to an obstacle position. Every state x_{ng} which has a current position different to the goal position p_g was modeled to have state cost $q(x_{\text{ng}}) = 1$. Every state x_g which has current position equal to the goal position was modeled to have state cost $q(x_g) = 0$. In all experiments $hp = 0.9$.

Every experiment consisted of episodes. When an episode starts, the agents occupies a random initial position p_i, which is different from the goal position. The initial state corresponds to the state which has the current and previous positions equal to p_i. When the agent reaches the goal position, the current state x_g has current position equal to x_g, but previous position different from x_g. The agent then automatically stops at the goal position, so at the next time step the current state will be a state in which both the current and previous position correspond to p_g, denoted x_G. A new episode automatically begins at the next time step. Each experiment lasted for a predefined number of time steps.

The problem of obtaining the optimal cost-to-go function v of the above described environment was first solved both analytically, using the linear Bellman equation (equation 2.6), before applying Z learning. Therefore, the correct values of v are previously known before the execution of Z learning, and the quality of the current estimates of v obtained by Z learning can be measured by comparing those estimates with the correct values. The error of the v estimates is calculated at each step using the formula:

$$\text{Error of } \hat{v} = \frac{\sum_{i=1}^{N} |\hat{v}(x_i) - v^*(x_i)|}{\sum_{i=1}^{N} v^*(x_i)} \tag{3.1}$$

where x_i represents the i^{th} state; N is the total number of states; $\hat{v}(x)$ is the current approximation of the optimal cost-to-go function (obtained by taking $-\log(\hat{z}(x))$ where $\hat{z}$ is the current estimate of the desirability function) and $v^*(x)$ is the optimal cost-to-go function obtained analytically.

Two different passive dynamics distributions were used in the experiments. One corresponded to a reflexive environment, and the other corresponded to an absorptive environment. These environments differ with respect to their passive dynamics distributions regarding collisions (as shown previously).

During the experiments, two types of Z learning were used. In one, the agent explores the environment by following the passive dynamics. In the other, the agent

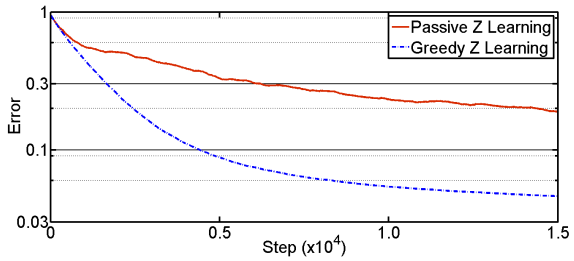


Figure 3.4: Z learning approximation errors: estimated optimal cost-to-go v. The passive dynamics corresponds to reflexive walls and obstacles (vertical axis in log scale).

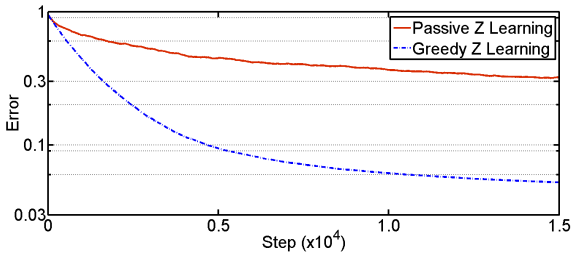


Figure 3.5: Z learning approximation errors: estimated optimal cost-to-go v. The passive dynamics corresponds to absorptive walls and obstacles (vertical axis in log scale).

explores the environment by following the greedy policy (14). Such policy is defined as the policy which appears optimal given the current estimates of the desirability function $\hat{z}$, and can be calculated as:

$$\hat{u}^* \left(x'|x\right) = \frac{p_d \left(x'|x\right) \hat{z} \left(x'\right)}{\sum_{x'} \left[p_d \left(x'|x\right) \hat{z} \left(x'\right)\right]} \tag{3.2}$$

where $\hat{u}^* \left(x'|x\right)$ represents the probability to go to state x' from state x according to the greedy policy. As seen in (23), (14), it is expected that Z learning following the greedy policy converges faster than when following the passive dynamics.

In all simulations the learning rate η_t decays obeying $\eta_t = c/(c+t)$ where t is the current simulation step and c is a constant value (for "greedy Z learning" $c = 10,000$ was used, and for "passive Z learning" $c = 30,000$ was used).

The results of the experiments can be seen in figures 3.4 and 3.5. As expected, the errors of the estimates of the optimal cost to go function $\hat{v}$ consistently reduce as the simulation proceeds. In previous related work (23) (14) the convergence is slower when following the passive dynamics then when following the greedy policy. In our experiments the same fact could be observed.

It can be observed that the framework of LMDPs and Z learning are consistent for working with different passive dynamics distributions, which can be different than "random walk".

3.3 Final Considerations of This Chapter

In this chapter we have proposed a passive dynamics distribution which includes a simple model of inertia and collisions. Such model is more realistic than "random walk", but has less characteristic of exploration.

We have presented experiments which have shown that the Z learning method can be used with such passive dynamics distribution, and converges to the correct values.

In the next chapters we will discuss the main idea of our method to calculate passive dynamics probabilities and states costs from measured costs. After that we will show how this method can be integrated with Z learning.

4

Calculating the passive dynamics distribution and state costs from measured costs

In this chapter we propose a method to calculate the passive dynamics of a given system from measurements of action costs. Later, the method is extended in order to calculate both the passive dynamics and state costs of a given system from measured immediate costs.

In our proposed approach, every possible transition starting from each state to each possible future state is considered as an unknown variable.

Linear equations are created and gathered from measurements of the action cost or the immediate cost, using the action cost equation or the immediate cost equation. Such systems are then solved and the correct values of the passive dynamics transition probabilities are calculated. The main concept and related computational experiments can be found in (28), (29), (30).

