A HEURISTIC TEMPORAL DIFFERENCE APPROACH WITH ADAPTIVE GRID DISCRETIZATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

OZAN BORA FIKIR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2016

Approval of the thesis:

A HEURISTIC TEMPORAL DIFFERENCE APPROACH WITH ADAPTIVE GRID DISCRETIZATION

submitted by **OZAN BORA FIKIR** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Prof. Dr. Faruk Polat _____
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Prof. Dr. İ. Hakkı Toroslu
Computer Engineering Department, METU _____

Prof. Dr. Faruk Polat
Computer Engineering Department, METU _____

Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU _____

Prof. Dr. Göktürk Üçoluk
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Tansel Özyer
Computer Engineering Department, TOBB ETU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: OZAN BORA FIKIR

Signature :

ABSTRACT

A HEURISTIC TEMPORAL DIFFERENCE APPROACH WITH ADAPTIVE GRID DISCRETIZATION

Fikir, Ozan Bora

M.S., Department of Computer Engineering

Supervisor : Prof. Dr. Faruk Polat

September 2016, 43 pages

Reinforcement learning (RL), as an area of machine learning, tackle with the problem defined in an environment where an autonomous agent ought to take actions to achieve an ultimate goal. In RL problems, the environment is typically formulated as a Markov decision process. However, in real life problems, the environment is not flawless to be formulated as an MDP, and we need to relax fully observability assumption of MDP. The resulting model is partially observable Markov decision process, which is a more realistic model but forms a difficult problem setting. In this model agent cannot directly access to true state of the environment, but to the observations which provides a partial information about the true state of environment. There are two common ways to solve POMDP problems; first one is to neglect the true state of the environment and directly rely on the observations. The second one is to define a belief state which is probability distribution over the actual states. However, since the belief state definition is based on probability distribution, the agent has to handle with continuous space unlike MDP case, which may become intractable easily in autonomous agent perspective.

In this thesis, we focus on belief space solutions and attempt to reduce the complexity of belief space by partitioning continuous belief space into well-defined and regular regions with two different types of grid discretization as an abstraction over belief space. Then we define an approximate value function which can be used in an online temporal difference learning.

Keywords: Reinforcement Learning, Partially Observable Markov Decision Process, Value Function Approximations, Freudenthal Triangulation

ÖZ

ADAPTİF IZGARA AYRIKLAŞTIRILMASI İLE SEZGİSEL ZAMANSAL FARK YAKLAŞIMI

Fikir, Ozan Bora

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Faruk Polat

Eylül 2016, 43 sayfa

Makine öğreniminin bir alt dalı olan pekiştirmeli öğrenme, otonom bir etmenin herhangi bir çevrede aksiyon alarak nihai bir hedefe ulaşmaya çalıştığı problemlere odaklanmaktadır. Bu problemlerde çevre bir Markov karar süreci olarak modellenmektedir. Ancak, gerçek hayat problemlerinde çevre, bu şekilde modellenebilecek kadar kusursuz değildir, bu durumda Markov karar sürecinin kabul ettiği tam gözlemlenebilirlik varsayımdan vazgeçmemiz gerekmektedir. Ortaya çıkan kısmi gözlemlenebilir Markov karar süreci modeli, daha gerçekçi olup daha zor bir problem alanı tanımlar. Bu problemlerin çözümünde karşımıza çıkan en önemli sorun otonom etmenin gözünde modelin hesaba dayalı denemelerinin sonuçsuz kalabilmesidir. Bu modelde, otonom etmen kanı adı verdiğimiz ve çevrenin gerçek durumları üzerine tanımlanmış bir olasılık dağılımı ile Markov özelliğini sağlar ancak bir olasılık uzayında çalışmak zorundadır.

Bu tezde, kısmi gözlemlenebilir Markov karar süreç problemlerinde karşımıza çıkan ve bir sürekli olasılık olayı olan kanı uzayının iki farklı yöntemle iyi tanımlanmış ve düzenli bölgelere ayrıştırılarak kanı uzayı karmaşıklığının bu soyutlama yöntemi ile azaltılamasına çalışılmıştır. Sonrasında, bu soyutlamayı sezgisel bir kestirme yöntemi içinde kullanılarak iki farklı çevrim içi pekiştirmeli öğrenme yöntemi sunulmuştur.

Anahtar Kelimeler: Pekiştirmeli Öğrenme, Kısmi Gözlemlenebilir Markov Karar Süreçleri, Değer Fonksiyonu Tahminleri, Freudenthal Üçgenleştirme

To my grandfather

Ahmet Bircan Akıl

ACKNOWLEDGMENTS

I would like to thank my supervisor Professor Faruk Polat for his tenacious support, and felicitous guidance. It was a pleasure to work with him for the last four years on this particular subject. There would be no other work that suits my background and profile exactly.

I also would like to thank Dr. Erkin Çilden for his support and guidance on this thesis. All along the way he endured my endless, and confusing repetitive questions, which I now realize that they're almost identical; and he always provoked me to get the result in any case during this work.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

MDP	Markov decision process
POMDP	partially observable Markov decision process
PWLC	piecewise linear and convex
TD	temporal difference
VI	value iteration
PBVI	point-based value iteration

CHAPTER 1

INTRODUCTION

Decision making is the thought process to make logical choices from a set of available options presented by the environment where the decision maker exists in. This thought process can be considered as a problem solving activity terminated by a solution deemed to be acceptable. The problems are generally formalized as a *Markov decision process* (MDP) where environment composed of a set of states, their transitions, and available actions regarded as options induced by environment. The ultimate goal is to take actions in a sequential manner either starting from any non-terminal state to reach some terminal state that returns maximum sum of discounted rewards.

Usually, an autonomous agent is designed to act on behalf of a decision maker and is expected to behave rationally to optimize its long term utility. The solution, which can also be interpreted as the consecutive decisions of actions or the general decision rule executed by the agent, is called a policy. A policy can be constructed before its execution, called an offline policy; or it can be constructed while the agent is exploiting it within the environment, called an online policy. The quality of a policy can be measured in various ways: using its response time in an environment, by number of actions to reach the goal, etc. Constructing the policy offline or online is a parameter that affects its success in terms of some quality metrics. The designer may choose one of the approaches with respect to the constraints introduced by the problem itself. For instance, there might not be enough time to construct an offline policy, or the agent may be memoryless.

In MDP model it is assumed that the state transition information is perfectly available to both the designer and the agent. Besides the design issues about policy construction given above, in many real world problems, unfortunately, the environment information can not be perceived perfectly and the information residing in it is partially hidden from both the designer and the agent. Under these circumstances, the environmental state information is not fully observable but there is a set observations that gives clues about the true state. In most cases these observations are gathered from the agent sensors, where both agent and designer have to rely on and act accordingly. The theory of *partially observable Markov decision process* (POMDP) [1, 22, 15] aims to model this situation and provides fundamentals to compute optimal policies based on observations.

There are solutions directly relying on the observation gathered from agent sensors, where the agent generally is designed to be *reactive* or *memoryless* [12], [14]. However, for many problems, there is no optimal (nor neal-optimal) policy over observations semantic alone. Because the main problem of the observation model is *perceptual aliasing* where two or more different true states of the environment becomes indistinguishable by the agent. An obvious

alternative to solve *perceptual aliasing* problem is to incorporate some form of memory to discriminate perceptually aliased observations[5]. However, within the scope of this thesis we focus on the belief space solutions as defined below.

Only difference of POMDP from MDP is its stochastic foundation, and there is an underlying MDP model the environment has to follow. As an extension, POMDP framework provides all sufficient statistics that satisfies Markov property with addition of an initial probability distribution over the true states of environment when the underlying MDP model is available in advance. With this prior information, the agent becomes able to keep track of its internal belief space, i.e., instead of knowing the exact state, the agent now has its instincts to be in a state with some probability. Since a belief state is defined as a probability distribution over actual states of the environment, the agent has to handle a continuous state space to solve the problem. The advantage of this approach is that any POMDP problem can be transformed to an MDP problem and that allows us to utilize existing MDP solutions. On the other hand, belief state space has dimensionality equal to number of states. While the size of the state space grows exponentially with number of states, any solution to the problem suffers from the *curse of dimensionality*.

A variety of algorithms exists focusing on belief model trying to solve POMDP problems either exactly [22] or approximately [7]. The main challenge in POMDP domain is its stochastic foundation that increases computational complexity. Particularly, finding an optimal policy for a finite horizon case is known to be PSPACE-hard [18] and for the discounted horizon case, it may not be even computable [16]. Not only computation of the exact solution is intractable in POMDP domain [13], but also that of the approximate solutions are intractable as well [15].

The very first attempt to solve POMDP problems for discounted infinite horizon case is Policy Iteration method [9, 24]. The idea is to search policy space for a given POMDP problem instead of iteratively improving value function. The method consists of two steps performed iteratively:

- *policy evaluation*, that computes expected value for the current policy
- *policy improvement*, that improves the current policy

The policy iteration method could not draw significant attention, since it suffers from computational complexity to solve POMDP problems with larger state spaces, specifically more than 10-15 [6].

Approximate POMDP solutions to some ε-precision is another attempt. Lovejoy showed that there are value function approximations defined with dynamic programming operator that becomes the supremum and infimum of optimal value function [15]. Lovejoy partitions the entire belief space into a regular grid structure, whose elements are called simplexes in higher dimensions, with the help of Freudenthal triangulation. And then, it works on the vertexes of simplexes, which can be mapped to certain belief points in belief space, to evaluate value function approximations. For any other belief points falling into the simplex region, he used approximate value functions to decide the value of belief with ε-precision. This thesis is mainly inspired by Lovejoy's studies on approximate solutions, and they are explained extensively in the following chapters.

Another attempt to solve POMDP problems is *forward decision tree* methods. It is the first

attempt to decrease computational complexity by taking initial belief state into consideration. This method is efficient if and only if the problem is defined to have an initial belief. Instead of working the entire belief space, forward decision tree method focuses on the reachable belief space, from given initial belief. It combines dynamic programming techniques and decision trees in such a way that, while a decision tree is constructed to forward lookahead to decide which action is much more beneficial, a dynamic programming update tries to construct a policy [8]. Unfortunately, this method is also relatively inefficient; decision tree method suffers from its exponential growth, and policy construction with dynamic programming suffers from exponential growth in value function complexity. However, there are solutions to bound the growth problem in decision trees by branch and bound pruning [21] and approximation methods can be used for policy construction. Many online solutions to POMDP problems are influenced by this approach [20].

As stated above, the major problem about POMDP framework is its computational complexity, where exact solutions are intractable, approximations with ε-precision is also intractable or not efficient as expected. This is why POMDP research has focused on heuristic methods (or approximations without any error parameters) which can be considered much more efficient. Many heuristic approximations are introduced, and compared in terms of performance in small sized POMDP problems [3].

Most promising and effective solution to POMDP problems is introduced by Pineau et al. called *Point Based Value Iteration* (PBVI) [19], which can also deal with large POMDPs as well. The idea is to sample a set of points from state space and use it as an approximate representation of the state space, instead of representing it exactly. Contradictory to Lovejoy's uniform sampling idea, research area draws attention to point-based algorithms considering to sample reachable belief space that starts from initial belief [25, 11]. Both solutions uses Value Iteration approach over a finite set of sample belief state to produce an approximated value function which requires an offline policy generation, i.e., all sampled belief states are required to be processed to form a new approximated value function at the next step. The overall goal is to have an approximated value function with an ε-precision to induce an approximately optimal policy.

There are also heuristic approaches to solve POMDP problems where the optimal policy is not guaranteed, but empirically shown to be satisfying. While some of these heuristic approximations strictly requires offline policy generation, some of them are defined in reinforcement setting to generate online policies, i.e., an approximated value function is built while the agent is executing in the environment. These heuristic approximations are introduced in the following sections one by one, and our approach is considered in this category as well.

In these thesis, we focused on a belief space POMDP solution which can be defined in a reinforcement learning. The idea is to reduce belief space complexity by partitioning belief space into regular and well-defined regions as an abstraction over entire belief space. We used two different types of grid discretization methods inspired by Lovejoy, as he used to generate approximately optimal offline policies. With the help of this abstraction, we are able to define a value function approximation that can be utilized in a reinforcement setting to generate online policies. Despite the other heuristic online approximations, our approach allows agent to update approximated value function locally, i.e., in a partition of entire belief space, so value function approximation over the rest of belief space is not effected.

The outline of this thesis is as follows. Following this introduction chapter (Chapter 1),

Chapter 2 gives a detailed literature survey on the subject which the thesis work is based on. Chapter 3 presents our heuristic online approximation to solve POMDP problems. In Chapter 4, we compare our method with other heuristic approximation techniques by conducting several experiments on well-known POMDP problems. Finally, we give our concluding remarks and state future research directions in Chapter 5.

CHAPTER 2

BACKGOUND

In this chapter, we summarize the necessary background to formally define the problem that we attack on this thesis by introducing relevant topics and pointing out the related work in the literature. We first introduce the sequential decision making problem, then decision process models are defined formally; namely Markov decision process, and partially observable Markov decision process. Afterwards value function definition is formally given in a dynamic programming approach for both MDP and POMDP models, its properties and algorithms are introduced. Two different approximation of value function in POMDP case is explained in details with formal boundaries and isomorphic property of the belief space is discussed. Utilizing a value function approximation to provide an action selection strategy is explained. Two different grid discretization approach in the literature is given and their use cases are explained in details. Finally, similar heuristic approaches to POMDP problems, which we also use to compare in our experiments, are formally introduced.

2.1 Markov Decision Processes

Decision making is the thought process to make rational choices from a set of available options provided by the environment to optimize some utility metric. Similarly, sequential decision making consists of consecutive decisions made by the decision maker and it is the fundamental task to achieve an overall goal aimed by the decision maker. The essence of sequential decision making is that any decision made at time t has both an immediate and long-term effects, despite the single decision making problem. In sequential decision making problems, the decision maker has to consider a best choice at time t which critically affects his future situations.

