INVESTIGATION ON PATH PLANNING FOR AUTONOMOUS MOBILE ROBOTS IN PARTIALLY OBSERVABLE ENVIRONMENT

LAW CHENG QUAN

A report submitted in partial fulfilment of the requirements for the degree of Bachelor of Mechatronics Engineering

UTeM
اونيۏرسيتي تيکنيکل مليسيا ملاک
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Faculty of Electrical Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2017

“I hereby declare that I have read through this report entitle “Investigation on Path Planning for Autonomous Mobile Robots in Partially Observable Environment” and found that it has comply the partial fulfilment for awarding the degree of Bachelor of Mechatronics Engineering.

Signature : .....

Supervisor’s Name : NUR ILYANA BT ANWAR APANDI

Date : .....

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

I declare that this report entitled “Investigation on Path Planning for Autonomous Mobile Robots in Partially Observable Environment” is the result of my own research except as cited in the references. The report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature : .....

Name : LAW CHENG QUAN

Date : .....

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

To my beloved father and mother

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ACKNOWLEDGEMENT

First and foremost, I would like to express my immeasurable appreciation and deepest gratitude to University of Technical Malacca (UTeM) for providing an opportunity for me to undertake my Final Year Project in partial fulfilment for Bachelor of Mechatronics Engineering.

I am deeply indebted towards my project supervisor, Miss Nur Ilyana bt Anwar Apandi for her patience guidance and keen interest on me at every stage of my project progression. Her prompt encouragements, timely assistance, erudite advice, and warm kindness have motivated me to perform better and widen my research boundaries in the completion of my Final Year Project.

Thanks are also extended to my panels, Ms. Nurul Fatiha bt Johan and Ms. Nurdiana bt Nordin who have assessed my presentation and gave valuable comments for my project. Also, I would take this opportunity to express my gratitude to my parents for their continuous shower of love, unceasing encouragement and support throughout all these years.

Last but not least, I place on record, my sense of gratitude to one and all who, directly or indirectly, have offered their helping hand during the entire period of Final Year Project.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ABSTRACT

The autonomous mobile robot uses partially observable Markov decision processes (POMDP) model for the shortest and the best path planning to reach a destination in a partially structured environment. POMDP model is applied to improve computational efficiency of path planning problem. Sensing and information processing is important in autonomous mobile robots. Path planning in the real world is difficult because of partial observability and dynamic changes in the environment. Computational complexity increases when more variables are involved. The Perseus algorithm is investigated and the outcomes such as value function, reward and number of vectors are evaluated on different POMDP problems. Perseus algorithm improves belief point collection and selection to compute for better value functions. The algorithm randomly explores the belief space of an environment and collect a set of reachable belief points which will be fixed throughout the algorithm. Then, new value functions are computed to update the belief points. The algorithm repeats until a convergence criterion is met. Varying number of states and actions have significant effects on value function and number of vectors. While reward depends on the value of reward state and cost state.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ABSTRAK

Robot mudah alih autonomi menggunakan pemerhatian sebahagian proses keputusan Markov (POMDP) dalam perancangan jalan yang singkat dan terbaik untuk sampai ke destinasi yang dalam persekitaran berstruktur sebahagian. Pemprosesan sensing dan maklumat adalah penting dalam robot mudah alih autonomi. Perancangan laluan dalam dunia sebenar adalah sukar kerana keteramatan separa dan perubahan dinamik dalam persekitaran. Kerumitan pengiraan meningkat apabila lebih pembolehubah yang terlibat. Algoritma Perseus disiasat dan hasil seperti fungsi nilai, ganjaran dan bilangan vektor dinilai pada masalah POMDP berbeza. Algoritma Perseus meningkatkan koleksi titik kepercayaan dan pilihan untuk membuat pengiraan untuk fungsi nilai yang lebih baik. Algoritma secara rawak meneroka ruang kepercayaan alam sekitar dan mengumpul satu set mata kepercayaan dicapai yang akan tetap sepanjang algoritma. Kemudian, fungsi nilai baru dikira untuk mengemaskini mata kepercayaan. Algoritma mengulangi sehingga kriteria penumpuan dipenuhi. nombor yang berbeza-beza negara dan tindakan mempunyai kesan yang besar ke atas fungsi nilai dan bilangan vektor. Walaupun ganjaran bergantung kepada nilai ganjaran negeri dan negeri kos.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