4.1 Calculating the Passive Dynamics from Measured Action Costs

Let us consider a discrete state space with cardinality $|X| = N$ (i.e. there are N possible states), such as the one represented in figure 4.1. Consider a given arbitrary transition distribution u_1 which is followed, and its transition probabilities $u_1(x_j|x_i)$ are all known

4. CALCULATING THE PASSIVE DYNAMICS DISTRIBUTION AND STATE COSTS FROM MEASURED COSTS

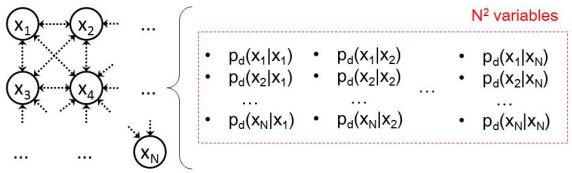


Figure 4.1: A state space with N possible states (and consequently N^2 possible transitions.)

for every possible i and j. No restrictions are assumed except that for all cases in which $p_d(x_j|x_i) = 0$ $(i, j \,\epsilon\, \mathbf{N} | 1 \leq i \leq N,\ 1 \leq j \leq N)$ then $u_1(x_j|x_i) = 0$ as well. Let us denote $L_a(u_1, x_i)$ the action cost incurred when following the controlled transition distribution u_1 from an arbitrary state x_i. Such cost is given by the rightmost addend of equation (2.4), which is the KL divergence from the controlled dynamics distribution $u_1(\cdot|x_i)$ to the passive dynamics distribution $p_d(\cdot|x_i)$, both starting from state x_i. It can be written as:

$$\begin{aligned} L_a\left(u_1, x_i\right) &= \mathrm{KL}\left(u_1\left(\cdot|x_i\right) \| p_d\left(\cdot|x_i\right)\right) \\ &= u_1\left(x_1|x_i\right) \log\left(\frac{u_1\left(x_1|x_i\right)}{p_d\left(x_1|x_i\right)}\right) + \cdots \\ &+ u_1\left(x_N|x_i\right) \log\left(\frac{u_1\left(x_N|x_i\right)}{p_d\left(x_N|x_i\right)}\right) \end{aligned} \tag{4.1}$$

Let us assume that the action cost $L_a(u_1, x_i)$ can be measured, and such measured value is denoted as $A_{1\mathrm{i}}$. If we apply the logarithm quotient identity $\frac{\log(a)}{\log(b)} = \log(a) - \log(b)$ to equation (4.1), we have a linear equation in the variables $\log\left(p_d(x_j|x_i)\right)$:

$$\begin{aligned} A_{1\mathrm{i}} &= u_1\left(x_1|x_i\right) \log u_1\left(x_1|x_i\right) - u_1\left(x_1|x_i\right) \log p_d\left(x_1|x_i\right) \\ &+ \cdots + u_1\left(x_N|x_i\right) \log u_1\left(x_N|x_i\right) - u_1\left(x_N|x_i\right) \log p_d\left(x_N|x_i\right) \end{aligned} \tag{4.2}$$

Rearranging the terms:

$$\sum_{j=1}^{N} u_1(x_j|x_i) \log p_d(x_j|x_i) = \sum_{j=1}^{N} u_1(x_j|x_i) \log u_1(x_j|x_i) - A_{1\mathrm{i}} \tag{4.3}$$

it is possible to produce up to N equations by using the controlled distribution u_1, starting from each of the N possible existing states x_i and measuring the action cost $L_a(u_1, x_i)$ incurred when making a transition.

Repeating the procedure for N different arbitrary controlled distributions u_k ($k \epsilon \mathbf{N} | 1 \leq k \leq N$), it is possible to produce N^2 different equations and the system of linear equations can finally be solved.

So far the above formulation is general enough to be applied to any reinforcement learning system with discrete time and discrete state space, but the number of necessary equations seems to be considerably large. However, when considering real physical systems it is not common to have the possibility to make transitions to any state $x_j \epsilon X$ from any state $x_j \epsilon X$. Real physical systems have additional constraints that can greatly reduce the number of necessary equations. For example, consider a two-dimensional grid world (such as the one treated in chapter 3, figure 3.1) in which only transitions to adjacent states are possible. In such an environment the passive dynamics transition probability from a given state $x_i \epsilon X$ to any non-adjacent state $x_j \epsilon X$ would necessarily be $p_d(x_j|x_i) = 0$. As a consequence, starting for each state the number of unknowns (and consequently the number of equations to be gathered) is only the number of adjacent states N_{adj}, and therefore is greatly reduced. In a two-dimensional grid world the maximum possible number of adjacent states to any given state is $N_{\text{adj}} = 9$ (also counting transitions from the state to itself). The resulting algorithm is illustrated in algorithm 1.

Algorithm 1 Calculating the passive dynamics distributions for all states

for each state $x_i \epsilon X$ **do**
  $N_{\text{adj}} \leftarrow$ number of adjacent states to x_i
  for $n \leftarrow 1$ until $n = N_{\text{adj}}$ **do**
    produce an arbitrary transition distribution $u_n(\cdot|x_i)$
    (must satisfy $u_n \neq u_m$, $\forall m < n$)
    follow u_n from state x_i
    $A_{\text{ni}} \leftarrow$ measured action cost
    gather one equation for the system of linear equations to determine $p_d(\cdot|x_i)$

$$A_{\text{ni}} = u_n(x_1|x_i) \log u_n(x_1|x_i) - u_n(x_1|x_i) \log p_d(x_1|x_i) + \cdots + u_n(x_{N\text{adj}}|x_i) \log u_n(x_{N\text{adj}}|x_i) - u_n(x_{N\text{adj}}|x_i) \log p_d(x_{N\text{adj}}|x_i)$$

  end for
  solve the obtained system of linear equations and obtain $p_d(\cdot|x_i)$
end for

4.2 Calculating the Passive Dynamics and State Costs from Measured Total Costs

The method proposed so far can only calculate the passive dynamics when the action costs can be observed. However, in real applications it might not be possible to observe state costs and action costs separately. It might be a more realistic approach to consider measured immediate costs ℓ for the calculation of the passive dynamics distribution. To extend our method to work with immediate costs ℓ, we subtract the state cost q on both sides of equation 4.2, which becomes a new unknown in the system. By using the fact that $\ell(x_i) = q(x_i) + KL\left(u_1(\cdot|x_i)|p_d(\cdot|x_i)\right)$ (equation 2.4), we have:

$$\sum_{j=1}^{N} u_1\left(x_j|x_i\right) \log p_d\left(x_j|x_i\right) - q\left(x_i\right) = \sum_{j=1}^{N} u_1\left(x_j|x_i\right) \log u_1\left(x_j|x_i\right) - T_{1\mathrm{i}} \tag{4.4}$$

Where $T_{1\mathrm{i}}$ denotes the measured immediate cost $\ell(x_i, u_1)$ incurred when following the controlled transition distribution u_1 from state x_i. For a given state x_i we have $N_{\mathrm{adj}}+1$ variables to determine in order to calculate the passive dynamics distribution $p_d(\cdot|x_i)$ and the state cost $q(x_i)$.