Intuitively, it is easier to model a sequential decision making process assuming that any state defined in the model summarizes everything about the current state of the environment, and there is no need to refer any former states. This assumption is called *Markov property* and any process satisfies this property, i.e., processes whose future states only depend on the present state, is called as *Markov process* [10]. A *Markov decision process* (MDP) is used to define a formal framework for the sequential decision making problems under uncertainty having Markov property, and it is formally defined as follows:

Definition 1. *An MDP is a tuple* $\langle S, A, R, T \rangle$*, where*

- S *is a finite set of states,*

- *A is a finite set of actions,*
- *$R : S \times A \to \mathbb{R}$ is a reward function, which indicates the immediate reward that agent receives after taking action a at state s,*
- *$T : S \times A \times S \to [0, 1]$ is a state transition function such that $\forall s \in S$, $\forall a \in A$, $\sum_{s' \in S} T(s, a, s') = 1$, which actually describes the probability of transitioning from state s to s' when action a is taken.*

For an MDP, a *policy* is defined as a function π that maps the set of states to set of actions:

$$\pi : S \to A \tag{2.1}$$

Simply a policy defines which actions to choose at each step of the decision process in any state, and finding a solution to a given MDP problem turns out to be finding the best policy. One can simply search through the policy space to figure out the best policy, and this approach can be realized with a *policy iteration*. The alternative is the *value iteration* approach that will be covered in following section. But both approaches are based on *value function* concepts, which defines a mapping from any state s to the sum of accumulated expected rewards return to evaluate a policy. For any given policy π, value function under π can defined as follows:

$$V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V_\pi(s'), \tag{2.2}$$

and the optimal or best policy becomes,

$$V^*(s) = max_{a \in A} \left[R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s') \right]. \tag{2.3}$$

Solving an MDP problem using policy iteration and value iteration is usually impractical for large state space. On the other hand, reinforcement learning framework which is a synthesis of classical dynamic programming, artificial intelligence (temporal differences), stochastic approximation (simulation), and function approximation (regression, Bellman error, and neural networks) provides efficient solution to the MDP problems.

In a reinforcement learning setting, an MDP is used to model the environment where an autonomous agents ought to take actions. The agent is supposed to interact with the environment by taking actions and observing changes in the environment, without knowing the transition and reward function of the underlying MDP in advance. The interaction between the autonomous agent and the environment in reinforcement learning framework can be depicted as in Figure 2.1. Basically, in this setting, agent has percepts through its sensor denoting the changes in the environment. Particularly in MDP case these percepts are generally same as the true state of the environment since the environment is fully observable and the agent generally keeps track of its environment in an internal model using true state of the environment. Action selector is responsible to choose the best action in each step, based on the policy induced by internal model.

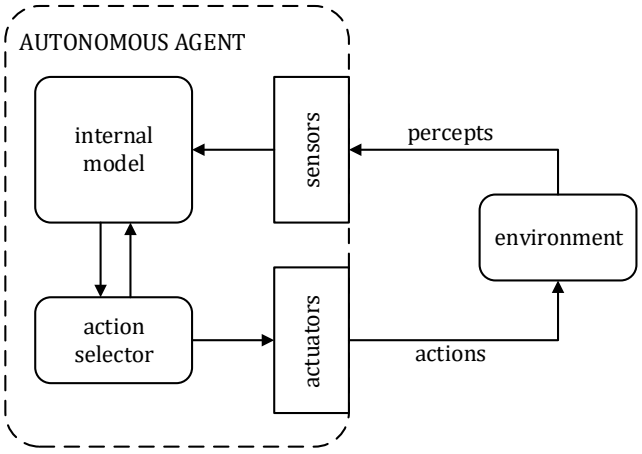


Figure 2.1: Conceptual model of an autonomous agent in reinforcement learning.

2.2 Temporal Difference (TD) Learning

Within the context of this thesis, we focused on the temporal difference learning which is a combination of Monte Carlo and dynamic programming methods. The strong side of TD methods is that it can learn directly from raw experience; sample sequences of states, actions, and rewards from online or simulated interaction with an environment[26]. Unlike Monte Carlo methods, TD methods do not require to complete a task by following a policy π to evaluate it and update value function approximation V of $V\pi$. The simplest TD method known as $TD(0)$ requires only the observed reward at time $t+1$ to update and the estimate of V at $t+1$ to update value function approximation, which can be defined as follows:

$$V(s) \leftarrow V(s) + \alpha \left[r + \gamma V(s') - V(s) \right]. \tag{2.4}$$

Since TD method updates based on an existing estimate (a *bootstrapping* method), i.e., allows an autonomous agent to learn from scratch without any need for external input but only the experience. In a autonomous agent perspective, the following procedure can be followed in an environment modeled as an MDP.

Algorithm 1 Temporal Difference Learning Algorithm

```
procedure TD-0(MDP, α, γ, π)
    initialize V, ∀s ∈ S arbitrarily
    for all episodes do
        repeat(for each step of episode)
            a ← action given by π(s)
            Take action a; observe reward r and next state s'
            V(s) ← V(s) + α [r + γV(s') − V(s)]
            s ← s'
        until s is terminal
    end for
end procedure
```

2.3 Partially Observable Markov Decision Processes

Partially observable Markov decision processes (POMDPs) allows us to model a decision theoretic planning problem under incomplete and imperfect information about the states available to an agent. The agent is expected to make a sequence of decisions to maximize its utility under these circumstances. Formally, a POMDP can be represented with a tuple $\langle S, A, Z, R, T, \Omega \rangle$ where

- S is a finite set of states,
- A is a finite set of actions,
- Z is a finite set of observations,
- $R : S \times A \rightarrow \mathbb{R}$ is a reward function, which indicates the immediate reward that agent receives after taking action a at state s,
- $T : S \times A \times S \rightarrow [0, 1]$ is a state transition function such that $\forall s \in S$, $\forall a \in A$, $\sum_{s' \in S} T(s, a, s') = 1$, which actually describes the probability of transitioning from state s to s' when action a is taken. $T(s, a, s') := Pr(s_{t+1} = s' | a_t = a, s_t = s)$,
- $\Omega : S \times A \times Z \rightarrow [0, 1]$ is an observation function, which is a probability distribution describing the probability of observing z from state s when action a is taken. $\Omega(s, a, z) := Pr(z_{t+1} = z | a_t = a, s_t = s)$.

In a POMDP, the agent is unaware of the actual state of the environment, i.e., the actual state is hidden from the agent, so that it cannot directly use it for decision making. However, POMDP framework actually provides all sufficient statistics that satisfies Markov property with addition of an initial probability distribution over states, which can be denoted by $b_0(s)$. With this prior information agent is able to keep track of its *belief state*, which is a probability distribution over states. Belief state becomes a sufficient statistic for a given history

$$b_t(s) := Pr(s_t = s | b_0, a_o, z_1, ..., z_{t-1}, a_{t-1}, z_t), \tag{2.5}$$

and a successor belief state can be determined if an action a_t is taken by the agent at time t and an observation z_{t+1} is observed at time $t + 1$ such that

$$b_{t+1}(s') = \tau(b_t, z_{t+1}, a_t) = \frac{\sum_{s \in S} Pr(z_{t+1}, s' | s, a_t) b_t(s)}{Pr(z | b_t, a_t)}, \tag{2.6}$$

where $Pr(z|b, a)$ is a normalizing constant defined as

$$\begin{aligned} Pr(z|b, a) &= \sum_{s' \in S} \Omega(s', a, z) \sum_{s \in S} T(s, a, s') b(s) \\ &= \sum_{s' \in S} Pr(z|s', a) \sum_{s \in S} Pr(s'|s, a) b(s), \end{aligned}$$

and

$$\begin{aligned} Pr(z, s'|s, a) &= \Omega(s', a, z) T(s, a, s') \\ &= Pr(z|s', a) Pr(s'|a, s). \end{aligned}$$

In reinforcement setting learning depicted in Figure 2.1, agent now receives observation from the environment as percepts. In this setting inaccurate sensor information, i.e., noise in sensor data can also be introduced. However, in order to derive an internal belief representation agent should know the transition and observation functions of underlying POMDP, or the belief state should be provided from the environment itself.

Remember that for an MDP problem where the state space is discrete, the goal is to maximize expected accumulated reward for each discrete state, which can be defined by a value function. In POMDP case, our aim is the same, but the state space is now continuous, which means that we have to maximize the expected reward for infinitely many belief states. The value function in this case can be formulated as

$$V(b) = max_{a \in A} \left[R(b,a) + \gamma \sum_{b' \in B} T(b,a,b') V(b') \right] \tag{2.7}$$

$$= max_{a \in A} \left[\sum_{s \in S} R(s,a) b(s) + \gamma \sum_{z \in Z^+} \sum_{s \in S} Pr(z|s,a) b(s) V(\tau(b,z,a)) \right], \tag{2.8}$$

where Z^+ is the set of all observation that can be observable when a particular action $a \in A$ is taken.

There are nice properties of the above value function. In order to show these properties we need to rewrite the value function definition 2.7 in value function mapping form where a value function mapping induce an action selection strategy.

2.4 Value Function Mappings

A *policy* is a decision rule, δ, that maps a set of information into actions in A and a *strategy* is a sequence of policies $\{\delta^1, \delta^2, \delta^3, ...\}$ such that δ^t is the policy that determines which action to choose in stage t of the process. If a policy requires only the current information at stage t, that is $\delta : \mathcal{I} \rightarrow A$ then δ is a *Markov policy* and if each policy in a strategy is Markov, then the strategy becomes a *Markov strategy*. Furthermore, if each policy in a strategy is the same, i.e., the decision rule does not change from one stage to another, then we say the strategy is *stationary*.

Let $\mathcal{V}(\mathcal{I})$ be the space of real-valued bounded functions $V : \mathcal{I} \rightarrow \mathbb{R}$ defined on the belief information space $\mathcal{I}$, and the common supremum metric ρ defined by $\rho(V,U) = \sup\{|V(b) - U(b)| : \forall b \in \mathcal{I}\}$. $\mathcal{V}(\mathcal{I})$ is a complete space with its common metric ρ. (Note that the space of piecewise linear convex functions is a subspace of $\mathcal{V}(\mathcal{I})$.)

Now let $h : \mathcal{I} \times A \times \mathcal{V}(\mathcal{I}) \rightarrow \mathbb{R}$ be a real valued function, $\mathcal{D}$ be the set of Markov policies, and define two value function mappings by

$$H_\delta V(b) = h(b, \delta(b), V), b \in \mathcal{I}, \delta \in \mathcal{D}, V \in \mathcal{V}(\mathcal{I}), \tag{2.9}$$

which induces the Markov policy δ, and

$$HV(b) = \sup\{h(b,a,V) : a \in A\}, b \in \mathcal{I}, V \in \mathcal{V}(\mathcal{I}), \tag{2.10}$$

which induces the maximal policy, such that $HV = \sup\{H_\delta V : \forall \delta \in \mathcal{D}\}$. If we define the real valued function h as

$$h(b,a,V) = \sum_{s \in S} R(s,a) b(s) + \gamma \sum_{z \in Z^+} \sum_{s \in S} Pr(z|s,a) b(s) V(\tau(b,z,a)), \tag{2.11}$$

then both H_δ and H value function mappings are *isotone* and *contraction mapping*s.

Definition 2. *A mapping H is isotone, if $\forall V, U \in \mathcal{V}$ such that $V \leq U$ implies $HV \leq HU$.*

Definition 3. *Let ρ be the supremum norm. The mapping H is a contraction under the supremum norm, if $\forall V, U \in \mathcal{V}(\mathcal{I}), \rho(HV - HU) \leq \gamma\rho(V - U)$ holds for some $0 \leq \gamma \leq 1$. [8, 15]*

Lemma 1. *The value function mappings H_δ and H defined in 2.9 and 2.10 respectively are both isotonic and contraction mapping. [8, 15]*

Proof. See Appendix A. □

With the definition of value function mappings we can easily compute the optimal value function or its approximations using *dynamic programming techniques*. The well known approach is *value iteration* [2], defined as a sequence of value-iteration steps $V_i = HV_{i-1}$ where V_i is the i^{th} approximation of the value function. The algorithm for the value iteration procedure given in 2. We can show that the recursive definition of value iteration will converge to a unique fixed point, i.e., an optimal value function V^*, in the limit. This result is a direct consequence of Banach's Fixed Point theorem. [8, 15]

Lemma 2. *The sequence of (V_t) defined by $V_t = HV_{t+1}$ converges to a unique fixed point V^* in the limit such that $V^* = HV^*$.*

Proof. See Appendix A. □

The results above allows us to define ε that is the maximum difference between two consecutive value function in the iteration, so called Bellman error, such that $\varepsilon = \rho(V_{t+1} - V_t)$ [8, 15] and to use in value iteration algorithm given in 2.

Theorem 1. *Let $\varepsilon = \rho(V_{t+1} - V_t)$ be the magnitude of the Bellman error. Then $\rho(V^* - V_t) \leq \frac{\gamma\varepsilon}{1-\gamma}$ and $\rho(V^* - V_{t-1}) \leq \frac{\varepsilon}{1-\gamma}$.*

Proof. See Appendix A. □

The above theorem allows us to determine how precisely the value iteration is approximated, such that to obtain the approximation of V^* with precision δ the Bellman error should fall below $\frac{\delta(1-\gamma)}{\gamma}$.

Algorithm 2 Value Iteration

```
1: procedure VALUE ITERATION(POMDP, ε)
2:     initialize V, ∀b ∈ I
3:     repeat
4:         V' ← V
5:         update V ← HV', ∀b ∈ I
6:     until ρ(V − V') ≤ ε return V
7: end procedure
```

The major problem about the value iteration algorithm given in 2 is that the belief space is infinite and we need to compute V_t for all of it. So, V_t becomes incomputable in a finite number of steps.