TABLE OF CONTENTS

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

LIST OF TABLES

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

LIST OF FIGURES

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHAPTER 1

INTRODUCTION

1.1 Introduction

Uncertain environment in robotics research is a challenge to the operation of autonomous robotic systems [1]. Autonomous navigation in partially observable domains is an extensive research area in mobile robotics. For this reason, researchers have developed methods for mobile robots to overcome dynamic changes in the environment since the last few decades.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Partially observable Markov decision processes (POMDP) provide a powerful framework for mobile robot planning under uncertain environment. POMDPs generalize Markov decision process (MDP) model and offer a natural and principled framework for sequential programming to allow more variables to be incorporated in the process [7]. The POMDP model contains several qualities such as abstraction, adaptability and robustness [7]. The POMDP model is used for robot navigation [2], exploration tasks [8], machine learning [9] and other purposes. Consider a mobile robot is moving in a real world, which the robot perceives it as a grid world in discrete time, each grid represents a state in which the robot acts. A transition probability function tells the robot what to do by observing the environment through its sensors. The robot receives a reward or cost after performing the action. Thus, the POMDP component will have system state, action, transition probability function, reward function, observations, observation function, belief state and discount factor if it is a finite criterion. The robot has to generate a belief-state space over the

underlying state space by using an algorithm to compute the robot's current location. The robot must be equipped with sensors to detect obstacles such as walls and landmarks in order to update the belief state.

In the principle of mathematics, the complexity of algorithms increases when the number of variables increases. The number of variables especially state space and observation space grow exponentially over time, making computation for exact solution impossible. Over the years, many researches have done to increase the scalability of the algorithms that is able to solve larger problems such as decentralized-POMDP [5], hierarchical-POMDP [3], and dynamic Bayesian networks [10]. Although there are many advance navigation algorithms are introduced, but they are still not ready for dynamic changes of the real world.

1.2 Motivation

The primary challenge in implementing POMDPs for navigation is that the robot has to first model the physical environment to state spaces. Due to partial observability of the state, the robot does not know the exact location it is in, so it cannot execute the action recommended for that state [28]. This increases the computational costs of any associated algorithm resulting in high computational complexity [2]. In reality, the state is not always giving the exact information, and the sensor of the robot is not always giving the accurate value.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

A significant of research have done on the application of POMDP in mobile robots over the past few decades. The algorithm's scalability can be improved by decreasing the order of observation functions from exponential to polynomial [5]. Hierarchical decomposition of POMDP enables mobile robot to breakdown large and complex maps to formulate a sequence of sensing and processing suitable for its main objective [3]. The robot needs to plan sequentially one after another to find coordinated trajectories with an adapted version of classical prioritized planning [6]. However, the algorithm becomes impossible to compute when the size of the observation set increases. The motivation of this research is to solve the problem by using an extension to point-based value iteration

algorithm, known as Perseus algorithm. The algorithm samples a finite set of reachable belief points and then update the belief points.

1.3 Problem Statement

Development of autonomous mobile robot is heavily based on sensing and information processing to a specific task. Path planning in the real-world domain is particularly difficult because partial observability and dynamic changes occur continuously. The existing mobile robot system is built for static environment only, which is prone to sensing error during navigation when it is equipped with sensors [3]. Every action executed by the mobile robot may affect the total reward it will receive. However, the mobile robot may take a considerable amount of time to evaluate the long-term reward from its action. Moreover, human proficiency and time to provide detail and accurate feedback is crucial in designing a mobile robot to navigate in complex domains [3].

It is necessary for a mobile robot to respond quickly to dynamic changes on the environment so that it would not need human intervention during operation. The sensor of a robot is important to provide information on changes of the environment. However, sensor is not reliable because it does not provide accurate information of the real world consistently. The sensor may not work properly due to its physical constraints or when it breaks down. Application of the project is an exploration task specifically for logistics. A mobile robot is deployed with a predefined of an environment. The task of the mobile robot is to transfer an object from a source to a destination. Essentially, it has to plan the best route to transfer the object. The main objective of the mobile robot is to maximize the reward when undergoing each path planning algorithm [4].

1.4 Objectives

The objectives of the project are:

1. To investigate path planning for mobile robots by using Perseus algorithm.
2. To evaluate the performance of the algorithm in terms of value function, reward and number of vectors in partially observable environments.

1.5 Scopes

The scopes of the project are:

1. Partially observable Markov decision processes is applied to model the path planning problem in a partially observable environment.
2. A predefined map of an enclosed area in an indoor environment is stored in a mobile robot for path planning task.
3. The map is a 2D bounded environment.
4. A single robot will be navigating in the enclosed area.
5. Evaluated factor is the average reward collected by the robot when undergoing each path planning algorithm.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

1.6 Thesis Organization

The thesis is organized as follows. The next chapter presents the literature review on POMDPs and other well-known methods for solving POMDP problems. Chapter 3 describes the methodology and application of the algorithm in a mobile robot travelling in a partially observable area. Chapter 4 compares and discusses the result using Perseus algorithm between several well-known POMDP problems. Chapter 5 concludes the overall work and proposes recommendation for future research based on the outcomes of the research.