Unfortunately, in all experiments we performed with this method, the obtained system did not turn out to be a system with one single solution (i.e. the obtained system had infinite solutions). This fact happened even when we obtained a number of equations equal to the number of variables $N_{\mathrm{adj}} + 1$ (by following $N_{\mathrm{adj}} + 1$ different arbitrary transition distributions from state x_i).

However, this problem can be solved by adding the constraint that the sum of passive dynamics transition probabilities needs to be equal to one. We propose two ways to do this: one is an iterative method using a modified gradient descent algorithm, and another is a non-iterative method using a clever variable substitution. For both methods only N_{adj} different equations are necessary.

The iterative method consists in using the gradient descent algorithm to solve the system of linear equations. At each step of the algorithm, we normalize the current estimated passive dynamics probabilities to sum up to 1. This algorithm is illustrated in algorithm 2, where the system is written in vector notation (in the form $A\mathbf{x} = \mathbf{b}$). The matrix A corresponds to the matrix of coefficients of the unknowns, and the column vectors $\mathbf{x}$ and $\mathbf{b}$ are the vector of unknowns and a vector of constants, respectively. The

Algorithm 2 Gradient Descent (with Probability Normalization)

1: get an initial solution using the Moore-Penrose pseudoinverse of A
   $\mathbf{x} \leftarrow A^{+} \cdot \mathbf{b}$
2: Normalize the probabilities to sum up to 1
   For all j: $p_d\left(x_j|x_i\right) \leftarrow \frac{p_d(x_j|x_i)}{\sum_{j=1}^{N\text{adj}} p_d(x_j|x_i)}$
3: **repeat**
4: Take a step of the gradient descent algorithm
   $\mathbf{x} \leftarrow \mathbf{x} + \frac{1}{\gamma} A^T(\mathbf{b} - A.\mathbf{x})$ *where* $\gamma > 1$
5: Normalize the probabilities to sum up to 1
   For all j: $p_d\left(x_j|x_i\right) \leftarrow \frac{p_d(x_j|x_i)}{\sum_{j=1}^{N\text{adj}} p_d(x_j|x_i)}$
6: **until** convergence

other method is mathematically similar to a method used for continuous embedding of traditional MDPs with symbolic actions (14). The system can be solved by using a clever variable substitution. Let us consider the system written in vector notation as with the previous method (i.e. $A\mathbf{x} = \mathbf{b}$). However, the variable $q(x_i)$ is now separated from the vector of variables $\mathbf{x}$. This vector will now only contain variables corresponding to logarithms of passive dynamics transition probabilities. The system then becomes $q\mathbf{1} - A\mathbf{x} = \mathbf{b}$, where $\mathbf{1}$ is a column vector of ones, and q is the unknown state cost $q(x_i)$. By observing that A is a stochastic matrix ($A\mathbf{1} = \mathbf{1}$), the system can be rewritten as $A\left(q\mathbf{1} - \mathbf{x}\right) = \mathbf{b}$. The method to solve such system while forcing the sum of passive dynamics transition probabilities to sum up to one is described in algorithm 3. The advantage of this method is that it is not iterative, and can be used to calculate the correct solution to the obtained linear system directly. The total algorithm for calculating the passive dynamics distributions and state costs for all states from measured immediate costs can be seen in algorithm 4.

4.3 Computational Experiments to Validate the Proposed Methods

Experiments were performed to validate the proposed methods for calculating the passive dynamics distribution p_d from measured action costs (section 4.1), by following the algorithm 1, and for calculating the passive dynamics p_d and state costs q from

4. CALCULATING THE PASSIVE DYNAMICS DISTRIBUTION AND STATE COSTS FROM MEASURED COSTS

Algorithm 3 Variable Substitution Algorithm

change of variables: $(q\mathbf{1} - \mathbf{x}) = \mathbf{c}$

solve $A\mathbf{c} = \mathbf{b}$ for $\mathbf{c}$

now we need to determine the value of q
$q \leftarrow -\log\left(\sum_{x_i} \exp\left(-c_{\mathrm{xi}}\right)\right)$
(where x_{xi} and c_{xi} correspond to elements of the vectors x and c)

$\mathbf{x}$ needs to satisfy $\sum_{x_i} \exp\left(x_{\mathrm{xi}}\right) = 1$:
solve $(q\mathbf{1} - \mathbf{x}) = \mathbf{c}$ for $\mathbf{x}$

Algorithm 4 Calculating the passive dynamics distributions and state costs for all states

for each state $x_i \, \epsilon \, X$ **do**

  $N_{\mathrm{adj}} \leftarrow$ number of adjacent states to x_i

  for $n \leftarrow 1$ until $n = N_{\mathrm{adj}}$ **do**

    produce an arbitrary transition distribution $u_n(\cdot|x_i)$
    (must satisfy $u_n \neq u_m$, $\forall m < n$)

    follow u_n from state x_i

    $T_{\mathrm{ni}} \leftarrow$ measured immediate cost

    gather one equation for the system of linear equations to determine $p_d(\cdot|x_i)$

$$T_{\mathrm{ni}} = u_n\left(x_1|x_i\right)\log u_n\left(x_1|x_i\right) - u_n\left(x_1|x_i\right)\log p_d\left(x_1|x_i\right)$$
$$+ \cdots + u_n\left(x_{N\mathrm{adj}}|x_i\right)\log u_n\left(x_{N\mathrm{adj}}|x_i\right) - u_n\left(x_{N\mathrm{adj}}|x_i\right)\log p_d\left(x_{N\mathrm{adj}}|x_i\right) + q(x_i)$$

  end for

  solve the obtained system of linear equations using algorithm 2 or algorithm 3 and obtain $p_d(\cdot|x_i)$ and $q(x_i)$

end for

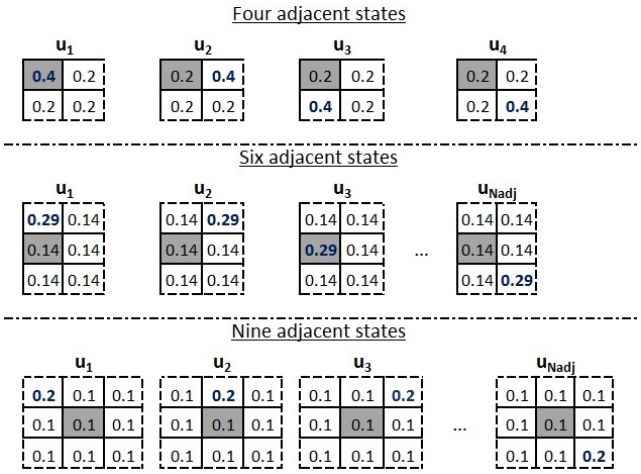
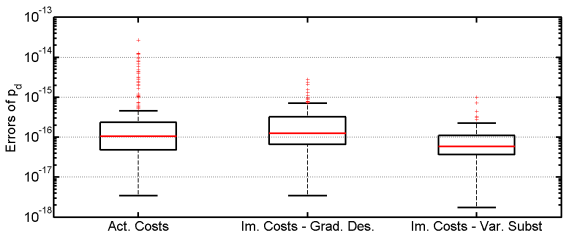