2.5 Lower Value Function Approximation

It is shown that the optimal value function V^* is piecewise linear and convex [23, 22], formally it can be defined as follows:

$$V^*(b) = max_{\alpha \in \Gamma^*} \sum_{s \in S} \alpha(s) b(s),$$

where $\Gamma^* = \{\alpha_1, \alpha_2, ..., \alpha_m\}$ is finite set of α-vector each represents an $|S|$ dimensional hyperplane. An optimal value function for 2 state POMDP domain is illustrated in Figure 2.2. Note that for any belief point b, V^* attains its maximum with one of $\alpha' \in \Gamma^*$, i.e., $\alpha' = \max_{\alpha \in \Gamma^*} \left\{ \sum_{s \in S} b(s)\alpha(s) \right\}$, herein this context we call α' is the *gradient* of b. Consid-

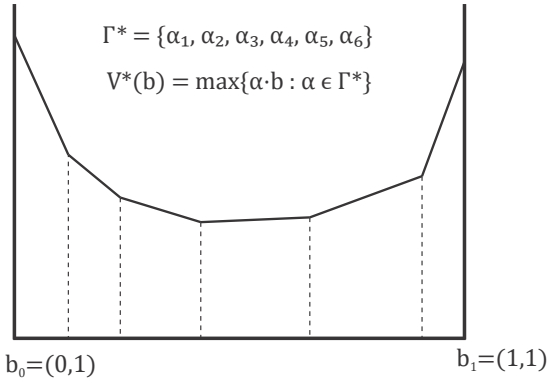


Figure 2.2: An optimal value function for 2 state POMDP domain.

ering this fact, they show that this finite representation of optimal value function can also be applied to any finite horizon problem, such that for any finite t, the optimal value function V_t^* is piecewise linear and convex, i.e., $V_t^*(b) = max_{\alpha \in \Gamma_t} \sum_{s \in S} \alpha(s) b(s)$ for some finite set Γ_t of vectors. Using this representation of V_t^* in the dynamic programming recursion $V_{t+1}^* = HV_t^*$, they show that this mapping can be represented as

$$\begin{aligned} V_{t+1}^*(b) &= HV_t^*(b) \\ &= max_{a \in A} \left\{ \sum_{s \in S}^{b}(s) \left[R(s,a) + \sum_{z \in Z^+} \sum_{s' \in S} Pr(s', z|s, a) \alpha^t_{\iota(b,a,z)}(s') \right] \right\}, \end{aligned} \tag{2.12}$$

where $\iota(b, a, z)$ indexes a linear vector $\alpha^t \in \Gamma_t$ for fixed b, a, z that maximizes:

$$\sum_{s' \in S} \left[\sum_{s \in S} Pr(s', z|s, a) b(s) \right] \alpha^t(s').$$

The linear vector that maximizes V_{t+1}^* at b then is:

$$\alpha_b^{t+1} = R(s,a) + \sum_{z \in Z^+} \sum_{s' \in S} Pr(s', z|s, a) \alpha^t_{\iota(b,a,z)}(s'),$$

and we have $\Gamma_{t+1} = \cup \{\alpha_b^{t+1} : b \in \mathcal{I}\}$ to represent V_{t+1}^*. It is shown that for any t the number of Γ_t is finite and can be computed by a finite algorithm[23, 22]. However, the number of linear vectors computed for each iteration grows exponentially and the problem becomes intractable. In [13], it is proved that all useful linear vectors of V_t^* can be solved efficiently if $RP = NP$.

So far we showed that the optimal value function can be represented by a finite set of vectors Γ^* in a finite horizon, and even if it is intractable to solve we showed that for any t, Γ_t can be computed exactly by a finite algorithm. However, we can define an approximation to the optimal value function for any t with a finite number of points in $b \in \mathcal{I}$. To illustrate this idea, consider the optimal value function example given in Figure 2.2 and further assume that it is optimal for a finite horizon. There are exactly 6 alpha vectors to represent optimal value function at horizon t, and there are exactly 8 extreme belief points. Now assume that, we want to approximate this optimal value function with a finite set of belief of points $b \in \mathcal{I}$, with equi-sized intervals. In this case we will have an approximation of the optimal value function at horizon t which gives us a lower bound. The idea is illustrated in Figure 2.3. As can be seen easily, there are exactly 5 belief points and we have an alpha vector for each belief points. Unfortunately, in this representation we lose one of the alpha vector, α_2, which we have in the optimal case.

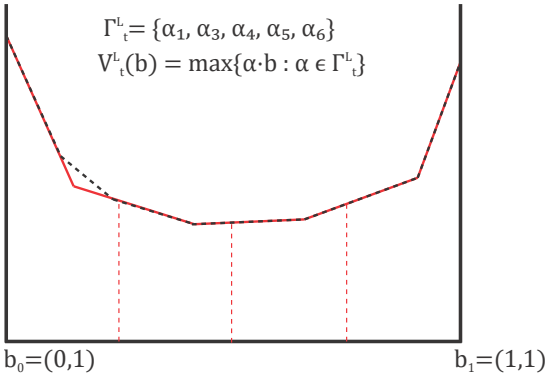


Figure 2.3: A lower bound approximation to optimal value function for 2 state POMDP domain.

Before defining the lower value function approximation formally, let us give Sondik's algorithm to find the next gradient vector in dynamic programming recursion. Let Γ_t be the set of all gradient vectors at horizon t, and we would like to find the gradient vector α_b^{t+1} at b. Note

Algorithm 3 Sondik Algorithm

procedure NEXT GRADIENT($POMDP, \Gamma_t, b \in \mathcal{I}$)
  for all $a \in A$, and $z \in Z^+$ **do**
    $\alpha^t_{i,a,z} \leftarrow \max_{\alpha^t_i \in \Gamma_t} \sum_{s \in S} \tau(b,a,z)(s)\alpha^t_i(s)$
  end for
  for all $a \in A$, and $s \in S$ **do**
    $\alpha^t_{i,a}(s) \leftarrow R(s,a) + \gamma \sum_{z \in Z^+} \sum_{s' \in S} Pr(z,s'|s,a)\alpha^t_{i,a,z}(s')$
  end for
  $\alpha_b^{t+1} \leftarrow \arg\max_{\alpha^t_{i,a}} \sum_{s \in S} b(s)\alpha^t_{i,a}(s)$ **return** α_b^{t+1}
end procedure

that any $V_t \in \mathcal{V}(\mathcal{I})$ that is piecewise linear and convex can be represented by a set of gradient vectors Γ_t, and Algorithm 3 can produce a new piecewise linear and convex function, which is V_{t+1} and represented by Γ_{t+1} whose elements are the gradient vectors obtained from Algorithm 3 with a finite number of belief points $b \in \mathcal{I}$.

Now we can formally define the lower value function approximation and make analysis in

the finite horizon case. For any $V_t \in \mathcal{V}(\mathcal{I})$ that is piecewise linear and convex, i.e., representable by a set of gradient vectors Γ_t, and $b \in \mathcal{I}$, let α_b^{t+1} be the gradient vector at b for V_{t+1} that is $V_{t+1}(b) = HV_t(b) = \sum_{s \in S} \alpha_b^{t+1}(s)b(s)$. Note that $\alpha_b^{t+1} \in \Gamma_{t+1}$. Let assume that we have a finite set belief point G, which will latter be grid points of fixed and adaptive grids,and further assume that we have $\Gamma_{t+1}^L = \cup\{\alpha_b^{t+1} : b \in G\}$ be the set of gradient vectors at next iteration calculated with the belief points in G. If we defined the value function mapping $H^L : \mathcal{V}(\mathcal{I}) \to \mathcal{V}(\mathcal{I})$ by $H^L V_t(b) = \max\left\{\sum_{s \in S} \alpha(s)b(s) : \alpha \in \Gamma_{t+1}^L\right\}$ and since $HV_t(b) = \max\left\{\sum_{s \in S} \alpha(s)b(s) : \alpha \in \Gamma_{t+1}\right\}$ we have the following lemma.

Lemma 3. *For any $V_t \in \mathcal{V}(\mathcal{I})$ that is piecewise linear and convex, $H^L V_t \leq HV_t$ on $\mathcal{I}$. [15]*

Proof. See Appendix A. □

Now recursively define the lower value function approximation as

$$V_{t+1}^L = H^L V_t^L, \tag{2.13}$$

where $V_0^L = 0$ then we have the following lemma in any finite horizon t.

Lemma 4. *For all t, $V_t^L \leq V_t^*$ on $\mathcal{I}$.[15]*

Proof. See Appendix A. □

We now have lower value function approximation, V_t^L, for any finite horizon t that is always less than or equal to optimal value function, V_t^*. More importantly V_t^L is generated with a finite number of belief points $G \subset \mathcal{I}$. Note that the number of gradient vectors in Γ_t^L is not necessarily to be same as the number of points G. Some of the belief points may attain their maximum at the very same gradient, but it cannot exceed $|G|$.

2.6 Upper Value Function Approximation

We already defined a lower value approximation that underestimates the optimal value function. Yet, we can also define an upper value function approximation that overestimates the optimal value function. Again, we consider a finite number of belief points in belief space $\mathcal{I}$ to approximate optimal value function. We use these points to calculate exact value of the optimal value function, but we use linear interpolation to approximate optimal value function any belief point other than these belief points. It is straightforward to interpolate the upper value function approximation as in Figure 2.4; all we need is to find the interval that any belief point b belongs to, then we can use the interval boundaries to approximate the upper value function. However, since state space size increases, we need a proper way of partitioning of belief space $\mathcal{I}$. A solution proposed by Lovejoy is Freudenthal Triangulation of $\mathbb{R}^n$ [15].

The generalization of the notion of a triangle in arbitrary dimensions is called *simplex*, and we will use this term in this context to refer to a particular region in partitioning of $\mathcal{I}$. A simplified triangulation example (for $\mathbb{R}^2$) is depicted in in Figure 2.5. A triangulation of $\mathbb{R}^n$ can be constructed with n-dimensional integer vectors as vertexes. Given any $x \in \mathbb{R}^n$, the vertexes of a particular simplex that contains x can be generated with Algorithm 4.

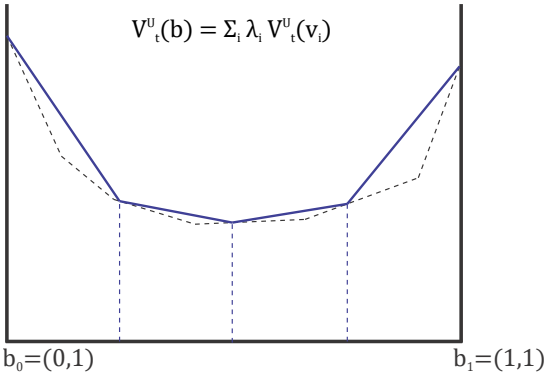


Figure 2.4: An upper bound approximation to optimal value function for 2 state POMDP domain.

Algorithm 4 Find Vertices of Simplex

procedure FINDVERTICESOFSIMPLEX($x \in \mathbb{R}^n$)
  Let $v \in \mathbb{Z}^n$ be the base vector
  for $i = 0, 1, ..., n$ **do**
    $v_i \leftarrow \lfloor x_i \rfloor$
  end for
  $d = x - v$
  Let p be the sort permutation vector of d in descending order, so that $d_{p_0} \geq d_{p_1} \geq d_{p_2} \geq ... \geq d_{p_{n-1}}$
  Let e^j be the j^{th} unit vector
  $v^0 \leftarrow v$
  for $i = 1, ..., n-1$ **do**
    $v^{i+1} \leftarrow v^{i-1} + e^{p_0}$
  end for
  return $\{v^0, v^1, ..., v^n\}$
end procedure

Since any simplex of $\mathbb{R}^n$ of is a convex hull, we can write $x \in \mathbb{R}^n$ as a convex combination of vertexes of the simplex that x resides in, such that $x = \sum_{i=0}^{n} \lambda_i v^i$, where $\forall \lambda_i, 0 \leq \lambda_i$ and $\sum_{i=0}^{n} \lambda_i = 1$. One can find the barycentric coordinates of x with Algorithm 5, where $d = x - v$ and p is the sort permutation of d in descending order.

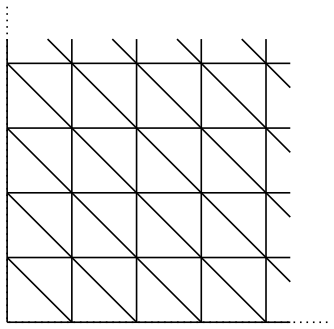


Figure 2.5: Freudenthal Triangulation of $\mathbb{R}^2$

Algorithm 5 Find Barycentric Coordinates

procedure FINDBARYCENTRICCOORDINATES(d, p)
  Let $\lambda \in \mathbb{R}^n$ be barycentric coordinate vector
  for $i = 1, 2, ..., n$ **do**
    $v_i \leftarrow d_{p_{i-1}} - d_i$
  end for
  $\lambda_0 = 1 - \sum_{i=1}^{n} \lambda_i$
  return λ
end procedure

Now let f be a piecewise linear convex function defined over $\mathbb{R}^n$, then we have:

$$f(x) = \sum_{i=0}^{n} \lambda_i f(v^i),$$

We use this interpolation technique to define our upper value function approximation. However, interpolation over $\mathcal{I}$ is not an easy task to do because $\mathcal{I}$ is not well defined for Freudenthal Triangulation. This situation is depicted for $\mathcal{I} \subset \mathbb{R}^3$ in Figure 2.6.

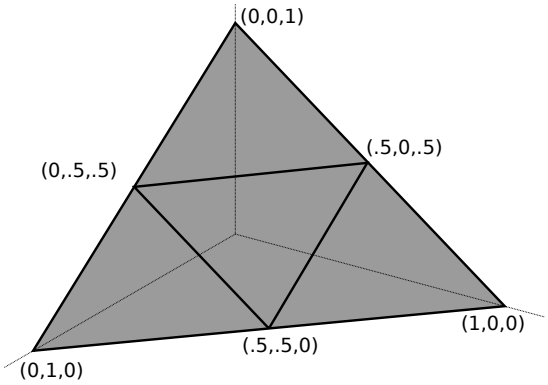


Figure 2.6: Triangulation of $\mathcal{I} \subset \mathbb{R}^3$.