CHAPTER 2

LITERATURE REVIEW

2.1 Path planning

Path planning of autonomous mobile robot is a challenge in the real world. In this project, the robot already has a grid-based world of the predefined map of an enclosed area. The grid world defines each state of the world that allows the robot to plan a path over it. Every action on a state is offered with a specific amount of cost or reward.

When there are changes in the domain map due to changes in object arrangements, the robot automatically updates the map and recalculates the path to a destination [12]. The path planning algorithm such as modified pulse-coupled neural network (MPCNN) [13] uses a simple neural network by first collecting the robot's location, destination and obstacles that plan the shortest collision-free path so that the robot moves to the grid cell containing the highest reward [12]. Energy consumed by the robot can be reduced when planning an optimal path [13]. However, MPCNN do not include a learning function that lets the robot to learn.

In recent years, new path planning methods have been introduced such as improved Q-learning (IQL) and heuristic searching techniques for mobile robots [14]. The methods limit the belief space and variation range of the mobile robot. There are two path planning methods that are used in a static environment. Global path planning is finding a path before execution in a static environment. This planning method is computationally intractable in more complex environment. Local path planning is used in partially observable

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

environment. The IQL algorithm is combined with several exploration strategies to reduce computational time [14]. However, IQL method is still having difficulty in path planning in a dynamic environment with a large number of state spaces.

2.2 Partially observable Markov decision processes (POMDPs)

When a mobile robot is moving in a partially observable environment, the mobile robot is often difficult to make the best decision to achieve task objective. Hence, this kind of problem has to be modeled as POMDPs. This method is gaining popularity in the modern autonomous mobile robot application though it is computationally complex. POMDPs planning is capable of predicting the future by studying the history for a finite time bound. However, applying the general POMDP model to perform simple tasks such as navigation only is PSPACE-hard [6], which means it is very complex and not efficient to compute [28]. Hence, many research focus on improving the navigation algorithm [1], [2] or reducing the number of variables for computation [19, 20].

An exploring autonomous mobile robot has to update state of the environment every time the state changes. Dynamic Bayesian Network (DBN) is used to monitor the environment changes and update the belief state at each node in the network [23]. However, new nodes are generated when updating the DBN and may reach high computational complexity after a short time for large problems.

Task planning and motion planning level are a sequence of operations that take the robot to reach the task objectives. Motion planning level are executed only after the task planning level has computed, and this may lead to undesired motion planning solutions when the proposed task plans are too difficult to be computed by the motion planning level. This problem is modeled as the simultaneous task and motion planning (STAMP) problem. The STAMP problem is integrated with task motion multigraph (TMM) algorithm to increase the efficiency in solving STAMP problem. TMM-based algorithm is able to solve problems that demonstrates Markov Decision Processes (MDPs) [24]. However, due to the nature of MDP problems, the integration with TMM often unable to accurately solve the problem and it takes longer time to compute, though this paper focuses on solving POMDP problems, which is an extension of MDPs.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2.3 Point-based value iteration (PBVI) algorithm

The problem is that the complexity of the process becomes more difficult because the number of state space increases exponentially as time goes by. The introduction of the PBVI algorithm has been able to approximately solve large POMDPs rapidly. This section discusses about the PBVI algorithm.

The PBVI algorithm samples a representative set of points from the belief space and use it to represent the space approximately. Recent algorithms are more efficient by sampling a set of reachable under arbitrary sequences of actions. Approximate POMDPs solutions can be obtained efficiently by using PBVI algorithm [19]. Ideally, the algorithm selects belief points that are spread evenly across the reachable belief space to cover as much reachable space as possible within a given horizon.

Successive Approximations of the Reachable Space under Optimal Policies (SARSOP) is another algorithm that computes the optimal policy on a range of optimally reachable belief space. SARSOP is proven to improve computational efficiency when performing simple robotic tasks including navigation [19]. Finding a range of reachable belief space is the key for this algorithm to solve for an optimal policy. However, finding the range that is close to the optimal value function is difficult even the size of the belief space is polynomial.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Several works have conducted on separated POMDP model into a hierarchy of processes to achieve much simpler computations. Hierarchical POMDP (H-POMDP) is capable for collaboration of human and mobile robot to achieve task objectives together. Multi robot collaboration is also achievable using the H-POMDP formulation by adding another layer for communication between robots. H-POMDP consists of three levels which is high-level for visual sensing, intermediate-level for information selection and low-level for information processing [3]. However, a significant amount of data and modelling algorithms have to be coded manually.