Figure 4.2: Illustration of the generated arbitrary controlled distributions for the computational experiments, for states which have four, six and nine adjacent states. Each controlled distribution has one specific state which has the double of transition probability than others (the occupied position is represented in gray color).

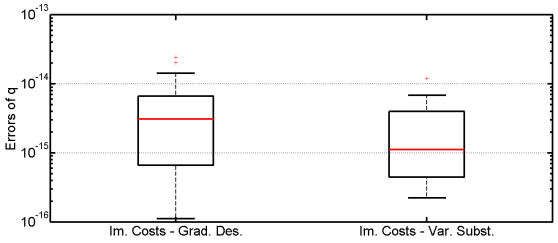
measured immediate costs ℓ (section 4.2) by following the algorithm 4. For these experiments, the same simple 10x10 size grid world (illustrated in figure 3.1) and the same models of passive dynamics (described in chapter 3) were considered. The same experimental environment of chapter 3 was used, with the same grid world (figure 3.1) and the same passive dynamics model with inertia and collisions ($hp = 0.9$).

The agent was placed occupying each and every possible state, one at a time. For each state, arbitrary controlled transition distributions were generated in equal number to adjacent states N_{adj}. Each controlled distribution had a different state as the state with biggest transition probability. All remaining adjacent states shared an equal transition probability. In all experiments the biggest transition probability corresponded to the double of each of the others. In other words: if there are N_{adj} adjacent states to the current state x_{cur}, denoted $x_1,\ x_2,\ \ldots\ ,\ x_{\text{Nadj}}$, then each one of the N_{adj} different controlled transition distributions u_{i} $(i\ \epsilon\ \mathbf{N} \mid 1 \leq i \leq N_{\text{adj}})$ will have a different state x_{m} $(m\ \epsilon\ \mathbf{N} \mid 1 \leq m \leq N_{\text{adj}})$ with the highest transition probability $(u_{\text{i}}\left(x_{\text{m}}|x_{\text{cur}}\right) = 2 \cdot u_{\text{i}}\left(x_{\text{i}}|x_{\text{cur}}\right), \forall i \mid i \neq m)$. This is illustrated in figure 4.2.

In the experiments to test the method for calculating the passive dynamics distribution p_d from measured action costs (section 4.1), the action cost incurred when following each of the controlled transition distributions in each of the states was measured. The



(a) Errors of the estimated p_d: estimation from action costs, from immediate costs with gradient descent method, and from immediate costs with variable substitution method.



(b) Errors of the estimated q: estimation from immediate costs with gradient descent method, and from immediate costs with variable substitution method.

Figure 4.3: Box plots of the errors of the estimated values of passive dynamics and state costs.

corresponding system of linear equations for each state was created (each equation is of the same form as equation (4.3)), and all obtained systems were solved. The calculated passive dynamics was correct, with the same transition probabilities as the original one. In the experiments to test the method for calculating the passive dynamics p_d and state costs q from measured immediate costs ℓ (section 4.2), the same procedure was repeated, however the immediate costs ℓ were measured instead of the action costs. The state costs were also unknowns in the system, as shown in equation (4.4). As in the previous case, each controlled distribution had a different state as the state with biggest transition probability (figure 4.2). Systems of linear equations corresponding to each state were created and solved, both by using the gradient descent with probability normalization method (algorithm 2) and and by using the variable substitution method (algorithm 3).

A box plot of the resulting errors in each case can be seen in figure 4.3. Those errors

were obtained by taking the absolute difference between each transition probability of the correct passive dynamics p_d^* (previously known) and its corresponding estimate $\hat{p_d}$. The errors of the state costs estimates q were obtained in the same manner (absolute difference between the correct q^* and its corresponding estimate $\hat{q}$). In all cases the errors are very small and are within the numerical precision of the simulator software.

These results show that our method can calculate the passive dynamics p_d and state costs q correctly, as expected. In the case of estimation from immediate costs, both proposed methods for solving the constrained system of linear equations, gradient descent (algorithm 2) and variable substitution (algorithm 3) were successful. When using the gradient descent with probability normalization method, it was observed that in all cases the correct passive dynamics was obtained right after taking the initial solution (using the Moore-Penrose pseudo inverse matrix), and normalizing the resulting probabilities to sum up to one (i.e. after step 3 is taken for the first time in algorithm 2). The iterative algorithm was only used to obtain the correct state costs q. The variable substitution method obtained similar results.

4.4 Final Considerations of This Chapter

In this chapter we have proposed a method to calculate the passive dynamics distribution of a system by applying different controlled transition distributions and measuring incurred costs. The main idea behind the method is to create and solve systems of linear equations by using the KL divergence equation (equation 2.5).

This method was then extended to calculate both the passive dynamics and state costs from measured immediate costs. Such extension is important especially when it is not possible to observe action costs separately, and state costs are also unknown.

We have presented computational experiments which show that this method successfully calculates the passive dynamics, and also state costs when applicable.

In the next chapter we will explain how this method can be extended to estimate the passive dynamics and state costs during Z learning execution.

5

Passive Dynamics Distribution Estimation During Z Learning

This chapter describes the main contribution of this thesis. We propose a method to execute Z learning in a real temporal difference approach (i.e. without the previous knowledge of immediate costs of the system), following a controlled transition distribution, and without the previous knowledge of the passive dynamics distribution nor state costs. The method proposed in chapter 4 is used to estimate the passive dynamics distribution p_d and state costs q during the execution of the Z learning algorithm.

Let us consider that the passive dynamics of the system p_d and state costs q are unknown. The only necessary previous knowledge is which transitions are allowed starting from each state (so that $p_d = 0$ is assumed for impossible transitions).