2.7 Isomorphic Grids and Belief Space

Note that, Freudenthal Triangulation is defined over the set integer vectors as vertexes. Lovejoy defines a transformation that maps $\mathcal{I}$ into a subset of Freudenthal Triangulation. Let M be the a positive integer that represent the resolution of Freudenthal Triangulation that we want to achieve. Consider the subset of vertexes in Freudenthal Triangulation of $\mathbb{R}^n$ such that:

$$G' = q \in \mathbb{Z}_+^n | M = q_1 \geq q_2 \geq ... \geq q_n = 0.$$

Define the $n \times n$ nonsingular matrix

$$B = \left(\frac{1}{M}\right) \begin{bmatrix} 1 & -1 & 0 & 0 & ... & 0 \\ 0 & 1 & -1 & 0 & ... & 0 \\ 0 & 0 & 1 & -1 & ... & 0 \\ . & . & . & . & ... & . \\ 0 & 0 & 0 & 0 & ... & 1 \end{bmatrix}.$$

Then for any $q \in G', Bq = (1/M)(M - q_2, q_2 - q_3, ..., q_{n-1} - q_n, dq_n) \in \mathcal{I}$, where the mapping B allows us to induce a triangulation of $\mathcal{I}$ with Freudenthal Triangulation of $\mathbb{R}^n$. The inverse of mapping B is just a scaling transformation that followed by conversion of probability distribution over states to its cumulative distribution. Considering Figure 2.6, the result of the transformation is depicted in Figure 2.7 for $M = 2$. Now, let us assume we have a finite set

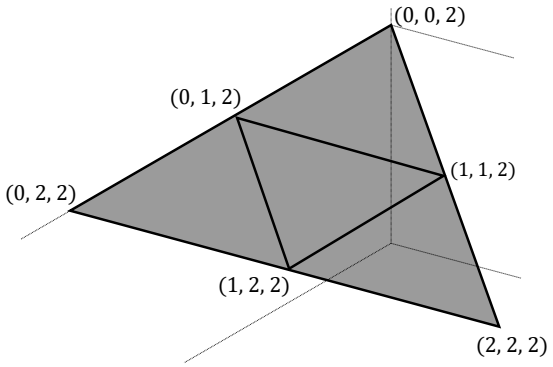


Figure 2.7: Triangulation of $\mathcal{I}$ under transformation B for $M = 2$.

of vertexes of a grid G isomorphic to set of Freduenthal Triangulation vertexes of $G' \subset \mathbb{R}^n$, and define value function mapping $HU : \mathcal{I} \to \mathcal{I}$ as follows:

$$H^U(V)(b) = \sum_{i=0}^{n} \lambda_i HV(Bv^i),$$

where $b = \sum_{i=0}^{n} \lambda_i (Bv^i)$. Note that, we are evaluating HV for belief points in G only, we use piecewise linear approximation for the rest of $\mathcal{I}$. If we define $V_0^U = V_0^* = 0$ and $V_t^U = H^U(V_{t-1}^U)$ then we have an upper value function approximation as the following lemma asserts.

Lemma 5. *For all t,* $V_{t+1}^* \leq min\left(V_{t+1}^U, H(V_t^U)\right)$*.[15]*

Proof. See Appendix A. □

Besides, since $V_t^U \geq V_t^*$, the monotonicity of H implies that $HV_t^U \geq HV_t^* = V_{t+1}^*$. So either, V_{t+1}^U or HV_t^U provides a tighter bound. □

2.8 Action Selection Strategy / Extracting Control Strategy

The ultimate goal of value iteration at the end is to find an optimal strategy for any state of the environment at any time. However, in the previous section we showed that finding an optimal value function even for a finite horizon t is intractable. Instead of optimal value function, we gave a lower value function approximation that is always lower than the optimal value function for a finite set of points. Assume that we computed the value function for a finite horizon t, which action should be the best for any given belief state? Hauskrecht[8] discusses alternative strategies to extract control strategies, one of them is the lookahead design also used by Lovejoy in [15] defined as follows.

For a finite horizon t, define the the Markov policy as

$$\delta_{t+1}(b) = \arg\max_{a \in A} h(b, a, V_t^L) \tag{2.14}$$

for any belief $b \in \mathcal{I}$. With this policy definition one can construct the action selection strategy such that $\Delta_{t+1} = \{\delta_{t+1}, \delta_t, ..., \delta_1\}$ and its value at any horizon $V_{\Delta_t}(b) = h(b, a, V_t^L)$ for any $b \in \mathcal{I}$. Note that the value function mapping that induce this action selection strategy is $V_{\Delta_t} = H_{\delta_t} V_{\Delta_{t-1}} = HV_{t-1}^L$ and it also defines another approximation to optimal value function that is better than the lower value function approximation which is showed in the following lemma.

Lemma 6. *For any t, $V_{t+1}^L \leq V_{\Delta_{t+1}} \leq V_{t+1}^*$ on $\mathcal{I}$.[15]*

Proof. See Appendix A. □

2.9 Fixed Grid Approximations

Lovejoy uses Fixed Grids to partition the belief space $\mathcal{I}$ with Freudenthal Triangulation, to obtain a finite set of vertexes so that he can find both upper and lower function approximations to bound optimal value function. The idea is to narrowing down the region that optimal value function falls into, which is illustrated in Figure 2.8.

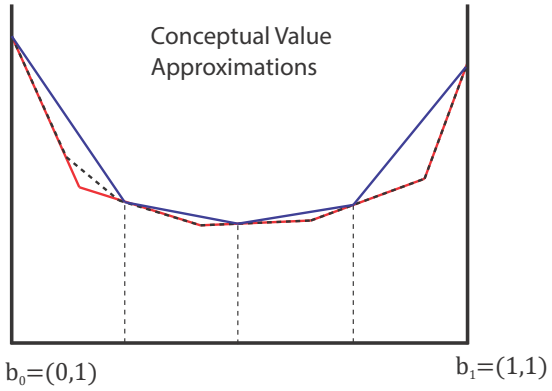


Figure 2.8: Upper and Lower Value Function Approximations against Optimal Value Function.

2.9.1 Finite Horizon

For a particular resolution M, one can enumerate all of the vertexes of grid G which is a subset of Freudenthal triangulation of $\mathbb{R}^n$, and then run both upper and lower value function approximations up to any horizon t. The following proposition is an immediate result of the analysis done so far.

Proposition 1. *For all t,*

$$\begin{aligned} V_t^L = H^L V_{t-1}^L \leq HV_{t-1}^L = H_{\delta_t} V_{t-1}^L \leq H_{\delta_t} V_{\Delta_{t-1}} = V_{\Delta_t} \leq H_{\delta_t} V_{t-1}^* \leq HV_{t-1}^* \\ = V_t^* \leq min\left(V_{t+1}^U, H(V_t^U)\right).[15] \end{aligned}$$

If we use the lookahead design that we proposed at horizon t, then value loss with respect to the optimal value function can be bounded as follows at any belief point $b \in \mathcal{I}$:

$$0 \leq V^*_{t+1}(b) - V_{\Delta_{t+1}}(b) \leq \min\left\{V^U_{t+1}(b), HV^U_t(b)\right\} - HV^L_t(b).[15] \tag{2.15}$$

The bound given in 2.15 depends on the given belief point. We can define an overall error bound for the entire fixed grid approximation, *uniform error bound* Lovejoy called, as follows:

Corollary 1. *For any t,*

$$\rho(V^*_t, V_{\Delta t}) \leq \min\left\{\rho(V^U_t, V^L_t), \gamma\rho(V^U_{t-1}, V^L_{t-1})\right\}.[15]$$

Proof. See Appendix A. □

Corollary 1 allows us to define a supremum over the uncountable belief space $\mathcal{I}$. Fortunately, the piecewise linear structure of V^U and V^L allows us to define a uniform error bound for $\rho(V^U_t, V^L_t)$ just considering grid points in G.

Corollary 2. *Let $\mathcal{F}$ denote the set of sub-simplexes in the triangulation of $\mathcal{I}$, and v denote a vertex of a simplex $\sigma in \mathcal{F}$. For any t,*

$$\rho(V^U_t, V^L_t) \leq \max_{\sigma\in\mathcal{F}} \min_{\alpha\in\Gamma^L_t} \max_{v\in\sigma} \left(V^U_t(v) - \alpha v\right).[15]$$

Proof. See Appendix A. □

Hence for any $\sigma \in \mathcal{F}$

$$\max_{b\in\sigma}\left[V^U_t(b) - V^L_t(b)\right] \leq \min_{\alpha\in\Gamma^L_t} \max_{v\in\sigma} \left(V^U_t(v) - \alpha v\right).[15]$$

The final statement can be obtained by taking maximum over all $\sigma \in \mathcal{F}$. □

Uniform error bounds for over all grid sounds promising, unfortunately it has complexity $O(|S|!|G|^2)$. Note that in order to produce all set of sub-simplexes, $\mathcal{F}$, we need to work through each vertex in G and for each vertex we need to consider all possible permutations of coordinate direction vector. Uniform error bound can be used for the problems where the number of state is less than 10, otherwise it becomes intractable.

2.9.2 Infinite Horizon

The analyses made so far are applicable to finite horizon case of value approximations. However, the ultimate goal is to approximate optimal value function and find an optimal policy accordingly. In infinite horizon case the problem turns out to be finding an optimal value function approximation at horizon T that is sufficiently close to optimal value function in infinite horizon case. We will show that for a large t, we can bound the value loss in infinite horizon case with the finite one. Assume that R_{min} and R_{max} are the minimum and maximum attainable rewards respectively, and let $\gamma \in [0, 1)$ be discount factor.

Lemma 7. *For all t,*

$$\frac{\gamma^t}{1-\gamma}R_{min} \leq V^* - V_t^* \leq \frac{\gamma^t}{1-\gamma}R_{max},$$

where V^* is the optimal value function in infinite horizon, V_t^* is the optimal value function at horizon t. [15]

Proof. See Appendix A. □

We infer that we can use a specific, presumably large, finite horizon t as an approximation to the infinite horizon problem.

2.10 Adaptive Grid Approximations

It is obvious that the quality of fixed grid approximations depends on the resolution of the grid. A finer grid means a better approximation, since we sample belief space with more points. Problem is that the size of grid grows exponentially with any increase in resolution. To solve this problem, Zhou and Hansen proposed a variable grid resolution approach where the resolution of grid varies in different regions of belief space [28] as depicted in Figure 2.9. The expectation is to obtain higher quality approximation with less computational cost.

To allow the resolution of the grid to vary, they adopt the rule that the resolution M is always a positive integer power of 2. With this rule, they ensure that vertexes of high-resolution sub-simplexes are also vertexes of low-resolution sub-simplexes, and still useful when resolution is increased.

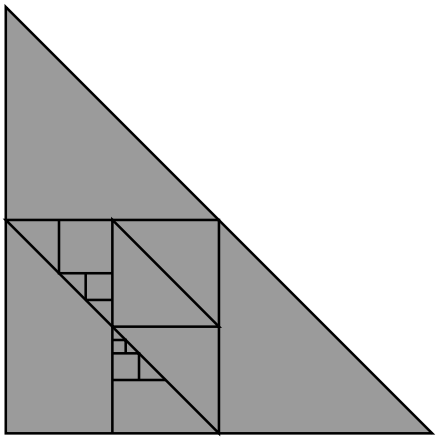


Figure 2.9: Variable Resolution Grid approach.

The algorithm they proposed is slightly different than Fixed Grid approximations, since the resolution of the grid is different in regions of belief space, and requires detailed analysis. We see that in Fixed Grid approximation, for any belief state b we need to find the simplex that b resides in to interpolate upper value function approximation.

In variable resolution grid, on the other hand, they need to search this simplex in different resolutions. Since, we adopted rule that M must be positive power of 2, for any $b \in \mathcal{I}$ we

can find the smallest complete sub-simplex, whose vertexes are in variable grid, with recursive search starting from minimum resolution. Once we identified the smallest complete sub-simplex, we can apply the upper value interpolation as described in Fixed Grid approximations.

Refining the grid, i.e., adjusting the resolution of grid in a part of belief simplex, depends on the error made at an iteration. Remember that we can find an error bound that is made in any t, at any b by Equation 2.15. Once we have identified the vertex/vertexes with the largest error, we can add new vertexes by increasing the resolution to reduce this error. Since the value function approximations at any grid vertex depends on its successors belief states, we can reduce the error at grid vertex by refining the grid at the successor belief state. Assume that for a successor belief b we identified the smallest complete sub-simplex at resolution $M = m$, we can refine value function approximation at this belief state by adding the vertexes of smallest sub-simplex at resolution $2m$ that b resides in.

2.11 Heuristic Approximations

In previous sections, we summarized the optimal or near optimal (with ε-precision) solution approaches. It is obvious that, in general, POMDPs cannot be solved optimally or near optimally in practice [4], furthermore they cannot be approximated efficiently. For these reasons, researchers focused on various heuristic approximation that does not require any error parameter. In this section, heuristic approximations techniques providing efficient POMDP solutions are summarized.

2.11.1 Truncated Exact Value Iteration

Cassandra et al. introduced *Witness Algorithm* [4] to find an exact solution to discounted finite-horizon POMDP problems using value iteration. The idea behind truncated value iteration method is to run Witness Algorithm up to k steps to find an arbitrarily accurate approximation to optimal infinite horizon policy. Because, it is often the case that a near-optimal policy is reached long before the value iteration is converged to optimal point in infinite horizon case.

2.11.2 Q_{MDP} Value Method

If we ignore the observation model in POMDP, we already have an MDP model for which we have dozens of efficient solutions. An ingenuous approach to solve a POMDP problem is to make use of its underlying MDP by ignoring observation model temporarily. Once $Q_{MDP}(s, a)$ table is constructed, one can make use of this information to approximate Q value of an action a, for a given belief space with:

$$Q_a(b) = \sum_{s \in S} Q_{MDP}(s, a),$$

which we can utilize to determine valuable action during agent execution, and we can produce efficient policies [4].

2.11.3 Replicated Q-Learning

Attempts to solve POMDP problems in a reinforcement learning setting and to learn transition and observation probabilities explored in [5] and [17]. *Replicated Q-learning* is an extension of well-know reinforcement learning method, *Q-learning* [27].