5.1 Gathering Equations for Passive Dynamics and State Costs Estimation During Z Learning

Let us consider that a Z learning algorithm will be performed in the considered system, in which the agent keeps exploring the environment by following a controlled transition distribution. Let us also assume that the controlled distributions starting at each state are not always the same, but actually vary over time. This can be obtained by following the "greedy policy", i.e. the controlled transition distribution which appears optimal given the current estimates of the desirability function, according to equation (3.2). Every time the agent passes by a given state, the controlled transition distribution

will be different (because the desirability function estimates are updated with each visit), and therefore a new equation for calculating the passive dynamics distribution p_d and state costs q can be gathered from the measured cost ℓ. The obtained equation corresponds to equation (4.4).

After a new equation is gathered, one solution to the incomplete system is found. This is done even if only one equation was gathered so far. The obtained solution must respect the constraint that the sum of passive dynamics probabilities equals to one. Even if the solution is not unique (because we still do not have enough equations), an arbitrary solution which respects the equations and constraint is obtained. This solution then replaces the current estimates of the passive dynamics distribution p_d and state cost q starting from that state. This procedure is repeated every time the state is visited, until the necessary number of equations is gathered. This number corresponds to the number of adjacent states N_{adj}. Once this happens, the correct values of the passive dynamics p_d and state costs q are calculated for the given state, using the same procedure as previously described (chapter 4). This is illustrated in algorithm 5. $N_{\text{eq}}(x_i)$ corresponds to the number of gathered equations for state x_i, and $N_{\text{adj}}(x_i)$ corresponds to the number of adjacent states to state x_i.

The estimation of the desirability z (and consequently the optimal cost-to-go v), is performed as follows: when a transition occurs from a given state x_{t} to a future state $x_{\text{t+1}}$, the current estimate of the desirability $z_{\text{cur}}(x_{\text{t}})$ is updated by following the same formula as in "greedy Z learning", but the current estimates of the passive dynamics and state costs are used instead of the correct values:

$$z_{\text{new}}(x_{\text{t}}) \leftarrow (1-\eta_{\text{t}})\, z_{\text{cur}}(x_{\text{t}}) + \eta_{\text{t}} \exp\left(-\hat{q}(x_{\text{t}})\right) z_{\text{cur}}(x_{\text{t+1}}) \frac{\hat{p_d}(x_{\text{t+1}}|x_{\text{t}})}{\hat{u}(x_{\text{t+1}}|x_{\text{t}})} \tag{5.1}$$

where $\hat{p_d}(x_{\text{t+1}}|x_{\text{t}})$ is the current estimate of the passive dynamics probability of transition from the current state x_{t} to the future state $x_{\text{t+1}}$, and $\hat{q}(x_{\text{t}})$, the current estimate of the state cost of state x_{t}.

By following this method, the passive dynamics p_d and state costs q, as well as the desirability function z (and consequently the optimal cost-to-go function v) of the system can be estimated by the same algorithm, in a temporal difference method.

Algorithm 5 Calculating the passive dynamics distributions and state costs during Z learning

When each state x_i is visited:

if $p_d(\cdot|x_i)$ and $q(x_i)$ are not yet known **then**

gather one more equation from the measured immediate cost T_{ni}:

$$T_{\text{ni}} = u_n\left(x_1|x_i\right)\log u_n\left(x_1|x_i\right) - u_n\left(x_1|x_i\right)\log p_d\left(x_1|x_i\right) + \cdots + u_n\left(x_{N\text{adj}}|x_i\right)\log u_n\left(x_{N\text{adj}}|x_i\right) - u_n\left(x_{N\text{adj}}|x_i\right)\log p_d\left(x_{N\text{adj}}|x_i\right) + q(x_i)$$

if $N_{\text{eq}}(x_i) < N_{\text{adj}}(x_i)$ **then**

get one solution for the incomplete system, which respects:

$$\sum p_d(\cdot|x_i) = 1$$

else

get one solution for the complete system, which respects:

$$\sum p_d(\cdot|x_i) = 1$$

consider that $p_d(\cdot|x_i)$ and $q(x_i)$ are known for x_i

end if

end if

5.2 Computational Experiments to Validate the Proposed Method

Experiments were performed to validate the proposed algorithms of Z learning with passive dynamics p_d and state costs q estimation from measured total immediate costs ℓ. The convergence curves of the estimated optimal cost-to-go v were compared to the previously obtained curves, to understand if the convergence speed of Z learning was affected.

These experiments were performed in the same grid world environment described in chapter 3 (figure 3.1), using both passive dynamics distributions models (reflexive environment and absorptive environment).

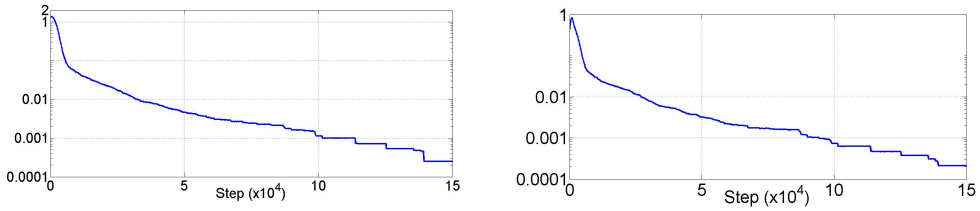
The chosen method for solving the obtained systems of linear equations in these experiments was the variable substitution method (chapter 4 algorithm 3), because it is faster than the gradient descent with probability normalization method (chapter 4 algorithm 2), which is iterative.

In order to calculate the error of the estimated passive dynamics p_d and state costs q, all estimated passive dynamics probabilities were initialized with non-null random values for transitions to adjacent states, which sum up to one. Passive dynamics transition probabilities to non-adjacent states are initialized with probability zero. The estimates of the state costs were initialized with positive random values between zero and one.