The idea is to use a single vector q_a for action a to approximate the Q function for each action for any belief state such that

$$Q_a(b) = \sum_{s \in S} q_a(s) b(s),$$

Whenever an agent makes a transition, q_a estimations updated for every $s \in S$ with

$$q'_a(s) = \alpha b(s)[r + \gamma \max_{a'} Q_{a'}(b') - q_a(s)],$$

where α is learning rate, b a belief state, a action taken, r is the reward gained with this transition, and b' is the resulting belief state. Note that, if $b(s) = 1$ for every $s \in S$, the update rule given below is reduced the original Q-learning update rule. However, this extension is not sufficient in case of high degree of uncertainty, because optimal value function cannot be expressed with q_a vector over the entire belief space in most cases.

2.11.4 Linear Q-Learning

Linear Q-learning is very similar to replicated Q-learning, but in contrast to replicated Q-learning it also considers each q_a vectors that estimates the Q values so the update rule becomes

$$q'_a(s) = \alpha b(s)[r + \gamma \max_{a'} Q_{a'}(b') - \sum_{s \in S} q_a(s) b(s)],$$

Similar to replicated Q-learning, if the belief state is deterministic then update rule is reduced to the original Q-learning update rule [4].

CHAPTER 3

A HEURISTIC TEMPORAL DIFFERENCE APPROACH WITH GRID DISCRETIZATION

There are many reinforcement learning algorithms that can deal with MDP problems [10]. However the reason that we cannot employ these approaches to POMDP case is due to the continuous belief space; we have to deal with probability distribution over states. The major drawback is that belief space is continuous and there are infinitely many believes that need consideration in a reinforcement learning. Discretization of belief space however, may lead us to make use of traditional reinforcement learning approaches. In this chapter, we propose a method to apply classical *temporal difference* learning to POMDP problems with grid discretization.

The key point to solve POMDP problem is to reduce its space complexity. As we have shown so far, there are many attempts to reduce this complexity. Initially, the true state of the environment is ignored and focused directly on the observations. The main problem of such an approach is perceptual aliasing, that the agent becomes unable to distinguish at least two different true states of the environment. With belief space extension, the problem evolves from a discrete state space to a continuous probability space. But it leads us to make use of PWLC property of the optimal value function and allows us to define approximated solutions. Value iteration approaches show that the number of vectors representing the value function, grows exponentially during the iteration and leaves problem intractable. Lovejoy [15] and Pineau [19] attacked to belief space to reduce its complexity, while Lovejoy used a regular sampling over belief with a limited number of vectors to represent value function approximation, Pineau used an irregular sampling over the reachable belief space with an unlimited number of vectors to represent value function approximation that is required to be pruned during iterations. Unfortunately, these approaches are not applicable to be adapted in a reinforcement learning settings. Because both requires that entire sample set belongs to belief space should be processed, updated, and expanded if required iteratively.

On the other hand, there are reinforcement learning approaches such as Linear-Q and Replicated-Q based on the belief extension of POMDPs. They both try to make use of piecewise linear property of optimal value function, with a limited number of linear vectors to represent value function approximation over the entire belief space. The main problem of this approach is that the belief space is completely ignored, such that critical regions of belief space that leads to optimal policy may easily be ignored.

In our novel approach, we try to reduce belief space complexity by partitioning belief space into regular and well-defined regions as an abstraction over the entire belief space. We first

use a fixed grid partitioning method as introduced by Lovejoy[15], and we also use an adaptive partitioning as introduced by Zhou and Hansen [28]. With the help of this partitioning we also define a value function approximation to introduce an abstraction into value function representation. To put all of them into reinforcement learning setting, we adopt temporal difference(TD) methods to estimate value function defined over belief space partitioning. This approach allows us to explore reachable belief space, update value function approximation locally, i.e., in particular region of belief space, and identify critical regions that lead to optimal solution. While the agent updates value function approximation locally, these updates spread out to the neighbor regions due to partitioning of belief space through the edges between regions. We empirically show that our value function approximation convergences and we conclude that our method induces policies no worse than other heuristic approaches.

3.1 A Heuristic TD Approximation with Fixed Grid Discretization

Remember that the classical temporal difference method TD(0) given in Section 2.2 tries to estimate the value function of a given MDP with following online update rule:

$$V(s_t) = V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)],$$

where s_t and s_{t+1} are the states of agent at time t and $t+1$ respectively, and r_{t+1} is the reward gained at time $t+1$.

If we try to employ classical TD(0) method in POMDP case with the following value function definition,

$$V(b_t) = V(b_t) + \alpha[r_{t+1} + \gamma V(b_{t+1}) - V(b_t)],$$

then V becomes a continuous function over belief space and our agent should have an experience table with infinitely many numbers of entries.

In order to solve this problem, we consider to partition entire belief space into regular regions using Freudenthal triangulation. Consider the belief space discretization for 3 state POMDP given in 2.6, where resolution $M = 2$; we have 6 distinct grid vertexes that yield 4 regular regions on belief space. Let's assume that our agent already has value approximations for each grid vertex, then for any given belief b we can find which region b belongs to and we can use a linear interpolation to find the value of being at belief, b.

Formally, let's assume that the entire belief space $\mathcal{I}$ is partitioned into regular simplexes with Freudenthal triangulation, and let $\mathcal{I}^G$ be the set of all belief points that belongs to our grid. For every belief point b that corresponds to a grid vertex, we use the value function approximation stored in our grid, and for the rest we use linear interpolation using barycentric coordinates of the smallest simplex that contains b which can be obtained by Algorithm 5 given in Chapter 2, i.e.,

$$V(b) = \begin{cases} V^G(b), & \text{if } b \in \mathcal{I}^G \\ \sum_{v^i \in S^G} \lambda_i V^G(v^i), & \text{if } b \notin \mathcal{I}^G, \end{cases} \tag{3.1}$$

where $\mathcal{I}^G$ is the set of all grid vertexes, λ_i is the barycentric coordinate that corresponds to vertex v^i of the smallest simplex that contains b.

Furthermore, let b_t is the internal belief of our agent at time t. One can find the smallest simplex that contains b_t, using Algorithm 4 in Chapter 2. In this case we can define the following update rule over smallest simplex that contains b_t

$$V^G(v_t^i) = V^G(v_t^i) + \alpha[r_{t+1} + \gamma V^G(b_{t+1}) - V^G(v_t^i)], \tag{3.2}$$

where v_t^i denotes one of the vertexes of smallest simplex that contains b_t.

To complete this reinforcement setting, we also need to define an action selection strategy as discussed in Section 2.8. For our method, we also use a similar lookahead design as Lovejoy [15] and Hauskrecht [8], which is a greedy one-step lookahead. Assume that the agent belief at time t is b_t, then best action can be selected with the following lookahead design.

$$\delta_{t+1}(b_t) = \arg\max_{a \in A} h(b_t, a, V_t^G), \tag{3.3}$$

where h is defined as in Equation 2.11 in Chapter 2.

In summary, our agent keeps its internal belief and belief space partitioned into regular regions in a particular resolution with a fixed grid discretization. In each vertex of this fixed grid, a value function approximation is defined and stored. Whenever the agent needs to take an action, it first evaluates all possible outcomes with one step lookahead; i.e., for each available actions and observation pairs an expected internal belief b_e in next step is calculated and its value induced by grid discretization, $V^G(b_e)$, is calculated. With $1-\epsilon$ probability, an action with maximum value is selected and executed, or a random action is selected and executed with ϵ probability. Once an observation and a reward is acquired, agent updates its experience induced by grid discretization. The formal steps of the method is given in Algorithm 6.

Algorithm 6 A Heuristic TD Approximation with Fixed Grid Discretization

procedure TD-FixedGrid($POMDP, M, \epsilon$)
  initialize $\mathcal{I}^G$ with M
  initialize $V^G, \forall v \in \mathcal{I}^G$
  initialize b_0
  repeat(for each step of episode)
    Take action $a_t = \arg\max_{a \in A} h(b_t, a, V_t^G)$ with $1-\epsilon$ probability, select a random action otherwise
    Observe b_{t+1}, r_t
    Scale b to b'' in max resolution M
    $S_t^G \leftarrow$ FindVerticesOfSimplex(b'')
    for $\forall v_t^i \in S_t^G$ **do**
      update $V^G(v_t^i) \leftarrow V^G(v_t^i) + \alpha[r_{t+1} + \gamma V^G(b_{t+1}) - V^G(v_t^i)]$
    end for
  until end of episode
end procedure

Note that even if we have value function induced by our grid discretization for the entire belief space, in each step of agent execution only a portion of value function is updated, which is a contradiction to lower and upper bound value approximations defined by Lovejoy. These local updates extinguish the piecewise linear and convex properties of our value function approximation, but allows us to define an *online approximated solution* to POMDP problems.

Any fixed grid constructed with resolution M by Freudenthal triangulation consist of $(M + |S| - 1)/M!(|S| - 1)!$ vertexes at maximum, and they can be stored in hash map with in

lexicographical order. Since, hashing strictly depends on both $|S|$ and M, the size of hash map is not a problem in small POMDP problems. In our test suits, we haven't had any collision in our hash maps, but this issue should be addressed in large POMDP problems.

Another important issue in our algorithm is the action selection by lookahead design. For any given belief b, a successor belief state can be calculated for a particular action and observation pair in $O(|S|^2)$ time. In our lookahead design, we have to evaluate all possible successors for all actions and observation, which means the worst case time complexity becomes $O(|S|^2|A||Z|)$. However in most of POMDP problems, with respect to observation transition model, there are a few observations available for a particular action which allows us to reduce this time complexity. To evaluate any given successor b', we have to calculate its approximate value. In best case scenario, b' may belong to one of the vertexes of fixed grid, and in this case it requires $O(|S|)$ time to find its value in our hash map. Otherwise, we have to search for the smallest sub-simplex that contains b', which requires $O(|S|^2)$ time.

3.2 A Heuristic TD Approximation with Adaptive Grid Discretization

Up to now, we have introduced our TD approximation approach via a fixed resolution grid discretization. But the problem of fixed grid approach is choosing an appropriate resolution for the problem. By appropriate, we mean that the value function approximation induced by fixed grid discretization shall allow agent to explore belief space fairly, but also it shall allow agent to exploit good policies. However, we can see that for very small resolutions, the agent might be stuck in particular regions of belief space, and inevitably the balance between exploration and exploitation is not achieved.

But the resolution of a fixed grid discretization is not the only parameter that affects the balance between exploration and exploitation. The nature of POMDP problems itself makes exploration harder. In most of the POMDP problems we are given an initial belief state, and neither all the actions nor the observation is available for a given belief state. These limitations result in a finite number of branches from a common ancestor belief as depicted in 3.1.

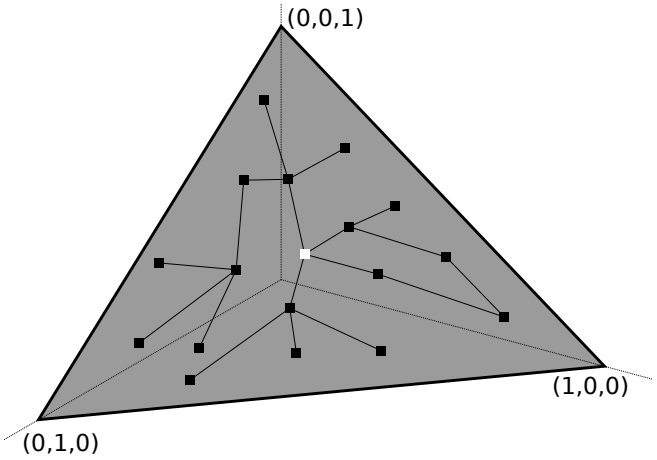


Figure 3.1: Reachable space from an initial belief state.

While the reachability is a major drawback for exploration, it may also help us reduce the problem size. If a region of the belief space is not utilized and not visited by the agent

frequently, can we simply ignore this region to approximate optimal value function? Due to PWLC structure of the optimal value function, we cannot ignore this region. But, we can assume that any error made by an approximate value function is negligible in such a region.

In order to take advantage of reachable belief space, we extend our fixed grid value function approximation with an adaptive grid discretization, where the resolution of grid adjusted during belief transitions of the agent. The agent execution is started with a fixed grid resolution $M = 1$, and a maximum resolution is provided as a power of 2.

For any internal belief of the agent, the smallest simplex is searched and then we increase the resolution on this simplex by adding virtual grid vertexes (if the maximum resolution is not reached yet) to define the set of candidate sub-simplexes. The one that contains b is called the smallest incomplete sub-simplex, and the search algorithm for smallest incomplete sub-simplex is given in Algorithm 7. The idea is to use a more refined grid over reachable belief space to make precise estimations, and use rough estimations over non-reachable belief space regions. Besides, it also helps us to reduce memory requirements to store a grid approximation using more grid vertexes over the reachable belief space and less over the non-reachable belief space.

Algorithm 7 Search For Smallest Incomplete Sub-Simplex

procedure SMALLESTINCOMPLETESUBSIMPLEXSEARCH($b, m, M, \mathcal{I}^G, searchHistory$)
  if $m > M$ **then** ▷ i.e., maximum resolution is reached out
    return
  end if
  Scale b to b' in max resolution, M
  if $b' \in \mathcal{I}^G$ **then**
    Append b' as a degenerate simplex to $searchHistory$
    return
  end if
  Scale b to b'' the given relative resolution m
  $S_m^G \leftarrow$ FINDVERTICESOFSIMPLEX(b'')
  Append $\mathcal{I}^m$ to $searchHistory$
  if $S_m^G \subset \mathcal{I}^G$ **then** ▷ A complete sub-simplex found.
    SMALLESTINCOMPLETESUBSIMPLEXSEARCH($b, 2m, M, \mathcal{I}^G, searchHistory$) ▷ Continue to search for an incomplete sub-simplex in higher resolution.
  end if
  return
end procedure

Note that searching for an incomplete sub-simplex is different from regular simplex search with a factor of $O(logM)$, and at the end of this search we have a precious information of how agent's internal belief follows a path in entire belief space in terms of sub-simplex partitions. With this additional information, we decided to extend the update rule of TD learning to include the values of all the vertexes emerged during this recursive search. Formally, let S_m^G be the set of vertexes belonging to the sub-simplex with resolution $M = 2^m$, and we apply our update rule defined in Equation 3.2 to all the vertexes in $\bigcup_m S_m^G$. This minor change allows the agent to explore and exploit more and more belief points in one execution step compared to the fixed resolution grid approach. It can be considered as a batch update to one branch of reachable belief space. Moreover, any reinforcement obtained at one execution is diffused over an entire reachable belief space region more coherently and fairly. The corresponding algorithm is given in Algorithm 8.