The errors of the estimated optimal cost-to-go v are calculated in the same manner as in chapter 3, using equation (3.1). The errors of the estimated passive dynamics distribution are calculated in a similar way, as the sum of the absolute differences between the estimated values and the correct values, divided by the sum of the correct values:

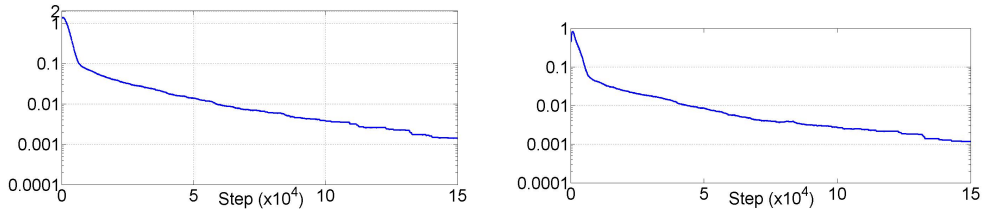
$$\text{Error of } \hat{p_d} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \left| \hat{p_d}\left(x_{\mathrm{j}} | x_{\mathrm{i}}\right) - p_d^*\left(x_{\mathrm{j}} | x_{\mathrm{i}}\right) \right|}{\sum_{i=1}^{N} \sum_{j=1}^{N} p_d^*\left(x_{\mathrm{j}} | x_{\mathrm{i}}\right)} \tag{5.2}$$

where N is the total number of states; $\hat{p_d}\left(x_{\mathrm{j}} | x_{\mathrm{i}}\right)$ is the current estimate of the passive dynamics transition probability from state x_{i} to state x_{j}; and $p_d^*\left(x_{\mathrm{j}} | x_{\mathrm{i}}\right)$ is the correct passive dynamics transition probability from state x_{i} to state x_{j}. Similarly, the esti-



(a) p_d estimation error (b) q estimation error

Figure 5.1: p_d and q approximation errors. The passive dynamics distribution corresponds to a reflexive environment. The vertical axes are in log scale.



(a) p_d estimation error (b) q estimation error

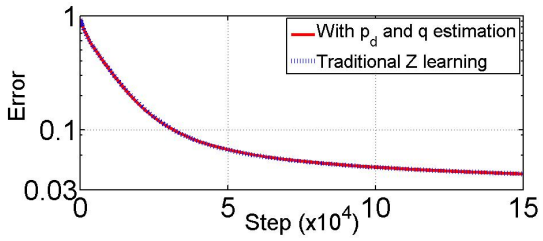
Figure 5.2: p_d and q approximation errors. The passive dynamics distribution corresponds to a reflexive environment. The vertical axes are in log scale.

mated state costs error was calculated as:

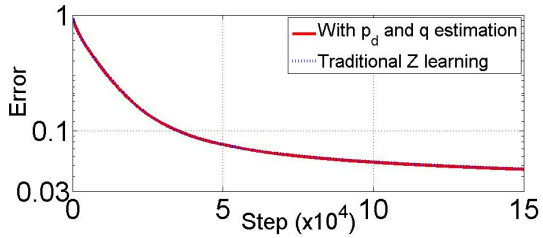
$$\text{Error of } \hat{q} = \frac{\sum_{i=1}^{N} \left|\hat{q}\left(x_{\mathrm{i}}\right) - q^{*}\left(x_{\mathrm{i}}\right)\right|}{\sum_{i=1}^{N} q^{*}\left(x_{\mathrm{i}}\right)} \tag{5.3}$$

where $\hat{q}\left(x_{\mathrm{i}}\right)$ is the current estimate of the state cost of state x_{i}; and $q^{*}\left(x_{\mathrm{i}}\right)$ is the correct state cost of state x_{i}. As can be seen in figures 5.1 and 5.2, as the Z learning with p_d and q estimation algorithm is executed, the errors of the estimated passive dynamics distribution p_d and the state costs q consistently reduce as the number of simulation steps progresses. In the end of the simulation the error was considerably small and only a few states still did not have their passive dynamics distribution and state cost completely calculated. In all experiments, by simulation step $5E-4$ this number was between a minimum of 0.5% and a maximum of 1.2% of all states. By the last simulation step this number was between 0 and 0.7% of all states.

Most importantly, the error in the estimation of those quantities does not seem to have a significant impact in the speed of convergence of the Z learning algorithm, as can be seen in figure 5.3. The convergence of the estimates of the optimal cost-



(a) Error of estimated v: passive dynamics corresponding to a reflexive environment

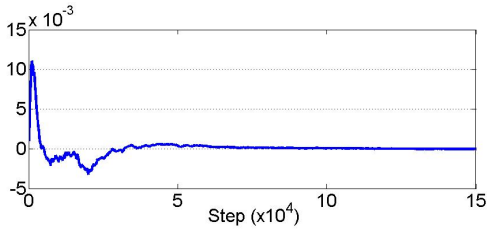


(b) Error of estimated v: passive dynamics corresponding to an absorptive environment

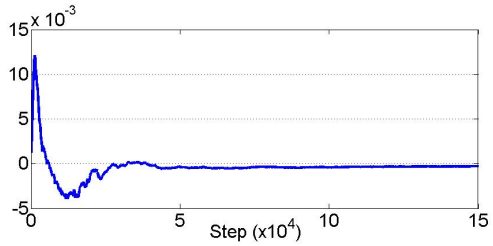
Figure 5.3: Comparison of the optimal cost-to-go v estimation errors during Z learning execution (according to equation (3.1)), for the two algorithms (Z learning with estimation of p_d and q from total costs, and pure Z learning without p_d nor q estimation), for both experimental passive dynamics distributions (reflexive environment and absorptive environment). The vertical axes are in log scale. It can be observed that for both p_d distributions (reflexive and absorptive) the errors of the two algorithms are very similar, which indicates that the proposed method does not significantly affect the convergence speed of the optimal cost-to-go estimates.

to-go function v for the cases with p_d and q estimation was only slightly slower than the convergence of the estimated v in traditional Z learning. This was consistently observed for both simulated passive dynamics distributions (absorptive environment and reflexive environment).

It is difficult to visualize the difference of the two convergence curves (our method and traditional Z learning) even in log scale (figure 5.3). For that reason we have plotted the difference of the two convergence curves (Z learning with estimation - traditional Z learning) for both environments in figure 5.4. It can be observed that our method starts with a bigger error, because the correct values of p_d and q have not yet been estimated. As the time steps evolve and the estimates of p_d and q improve, the error



(a) Reflexive environment



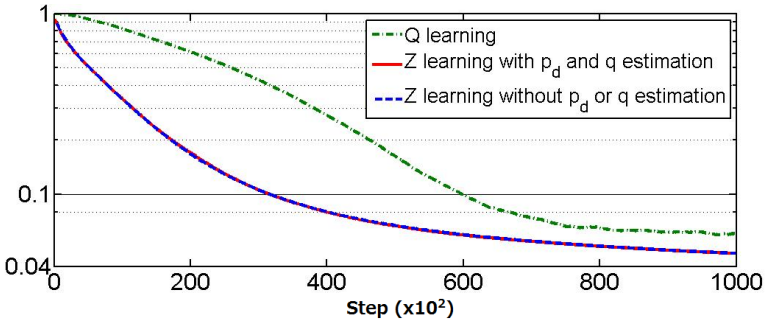
(b) Absorptive environment

Figure 5.4: Differences between the two convergence curves, Z learning with p_d and q estimation, and traditional Z learning. Positive values mean that the error of our method is bigger.