Algorithm 8 A Heuristic TD Approximation with Adaptive Grid Discretization

procedure TD-AdaptiveGrid($POMDP, M, \epsilon$)
  initialize $\mathcal{I}^G$ with $m = 1$
  initialize $V^G, \forall v \in \mathcal{I}^G$
  initialize b_0
  repeat(for each step of episode)
    Take action $a_t = \arg\max_{a \in A} h(b_t, a, V_t^G)$ with $1 - \epsilon$ probability
    Observe b_{t+1}, r_t
    initialize *searchHistroy*
    SmallestIncompleteSubSimplexSearch($b_t, 1, M, \mathcal{I}^G, searchHistory$)
    for $\forall v_t^i \in searchHistroy$ **do**
      if $v_t^i \notin \mathcal{I}^G$ **then** $V^G(v_t^i) \leftarrow V(v_t^i)$
      end if
      update $V^G(v_t^i) \leftarrow V^G(v_t^i) + \alpha[r_{t+1} + \gamma V^G(b_{t+1}) - V^G(v_t^i)]$
    end for
  until end of episode
end procedure

3.3 Successor Cache

As we noted above, the most time consuming step of our approach is the action selection which depends on the successor beliefs for each $a \in A$ and $z \in Z$. In order to reduce its complexity, we decided to use a successor cache using another fixed grid. As a rule of thumb we decided to use a 2 times finer grid, i.e., whose resolution is $2M$. Once a successor request is received on this cache for a belief b, we first transform b to isomorphic equivalent in $2M$ resolution to find the nearest grid vertex v. If such a grid vertex v does not exist in the successor cache, we generate all successors for each action $a \in A$ and $z \in Z$, and store them as $\langle v, a, z, b' \rangle$ tuples in our successor cache. If the grid vertex v is already in our cache, we can easily access any successor b' of b for a given action a and z. Note that generating such a cache for a given b again requires $O(|S|^2|A||Z|)$ time, yet requires $O(|S||A||Z|)$ time in worst case scenario if already exists, and requires constant time in best case scenario. Successor cache concept is depicted in Figure 3.2 for a 3 state POMDP case with 2 actions and 2 observations.

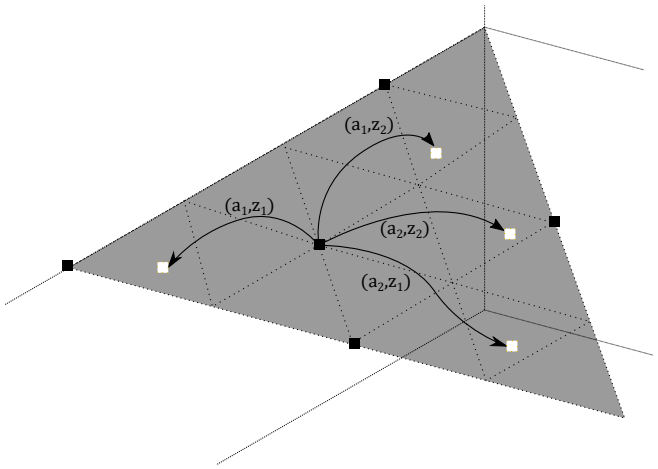


Figure 3.2: Sample successor cache for 3 state POMDP with 2 actions and 2 observations.

CHAPTER 4

EXPERIMENTAL RESULTS

We need to compare our solution with similar heuristic approaches to evaluate its performance. Littman et al. already compared performance of heuristic approximations given in previous sections [13]. To be fair, we use the same small sized problems with the settings they applied. Information about the test suit they applied is given in Table 4.1. Note that, "Noise" column indicates whether there is noise in transitions (T), observations (O), or both (T/O).

By respecting to the test configuration given by Littman et al. we trained our agent for 21 episodes each of which includes $75,000$ steps. At the end of each episode the agent resets its internal belief to the initial belief introduced by the problem itself. The learning rate was decreased according to the following schedule:

- 0.1 for steps 0 to $20,000$,
- 0.01 from $20,000$ to $40,000$,
- 0.001 from $40,000$ to $60,000$,
- 0.0001 thereafter.

Initial values of vertexes in the grid were assigned to 0. The test runs are performed for 101 episodes each of which consists of 101 steps, without any learning updates. The performance is evaluated as a mean per-step reward with 95% confidence interval. Experiments are conducted by TD-approximation using fixed grid resolution (TD_{FG}) and TD-approximation using adaptive grid resolution (TD_{AG}) approaches with different resolutions. Performance comparison with other heuristic approximations are given in Table 4.2.

Table 4.1. Suit for small POMDP problems

Name	$\lvert S\rvert$	$\lvert A\rvert$	$\lvert Z\rvert$	Noise
Shuttle	8	3	5	T/O
Cheese Maze	11	4	7	-
Part Painting	4	4	2	T/O
4x4 Grid	16	4	2	-
Tiger	2	3	2	O
4x3 Grid	11	4	6	T

Table 4.2. Performance Comparison

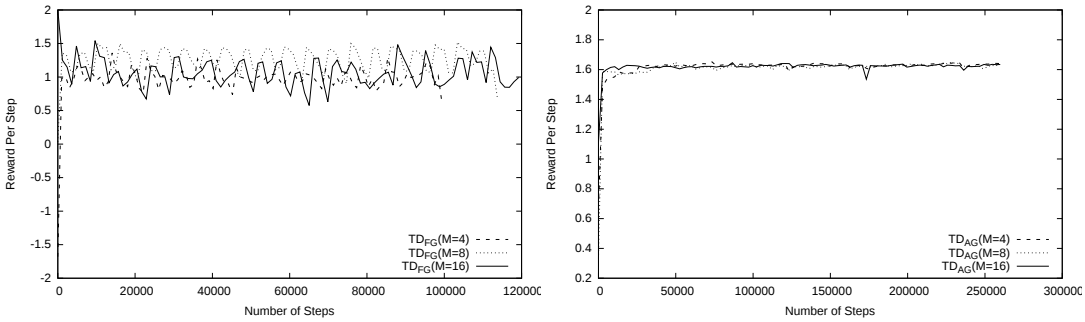
	Shuttle	Cheese-maze	Part painting	4x4-grid	Tiger	4x3-grid
Trunc VI	1.805 ± 0.002	1.188 ± 0.002	0.179 ± 0.012	0.193 ± 0.003	0.930 ± 0.205	0.109 ± 0.005
Q-MDP	1.809 ± 0.012	0.185 ± 0.002	0.112 ± 0.016	0.192 ± 0.003	1.106 ± 0.196	0.112 ± 0.005
Repl-Q	1.355 ± 0.265	0.175 ± 0.017	0.003 ± 0.005	0.179 ± 0.013	1.068 ± 0.047	0.080 ± 0.014
Linear-Q	1.672 ± 0.121	0.186 ± 0.000	0.132 ± 0.030	0.141 ± 0.026	1.074 ± 0.046	0.095 ± 0.007
Optimal	–	0.186 ± 0.002	0.170 ± 0.012	0.192 ± 0.002	1.041 ± 0.180	–
$TD_{FG}(M = 4)$	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.192 ± 0.003	0.929 ± 0.203	-0.036 ± 0.002
$TD_{AG}(M = 4)$	1.799 ± 0.013	0.185 ± 0.002	0.166 ± 0.009	0.193 ± 0.002	1.266 ± 0.180	0.106 ± 0.005
$TD_{FG}(M = 8)$	0.000 ± 0.000	0.000 ± 0.000	0.168 ± 0.009	0.192 ± 0.003	0.938 ± 0.213	-0.034 ± 0.002
$TD_{AG}(M = 8)$	1.795 ± 0.012	0.185 ± 0.002	0.162 ± 0.009	0.190 ± 0.002	1.030 ± 0.205	0.113 ± 0.005
$TD_{FG}(M = 16)$	1.525 ± 0.023	0.184 ± 0.002	0.142 ± 0.009	–	1.174 ± 0.188	-0.033 ± 0.002
$TD_{AG}(M = 16)$	1.794 ± 0.012	0.186 ± 0.002	0.161 ± 0.009	0.191 ± 0.003	0.760 ± 0.244	0.110 ± 0.005

The results show that our adaptive grid approximation is now worse than other heuristic approximations. Even if fixed grid approximation converges to a particular policy in most of the cases, obviously it is not the optimal one. It can be seen that in Shuttle and Cheese-maze problems, fixed grid approximation fails when resolution is set to $M = 4$ and $M = 8$, but successes in $M = 16$. This result shows the importance of higher resolutions, and indicates that any reinforcement obtained by the agent cannot be distributed fairly over the belief space. For the same reason, fixed grid approximation also failed in 4x3 grid problem, which has a pitfall. In this problem, if the agent makes a wrong choice to determine the path to the goal, it is punished by a negative reinforcement. We can say that, fixed grid approximation does not allow the agent to make enough exploration.

To see the convergence speed of our approximation, we also gathered statistics for per step rewards. Note that one can evaluate per step reward by dividing total rewards to number of steps in an episode. However, we calculate per step reward whenever agent reach to a goal state; we evaluate per step reward by dividing total rewards obtained to step size. With this approach, we believe that the policy induced by our method can easily be depicted and illustrated in per step reward graphics. This is the reason why we have oscillation patterns in per step reward graphics. For each problem, both fixed grid and adaptive grid approximations per step reward performances are depicted in the Figure 4.1 - Figure 4.6 for each problems with different resolutions.

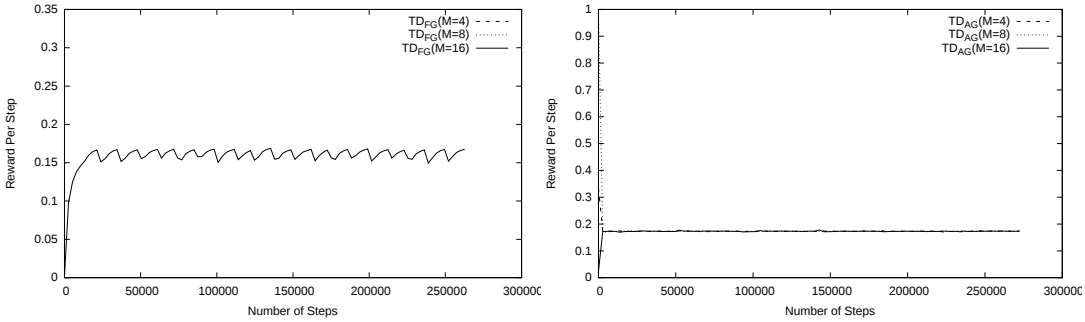
Note that per step rewards performance metrics are gathered during training sessions. Even in this case both approximations converged to a particular policy very fast. But in most of cases of fixed grid approximation, this particular policy is a sub-optimal policy. For adaptive grid approximations, the convergence speed is much faster than fixed grid approximations. Yet, there is another advantage of adaptive grid approximation compared to fixed grid approximations, which is memory consumption. Fixed grid approximation requires all grid vertexes to be generated initially, however in adaptive grid approximation only the boundaries are generated and any required vertex during execution is generated. In Table 4.3, grid sizes in terms of number of vertexes are listed for each problem. When we use the adaptive grid approximation we can observe that the number of vertexes to cover belief space is drastically reduced compared to fixed grid approximation. This observation is the direct consequence of reachable belief space from a particular initial belief.

Figure 4.1: Per Step Reward Performance for Shuttle Problem.



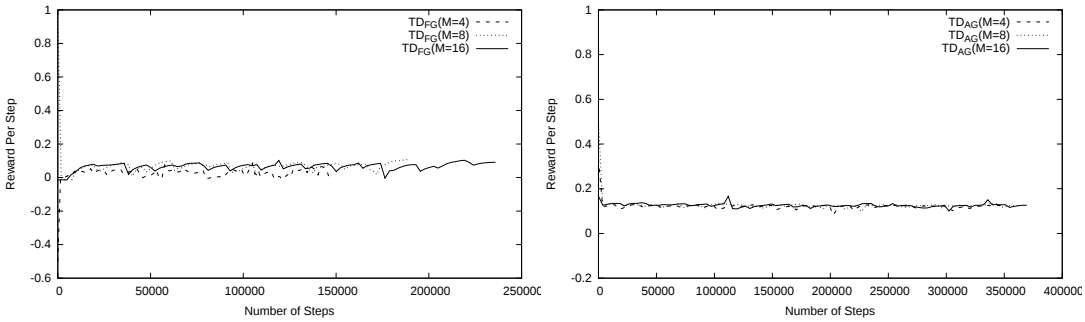
(a) Fixed Grid Convergence (b) Adaptive Grid Convergence

Figure 4.2: Per Step Reward Performance for Cheese-Maze Problem.



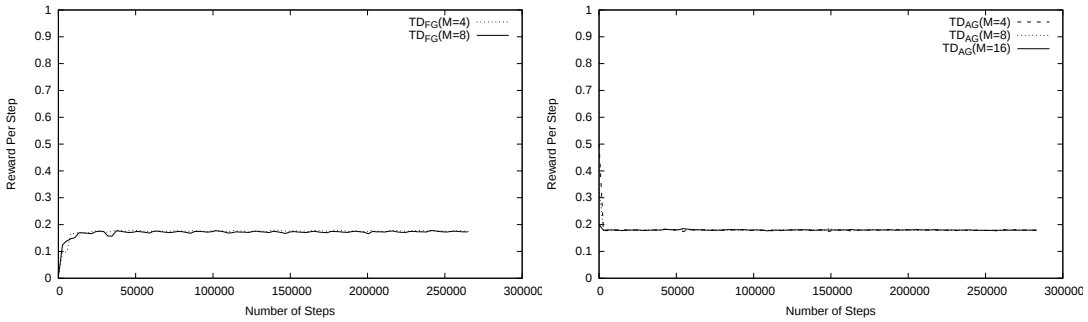
(a) Fixed Grid Convergence (b) Adaptive Grid Convergence

Figure 4.3: Per Step Reward Performance for Part Painting Problem.