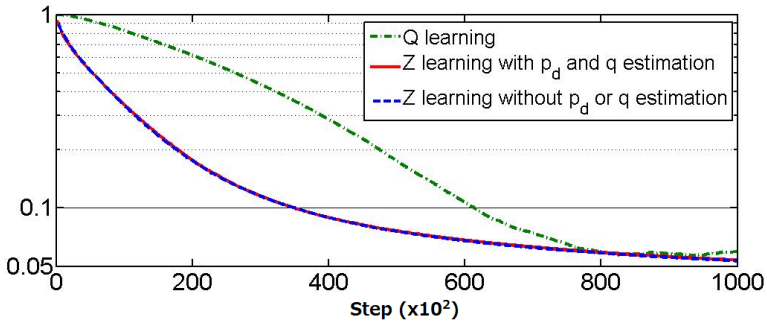
of out method becomes smaller and oscillates, and stabilizes near zero.

5.2.1 Comparison with a Traditional Reinforcement Learning Method

We have also compared the performance of our method and traditional Z learning with Q learning, a traditional reinforcement learning method used in traditional MDPs. For that it was necessary to create a corresponding MDPs with symbolic actions, with the same optimal cost-to-go (or optimal value) functions as our LMDPs. A method to achieve that is explained in (14). For each state x a symbolic action u_1 is defined, and its induced transition distribution corresponds to the optimal transition distribution $u^*(\cdot|x)$. The remaining $N_{\text{adj}} - 1$ symbolic actions $(u_2, \cdots, u_{N_{\text{adj}}})$ induce transition probability distributions obtained from $u^*(\cdot|x)$ by circular shifting. The obtained MDPs are guaranteed to have the same optimal cost-to-go as our LMDPs. As can be seen in figure 5.5, the Q learning method has slower convergence than traditional Z learning, and our method is also faster than Q learning. In our experiments the decaying rate η for all algorithms decayed according to $\eta = c/(c + t)$ where t is the current time step. The constant c was optimised for each algorithm.



(a) Reflexive environment



(b) Absorptive environment

Figure 5.5: Approximation error of the optimal cost-to-go function for Q learning, traditional Z learning and our method. The data was averaged over 10 runs, with different random number sequences. The vertical axes are in log scale.

5.2.2 Difficulties Faced in the Practical Application of the Method

A few difficulties have emerged regarding the application of this method in practice. Such difficulties are now described.

5.2.2.1 Uniqueness of Solutions to the Systems of Linear Equations

The obtained systems of linear equations should have an unique solution when the number of obtained equations equals the number of unknowns (i.e. the number of adjacent states N_{adj}). However, if the followed controlled distributions when visiting a given state (according to the greedy policy) are not sufficiently different from each other, the determinant of the matrix of equations can be too near zero, and actually be considered null within the numerical precision of the simulator software. This causes the software to be unable to find a unique solution to the system, even when N_{adj}

equations are gathered. In the experiments presented in this section a heuristic solution was adopted, by gathering taking the Moore Penrose pseudo-inverse matrix of the coefficients matrix, and also taking two more equations than necessary ($N_{\text{adj}} + 2$) for each state. After this solution was adopted, the number of cases in which the system did not present a unique solution was almost completely avoided.

5.2.2.2 Solution with many zeros for the partially constructed systems of linear equations

this difficulty was found when using the variable substitution method (chapter 4 algorithm 3) to solve incomplete systems, i.e. systems with a number of equations smaller than the number of adjacent states ($N_{\text{eq}} < N_{\text{adj}}$). When using a standard algorithm to solve such systems the obtained solution usually has many null values. For example, when using the standard Matlab routine the solution has at most a number of nonzero values equal to the number of equations N_{eq}. This is a problem when the correct state costs q are positive values (i.e. $q > 0$ which is usually the case), because each element of the unknowns vector $\mathbf{c}$ of the system $A\mathbf{c} = \mathbf{b}$ corresponds to a subtraction of the form $q(x) - log\left(p_d(x'|x)\right)$, where x is the current state and x' is a possible future state. Since the probability $p_d(x'|x)$ is smaller than one, the logarithm is smaller then zero, and therefore if the subtraction is to result in zero the estimated $q(x)$ turns out to be a negative number. This means that the estimated state costs become rewards when those should actually be costs. In order to overcome this problem the incomplete systems were solved using the Moore-Penrose pseudoinverse matrix, which maximizes the Euclidean (L^2) norm and finds a dense solution.

5.3 Final Considerations of This Chapter

In this chapter we have extended the method proposed in the previous chapter to estimate the passive dynamics distribution and state costs of a system during the execution of Z learning. Such method can be applied when following a distribution different than passive dynamics. The main concept behind this method is to follow a policy which changes over time (such as the greedy policy), and measure the incurred immediate costs every time a state is left. Since the policy changes over time, the

measured costs will always be different, therefore a different equation for calculating the passive dynamics and state costs for that state can be gathered.

Even if the necessary number of equations for correctly calculating the passive dynamics and state costs for a given state is not yet obtained, one solution of the incomplete system (with infinite solutions) will be taken as the current estimate.

We have shown with computational experiments that the method works correctly, and that the usage of the estimates of passive dynamics and state costs instead of the correct values does not represent a significant reduction of the convergence speed in terms of simulation steps.

In the next chapter we present concluding remarks and future work possibilities.

6

Conclusion

We have proposed a method for the direct application of Z learning in a true temporal difference approach, without the need for previous knowledge about the passive dynamics of the system nor of state costs. All that is required is the possibility to measure immediate costs (total costs) incurred in state transitions, as well as the knowledge of impossible state transitions and of the controlled state transitions imposed to the system. By following this approach, the system does not have to follow the passive dynamics if it is unknown. The agent is free to follow any policy (which has characteristic of exploration) during the Z learning execution. The characteristic of learning by trial and error of temporal difference methods is preserved, and Z learning can be applied without the previous knowledge of the passive dynamics of the system.