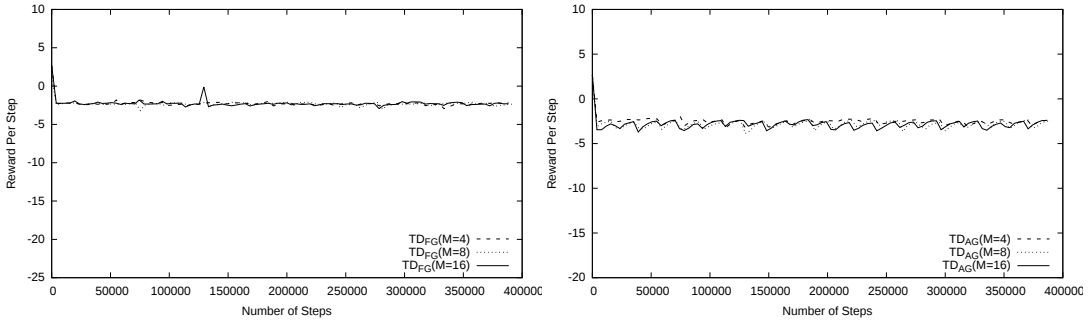
(a) Fixed Grid Convergence (b) Adaptive Grid Convergence

Figure 4.4: Per Step Reward Performance for 4x4 Grid Problem.



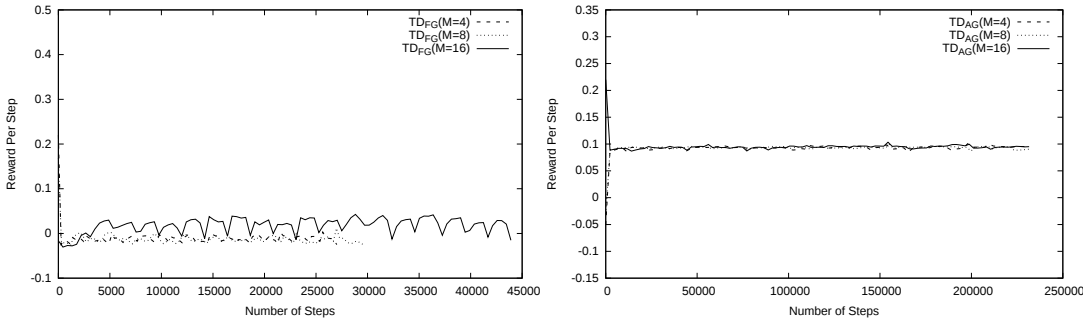
(a) Fixed Grid Convergence (b) Adaptive Grid Convergence

Figure 4.5: Per Step Reward Performance for Tiger Problem.



(a) Fixed Grid Convergence (b) Adaptive Grid Convergence

Figure 4.6: Per Step Reward Performance for 4x3 Grid Problem.



(a) Fixed Grid Convergence (b) Adaptive Grid Convergence

Table 4.3. Grid Sizes

	Shuttle	Cheese-maze	Part-painting	4x4-grid	Tiger	4x3-grid
$TD_{FG}(M = 4)$	358	1,001	51	3,922	9	1,078
$TD_{AG}(M = 4)$	67	78	29	383	5	155
$TD_{FG}(M = 8)$	6,474	43,758	190	490,363	13	43,978
$TD_{AG}(M = 8)$	89	78	44	565	9	317
$TD_{FG}(M = 16)$	245,221	5,311,735	1,017	-	21	5,312,438
$TD_{AG}(M = 16)$	103	78	63	618	9	797

As pointed in the previous section, another refinement that we have made is successor cache. The results for successor cache hits are given in Table 4.4. Each entry indicates the successful successor cache hits over total requests made to this cache. Note that, this results are obtained during training stages. The high percentage rates indicate that the internal belief state of agent follows a very limited path in the entire belief space and it is a direct consequence of the POMDP model. Note that the resolution of successor cache is always 2 times finer than the grid used in TD methods, and the changes of resolution in successor cache does not affect the path the agent follows. Successful successor hit in Tiger problem is 0.00001 due to high stochastic nature of the problem, i.e., a belief state is not likely to be revisited by the agent during its execution. As stated in Section 3.3, generating a successor cache does not increase the time complexity but requires much more memory. The results indicate that if there is no memory restrictions in the agent design, it can be utilized keeping in mind that the performance of successor cache entirely depends on the POMDP model itself.

Table 4.4. Successor Cache Hits

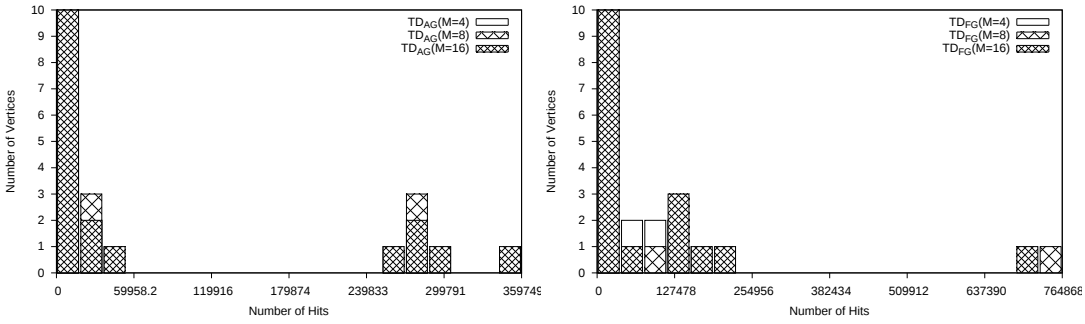
	Shuttle	Cheese-maze	Part-painting	4x4-grid	Tiger	4x3-grid
$TD_{FG}(M = 4)$	69.1%	85.7%	37,5%	33,7%	0.00%	46,8%
$TD_{AG}(M = 4)$	63.6%	72.3%	37,5%	32,1%	0.00%	42,7%
$TD_{FG}(M = 8)$	70.1%	85.7%	37,5%	34,4%	0.00%	47,0%
$TD_{AG}(M = 8)$	63.9%	72.3%	37,5%	32,2%	0.00%	43,5%
$TD_{FG}(M = 16)$	69.3%	72.8%	37,5%	-	0.00%	46,5%
$TD_{AG}(M = 16)$	63.4%	72.4%	37,5%	32,2%	0.00%	43,7%

In the previous section we stated that there are some portions of belief space that can be reachable if the agent start from a particular belief space. To investigate this claim we also gathered hit count metrics for all vertexes of grid that we used. Expectedly, we concluded the same result. Even if we generate a lot of vertexes and partition entire belief space into much more sub-simplexes, the agent's internal belief state is always in the neighborhood of particular vertexes of our grids. In Figures 4.7 - 4.12 the number of vertexes against hit counts depicted to show these results. Note that, there are very few vertexes for which the agent's internal belief is always in the neighborhood.

Considering time and space complexity, experiments shows that adaptive grid approach is always better than fixed grid approaches. Additionally, our adaptive grid approach seems also promising in large POMDP problems. In contradiction to other heuristic approaches, we do not introduce any constraint in the approximation of value function as in Replicated-Q and Linear-Q learning. They both have a limited number of vectors (a vector for each action) to

approximate optimal value function. Besides, we introduce 2 distinct abstractions, one is in belief space and the other in value function approximation. We strongly believe that belief space abstraction can be utilized for sub-goal identification, and it can be extended to identify macro actions which eventually yield better policies. Our method is also convenient for a multi-agent solution by sharing the grid structure among the agents as a common knowledge that can be utilized in large POMDP models.

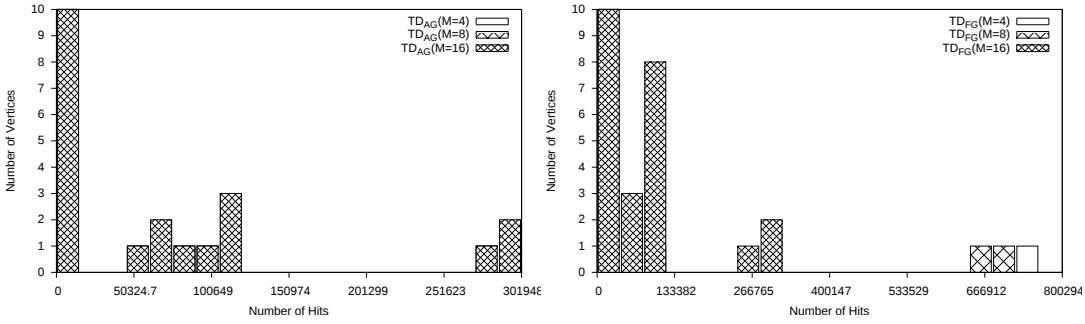
Figure 4.7: Hit Counts for Shuttle Problem.



(a) Fixed Grid Hit Counts

(b) Adaptive Grid Hit Counts

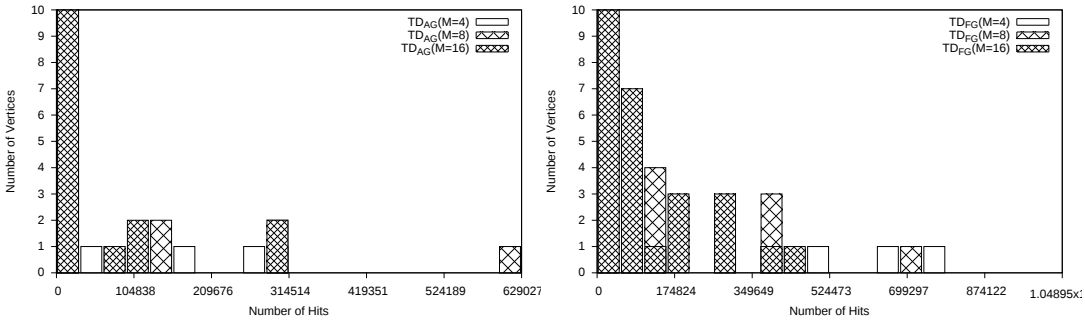
Figure 4.8: Hit Counts for Cheese Maze Problem.



(a) Fixed Grid Hit Counts

(b) Adaptive Grid Hit Counts

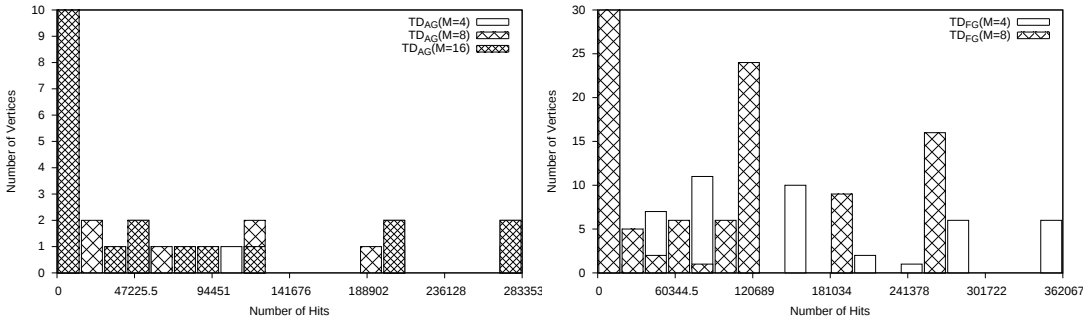
Figure 4.9: Hit Counts for Part Painting Problem.



(a) Fixed Grid Hit Counts

(b) Adaptive Grid Hit Counts

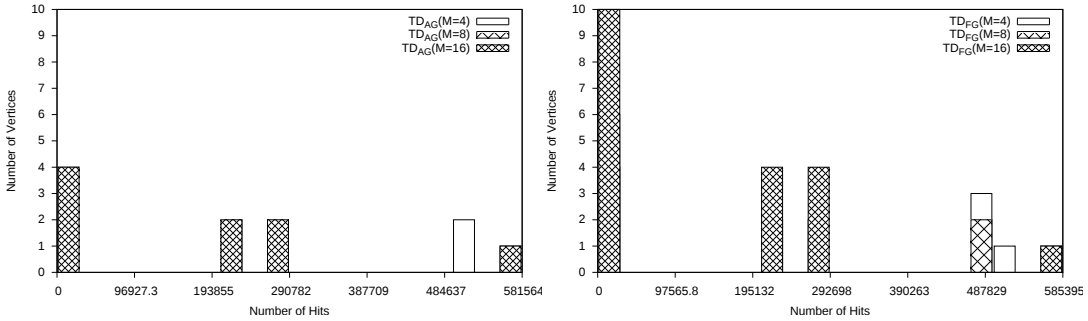
Figure 4.10: Hit Counts for 4x4 Grid Problem.



(a) Fixed Grid Hit Counts

(b) Adaptive Grid Hit Counts

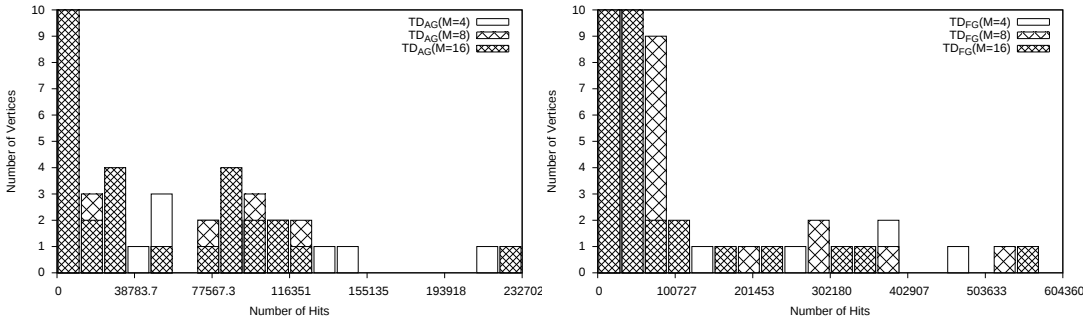
Figure 4.11: Hit Counts for Tiger Problem.