The current work represents an important step in the direction of achieving direct application of the framework of LMDPs to solve realistic problems formulated in discrete time, in a temporal difference approach.

6.1 Issues and Future Work

The proposed method needs to have the equations for calculating the passive dynamics of each state stored in memory. Those equations must be kept until the necessary number of equations is gathered and the correct values of the passive dynamics and state cost are calculated. Once this is done for a given state, then the equations for that state can be discarded. This might have an impact in memory consumption. Future work might address this problem, proposing methods in which the storage of

equations is not necessary. One possibility to solve this problem could take advantage of the geometrical interpretation of the solution of a linear system as a projection of hyperplanes. As soon as a new equation is gathered, the current estimate of pd and q (which corresponds to a point in the hyperspace of solutions) would then be projected into the hyperplane which corresponds to the newly acquired equation. The feasibility of such method still remains to be verified.

After this work, the only remaining limitation for the direct application of Z learning in real problems is the possibility to impose any desired controlled transition distribution. In most realistic problems, symbolic actions might exist and the transition distributions might be dependent of those actions. An extension of the current method to consider symbolic actions (and transition distributions which are caused by such actions) remains to be developed.

Bibliography

[1] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. Mit Press, 1998.

[2] K. Iwata, K. Ikeda, and H. Sakai. The asymptotic equipartition property in reinforcement learning and its relation to return maximization. *Neural Networks 19(1), 62.75*, 2006.

[3] K. Doya. Reinforcement learning: Computational theory and biological mechanisms. *HFSP J 1:30.40*.

[4] T. Kollar and N. Roy. Trajectory optimization using reinforcement learning for map exploration. *The International Journal of Robotics Research, vol.27, no.2, pp. 175-196*, 2008.

[5] M.A.P. Burdelis and K. Ikeda. Temporal difference approach in linearly solvable markov decision problems. In *16th International Symposium on Artificial Life and Robotics (AROB 16th)*, 2011.

[6] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal. Learning variable impedance control. *The International Journal of Robotics Research, vol. 30, no. 7, pp. 820-833*, 2011.

[7] T. Akiyama, H. Hachiya, and M. Sugiyama. Efficient exploration through active learning for value function approximation in reinforcement learning. *Neural Networks, vol.23, no.5, pp.639-648*, 2010.

[8] C.W. Anderson, P.M. Young, M.R. Buehner, J.N. Knight, K.A. Bush, and D.C. Hittle. Robust reinforcement learning control using integral quadratic constraints

for recurrent neural networks. *Neural Networks, IEEE Transactions on, vol.18, no.4, pp.993-1002*, 2007.

[9] J. Nie and S. Haykin. A dynamic channel assignment policy through q-learning. *Neural Networks, IEEE Transactions on, vol.10, no.6, pp.1443-1455*, 1999.

[10] K. Shibata and T. Kawano. Acquisition of flexible image recognition by coupling of reinforcement learning and a neural network. *SICE Journal of Control, Measurement, and System Integration, vol. 2, no. 2, pp. 122-129*, 2009.

[11] A.A.M. Faudzi and K. Shibata. Acquisition of active perception and recognition through actor-q learning using a movable camera. In *SICE Annual Conference 2010, Proceedings of*, pages 1950–1956. IEEE, 2010.

[12] L. Jian. An agent bilateral multi-issue alternate bidding negotiation protocol based on reinforcement learning and its application in e-commerce. In *Electronic Commerce and Security, 2008 International Symposium on, pp.217-220*, 2008.

[13] A. Gaweda, M. Muezzinoglu, G. Aronoff, A. Jacobs, J. Zurada, and M. Brier. Individualization of pharmacological anemia management using reinforcement learning. *Neural Networks, vol.18, no.5-6, pp. 826-834*, 2005.

[14] E. Todorov. Efficient computation of optimal actions. *Proc Natl Acad Sci USA, vol.106, no.28, pp.11478-11483*, 2009.

[15] Kenji Doya. How can we learn efficiently to act optimally and flexibly? *Proceedings of the National Academy of Sciences*, 106(28):11429–11430, 2009.

[16] E. Todorov. Compositionality of optimal control laws. 2009.

[17] M. da Silva, F. Durand, and J. Popović. Linear bellman combination for control of character animation. *ACM Transactions on Graphics, vol. 28, no. 3*, 2009.

[18] E. Todorov. General duality between optimal control and estimation. In *Proc. of the 47th IEEE Conference on Decision and Control, pp 4286 - 4292*, 2008.

[19] E. Todorov and M. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience 5(11): 1226-1235*, 2002.

[20] E. Todorov. Optimality principles in sensorimotor control. *Nature Neuroscience 7(9): 907-915*, 2004.

[21] Krishnamurthy Dvijotham and Emanuel Todorov. Inverse optimal control with linearly-solvable mdps. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 335–342, 2010.

[22] Emanuel Todorov. Finding the most likely trajectories of optimally-controlled stochastic systems. In *World Congress of the International Federation of Automatic Control (IFAC)*, 2011.

[23] E. Todorov. Linearly-solvable markov decision problems. *Adv Neural Informatin Proc Syst vol.19, pp.1369-1376, ed. Scholkopf et al, MIT Press*, 2007.

[24] E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 9999:3137–3181, 2010.

[25] Y. Sun and P. G. Mehta. Fundamental performance limitations with kullback-leibler control cost. In *Proc. of the 49th IEEE Conference on Decision and Control, pp. 7063-7068*, 2010.

[26] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 2007.

[27] K. Dvijotham and E. Todorov. A unifying framework for linearly solvable control. In *Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11). Corvallis, Oregon: AUAI Press, pp. 179-186*, 2011.

[28] M.A.P. Burdelis and K. Ikeda. Estimating passive dynamics distributions and state costs in linearly solvable markov decision processes during z learning execution. *SICE Journal of Control, Measurement, and System Integration (JCMSI)*, in press.

[29] M.A.P. Burdelis and Kazushi Ikeda. Modeling and estimating passive dynamics distributions in linearly solvable markov decision processes. In *IEICE Technical Report, NC2011-20/NC2011-44, 123-128*, 2011.

[30] M.A.P. Burdelis and Kazushi Ikeda. Estimating passive dynamics distributions in linearly solvable markov decision processes from measured immediate costs in reinforcement learning problems. In *the 21st Annual Conference of the Japanese Neural Network Society (JNNS 21st), Okinawa, Japan*, 2011.