(a) Fixed Grid Hit Counts

(b) Adaptive Grid Hit Counts

Figure 4.12: Hit Counts for 4x3 Grid Problem.



(a) Fixed Grid Hit Counts

(b) Adaptive Grid Hit Counts

CHAPTER 5

CONCLUSION & FUTURE WORK

In this thesis, we present a new heuristic reinforcement learning method to solve POMDP problems with classical temporal difference approach. We also used two distinct abstraction, the first one is Freudenthal triangulation of belief space, and the second one is the value function approximation with TD(0) updates defined over belief space abstraction. Lovejoy [15] was the first to utilize grid solutions for belief spaces years ago, and we showed that his work can still be improved and extended with new approaches to deal with POMDP problems in reinforcement learning perspective. We strongly believe that, grid discretization is just one step to introduce new ways of abstraction in POMDPs which will help us to reduce their complexity drastically.

Existing realizations of fixed grid approach cannot deal with large POMDPs, due to space complexity. But we believe that adaptive grid approach can be easily extended to deal with large POMDP problems, and it becomes much more scalable by introducing new compact data structures into implementations. This issue is noted and planned as a future work. With this new implementation we would like to work on sub-goal identification and macro-action extraction over an adaptive grid. We also would like to extend our adaptive grid approach to multi-agent case.

Note that our methods consists of consecutive *policy execution (action selection)* and *policy construction (backup)* steps as suggested by Ross et al. [20]. Currently, action selection is much more expensive than backup operations, which might become a problem with time critical problems, i.e., if the agent is supposed to act in predefined time interval. We try to solve this problem with a successor cache as discussed in Section 3.3. However, as the results indicate, the performance of successor cache directly depends on the POMDP model. We believe that this problem can be resolved by extending our approximation with Q-values instead of V values as a future work.

Our results indicate that our approximations converge to particular policies for each problem very fast, and we showed that these policies are competitive with the policies generated by the other heuristic methods. While comparing the performance of our methods with other approximate heuristic solutions, we initialized our grid values to 0. Our intention is to make a fair comparison, and respect to test suit introduced by Littman et al [12]. However, a more legitimate way is to initialize these values with respect to the underlying MDP value especially in the adaptive grid approximation since it's started with boundary belief states which can be reduced to original states and can be calculated efficiently. On the other hand, we would like to investigate exploration and exploitation trade-off by different configuration parameters to compare its performance with other heuristic approximations.

REFERENCES

[1] K. J. Åström. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965.

[2] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[3] A. R. Cassandra and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *Machine Learning Proceedings 1995: Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, July 9-12 1995*, page 362. Morgan Kaufmann, 2014.

[4] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 94, pages 1023–1028, Seattle, WA, 1994.

[5] L. Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 183–188, SanJose, CA, 1992. AAAI Press.

[6] E. A. Hansen. Solving pomdps by searching in policy space. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 211–219. Morgan Kaufmann Publishers Inc., 1998.

[7] M. Hauskrecht. Incremental methods for computing bounds in partially observable markov decision processes. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pages 734–739, Providence RI, USA, 1997. AAAI Press.

[8] M. Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, pages 33–94, 2000.

[9] R. A. Howard. *Dynamic Programming and Markov Processes*. Technology Press of the Massachusetts Institute of Technology, 1960.

[10] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, pages 237–285, 1996.

[11] H. Kurniawati, D. Hsu, and W. S. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, volume 2008. Zurich, Switzerland, 2008.

[12] M. L. Littman. Memoryless policies: Theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, volume 3, page 238. MIT Press, 1994.

[13] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Efficient dynamic-programming updates in partially observable markov decision processes. Technical Report CS-95-19, Brown University, Providence RI, USA, 1995.

[14] J. Loch and S. P. Singh. Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 323–331. Madison, 1998.

[15] W. S. Lovejoy. Computationally feasible bounds for partially observed markov decision processes. *Operations research*, 39(1):162–175, 1991.

[16] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *Sixteenth National Conference on Artificial Intelligence*, pages 541–548, Orlando, FL, 1999.

[17] R. A. McCallum. First results with utile distinction memory for reinforcement learning. 1992.

[18] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[19] J. Pineau, G. Gordon, S. Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 3, pages 1025–1032, Acapulco, Mexico, 2003.

[20] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa. Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research*, pages 663–704, 2008.

[21] J. Satia and R. Lave. Markovian decision processes with probabilistic observation of states. *Management Science*, 20(1):1–13, 1973.

[22] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[23] E. J. Sondik. *The optimal control of partially observable Markov processes*. PhD thesis, Stanford University, 1971.

[24] E. J. Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304, 1978.

[25] M. T. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for pomdps. *Journal of Artificial Intelligence Research*, pages 195–220, 2005.

[26] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An introduction*. MIT press Cambridge, 1998.

[27] C. J. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[28] R. Zhou and E. A. Hansen. An improved grid-based approximation algorithm for pomdps. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 707–716, 2001.

APPENDIX A

PROOFS OF STATEMENTS

Proof of Lemma 1. We will show that H_δ is isotone and contraction, so H case is trivial. To show that H_δ is isotone we have to show that $\forall V, U \in \mathcal{V}(\mathcal{I})$ such that $V \leq U$ should imply $H_\delta V \leq H_\delta U$.

$$\begin{aligned}
H_\delta V(b) &= h(b, \delta(b), V) \\
&= \sum_{s\in S} R(s, \delta(b))b(s) + \gamma \sum_{z\in Z^+} \sum_{s\in S} Pr(z|s, \delta(b))b(s)V(b') && (b' = \tau(b, z, a)) \\
&\leq \sum_{s\in S} R(s, \delta(b))b(s) + \gamma \sum_{z\in Z^+} \sum_{s\in S} Pr(z|s, \delta(b))b(s)U(b') && (V(b') \leq U(b') \forall b' \in \mathcal{I} \text{ given}) \\
&\leq h(b, \delta(b), U) \\
&\leq H_\delta U(b) \quad \square
\end{aligned}$$

On the other hand, for any $b \in \mathcal{I}$

$$\begin{aligned}
\rho(H_\delta V - H_\delta U) &= \sup\{\gamma \sum_{z\in Z^+} \sum_{s\in S} Pr(z|s, \delta(b))b(s)[V(b') - U(b')]\} && (b' = \tau(b, z, a)) \\
&\leq \gamma \sup\{|V(b') - U(b')| : \forall b' \in \mathcal{I}\} \sum_{z\in Z^+} \sum_{s\in S} Pr(z|s, \delta(b))b(s) \\
&&& (\textstyle\sum_{z\in Z^+} \sum_{s\in S} Pr(z|s, \delta(b))b(s) \leq 1) \\
&\leq \gamma\rho(V, U)
\end{aligned}$$

$\square$

Proof of Lemma 2. Let us assume that we start the value iteration sequence with a fixed value function V_0. We need to show that (V_t) is a Cauchy sequence. If $t \geq r \geq 1$, then we have

$$\begin{aligned}
\rho(V_t, V_r) &= \rho(H^t V_0, H^r V_0) \\
&\leq \gamma^r \rho(H^{t-r} V_0, V_0) && (H \text{ is contraction mapping.}) \\
&\leq \gamma^r \left[\rho(H^{t-r} V_0, H^{t-r-1} V_0) + \rho(H^{t-r-1} V_0, H^{t-r-2} V_0) + ... + \rho(HV_0, V_0)\right] \\
&&& \text{(Triangle inequality.)} \\
&\leq \gamma^r \left[\sum_{k=0}^{t-r-1} \gamma^k\right] \rho(V_1, V_0) \\
&\leq \gamma^r \left[\sum_{k=0}^{\infty} \gamma^k\right] \rho(V_1, V_0) && \text{(Geometric series.)} \\
&\leq \left(\frac{\gamma^r}{1-\gamma}\right) \rho(V_1, V_0)
\end{aligned}$$

which implies that (V_t) is Cauchy. Since, $\mathcal{V}(\mathcal{I})$ is a complete space with ρ, (V_t) converges to a limit $V^* \in \mathcal{V}(\mathcal{I})$. The fact that V^* is a fixed point follows from the continuity of H:

$$HV^* = H \lim_{t\to\infty} V_t = \lim_{t\to\infty} HV_t = \lim_{t\to\infty} V_{t+1} = V^*$$

Finally we need to show that V^* is unique fixed point solution. If V^* and $V^\#$ are two fixed points, then

$$0 \leq \rho(V^*, V^\#) = \rho(HV^*, HV^\#) \leq \gamma\rho(V^*, V^\#)$$

Since $\gamma \leq 1$, then we have $\rho(HV^*, HV^\#) = 0$, so $V^* = V^\#$ and the fixed point is unique. □

Proof of Theorem 1.

$$\begin{aligned}
\rho(V^* - V_{t+1}) &= \rho(HV^* - HV_t) \\
&\leq \gamma\rho(V^* - V_t) \\
&\leq \gamma\frac{\rho(V_t - V_{t-1})}{1-\gamma} \\
&\leq \frac{\gamma\varepsilon}{1-\gamma}
\end{aligned}$$

□

Proof of Lemma3. From the definitions of Γ_{t+1} and Γ^L_{t+1} it is obvious that $\Gamma^L_{t+1} \subseteq \Gamma_{t+1}$, which indicates that $H^L V_t \leq HV_t$ for all $b \in \mathcal{I}$. □

Proof of Lemma 4.

$$\begin{aligned}
V^L_t &= H^L(V^L_{t-1}) && \text{(Definition of } V^L_t\text{, Equation 2.13)} \\
&\leq H(V^L_{t-1}) && \text{(Lemma 3)} \\
&\leq H(V^*_{t-1}) && (H \text{ is monotonic, and } V^L_{t-1} \leq V^*_{t-1}) \\
&\leq V^*_t && \text{(Definition of } V^*_t\text{, Equation 2.12)}
\end{aligned}$$

□

Proof of Lemma 5. First, we show that for all t, $V^*_{t+1} \leq V^U_{t+1}$. Assume inductively that this inequality holds for t, then for any $b \in \mathcal{I}$ we have

$$\begin{aligned}
V^U_{t+1}(b) &= \sum_{i=0}^{n} \lambda_i V^U_{t+1}(Bv^i) \\
&= \sum_{i=0}^{n} \lambda_i HV^U_t(Bv^i) \\
&\geq \sum_{i=0}^{n} \lambda_i HV^*_t(Bv^i) && (H \text{ is monotonic}) \\
&= \sum_{i=0}^{n} \lambda_i V^*_{t+1}(Bv^i) \\
&\geq V^*_{t+1}\left(\sum_{i=0}^{n} \lambda_i (Bv^i)\right) && \text{(Jensen's inequality)} \\
&= V^*_{t+1}(b)
\end{aligned}$$

□

Proof of Lemma 6. Let $V_0^L = V_{\Delta_0} = V_0^* = 0$ and inductively assume that $V_t^L \leq V_{\Delta_t} \leq V_t^*$

$$\begin{aligned} V_{t+1}^L &= H^L(V_t^L) \\ &\leq H(V_t^L) && \text{(Lemma 3)} \\ &= H(V_t^L) \\ &\leq H_{\delta_t}(V_{\Delta_t}^L) && (H_{\delta_t} \text{ is monotonic.}) \\ &= V_{\Delta_t} \end{aligned}$$

The upper inequality is always true since Δ_t is a feasible strategy and cannot exceed optimal value. □

Proof of Corollary 1. $\forall b \in \mathcal{I}$,

$$\begin{aligned} V_t^*(b) - V_{\Delta t}(b) &\leq V_t^U(b) - V_{\Delta t}(b) \\ &\leq V_t^U(b) - V_t^L(b) \\ &\leq \rho(V_t^U, V_t^L) && \text{(Proposition 1)} \end{aligned}$$

and

$$\begin{aligned} V_t^*(b) - V_{\Delta t}(b) &\leq H(V_{t-1}^U)(b) - V_{\Delta t}(b) \\ &\leq H(V_{t-1}^U)(b) - H(V_{t-1}^L)(b) \\ &\leq \gamma\rho(V_{t-1}^U, V_{t-1}^L) && \text{(Proposition 1)} \end{aligned}$$

hence at any belief point b, $\rho(V_t^*, V_{\Delta t}) \leq \min\left\{\rho(V_t^U, V_t^L), \gamma\rho(V_{t-1}^U, V_{t-1}^L)\right\}$. □

Proof of Corollary 2. Let σ be a specific sub-simplex, on this simplex

$$\begin{aligned} \max_{b\in\sigma}\left[V_t^U(b) - V_t^L(b)\right] &= \max_{b\in\sigma}\left[V_t^U(b) - \max_{\alpha\in\Gamma_t^L}\alpha b\right] \\ &\leq \max_{b\in\sigma}\left[V_t^U(b) - \alpha b\right] \forall\alpha \in \Gamma_t^L \end{aligned}$$

We know that V_t^U is linear on this simplex by construction so the maximum must be attained at a vertex of σ, i.e.

$$\max_{b\in\sigma}\left[V_t^U(b) - V_t^L(b)\right] = \max_{v\in\sigma}\left[V_t^U(v) - \alpha v\right]$$

□

Proof of Lemma 7 . Let Δ_t be the finite horizon action selection strategy that we used in the previous sections, and assume that we decided to use Δ_t for the first t time periods and use any arbitrary policy after time t, Δ_t', for infinite horizon. The following is an immediate result of value functions mappings being a Cauchy series.

$$\frac{\gamma^t}{1-\gamma}R_{min} \leq V_{\Delta_t'} - V_{\Delta_t} \leq \frac{\gamma^t}{1-\gamma}R_{max}$$

which can be easily extended to if we have optimal finite horizon strategy

$$\frac{\gamma^t}{1-\gamma}R_{min} \leq V^* - V_t^* \leq \frac{\gamma^t}{1-\gamma}R_{max}$$

□