2.4 Perseus algorithm

One of the extension to the PBVI algorithm is the Perseus algorithm [32]. The Perseus algorithm performs random exploration in the belief space, then samples an action and observation to update the belief state by running several trials. The trials continue to get a large number of points over the belief space. For each successive iteration, Perseus improves the approximation of value function by performing a one-step backup of each belief. During each iteration, Perseus improves standard PBVI by omitting the improved beliefs by another backup.

2.5 Summary of Path Planning technique

In this chapter, autonomous mobile robot is more preferred for the industry because of the automaticity and absence of human operator. The POMDP model provides a powerful framework for modelling uncertainty and also predicting for close future. The PBVI algorithm can effectively compute belief points over a belief space to achieve near optimal outcome. This thesis discusses POMDP model based on Perseus algorithm to achieve the objective effectively in terms of execution time and success probability. The method is applied in path planning and navigation of autonomous mobile robot in an enclosed area such as a warehouse.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHAPTER 3

METHODOLOGY

The aim of the POMDP model is to solve autonomous mobile robot navigation uncertainty in a partially-structured environment. The mobile robot has to plan the best route from the start to a specified destination. Besides planning the shortest distance to a destination, the logical path is also considered. The robot has to continue to the next destination without returning to the starting point, until it has achieved the objectives. For autonomous mobile robot, the robot is always waiting at the starting point for new instructions.

When the state of the mobile robot is too large, solving for a policy requires tremendous time and computational power. Perseus is an extension to PBVI algorithm that is able to solve large state spaces without compromising time and computational power. This chapter discusses POMDP model, PBVI algorithm and Perseus to be applied in a mobile robot.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

3.1 The POMDP model

In the real world where mobile robot navigation is concerned, decision-making is the fundamental problem for the robot. The mobile robot has to determine the best action during the decision-making processes to achieve an optimal reward or accomplish the main objectives. When the environment around the robot changes dynamically, observations

need to be included into decision-making process. The problem is, the mobile robot has to consider the rewards after a sequence of action, which gives rise to sequential planning in a stochastic environment. POMDP serves as a powerful framework developed to account for the problem. A mobile robot will be represented as an agent from here onwards.

The objective of POMDP planning is to discover a policy π to select an action for the agent. The policy defines how the agent should act in order to maximize the rewards. There are several types of policies, history-dependent or Markov, stochastic or deterministic [27]. In POMDP formulation, the observation often depends only on the current state of the process, regardless of the history. Also, including histories into the process can be an exhaustive task, so the belief state is used in the space of probability distributions over states. POMDP models with belief states can be generalized into a belief-space MDP models. This formalism is widely used in the POMDP to model the agent's navigation. We focus on applying discrete and finite state space and action space. Continuous state space is also used in POMDP to scale up the algorithm to simulate a near actual environment [18].

POMDPs provide a framework for sequential planning to allow more forms of uncertainty into the process. The system states of the POMDP model is represented by belief states that are used for decision-making [16]. A POMDP [1-3, 15-17] is formally denoted as a tuple <*S, A, T, Z, O, R, b*, γ>. Variables in capital letter is the complete set variable in an environment, small letter denotes a certain set of variable used for calculation. The variables are defined in Table 3.1.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

When an agent does not know the exact state, the agent can only act depending on observations it can perceive. However, the sensor of the agent may not give accurate observations of the state. The agent has to assign a probability distribution over the state known as the belief state *b*. The probability of the belief state assigned to an actual state is written as *b*(*s*). The agent must update the current belief state to a new belief state for the actions taken and observations made so far. A technique called recursive function is used to calculate the new belief state b' from the previous belief state and new observation. The new belief state is given by

$$b'(s') = \alpha P(o|s') \sum_{s} T(s'|s,a)b(s), \tag{3.1}$$

Table 3.1: Variables of a general POMDP model.

Variable	Name	Description
S	State space	A discrete and finite set of all system states, which are observable and unobservable, that represents the environment where the robot acts.
A	Action space	A discrete and finite set of actions at each time instant.
T	State transition	A probability function that passes the current state to the next state. The value is within an interval [0,1]. It is defined as $T: S \times A \times S' \rightarrow [0,1]$. Also, $\sum_{s \in S'} T(s, a, s') = 1, \forall (s, a)$. The notation S' is the subsequent state S. Notation $T(s, a, s')$ is the state transition in current state s given action a moving into next state s'.
Z	Observation space	A finite set of observations. The observations include noisy inputs of the true state of the environment through the robot sensors.
O	Observation function	A function that represents the conditional probability given the action and the subsequent state. The function depends on the triplet (z, a, s').
R	Reward function	Immediate reward function that assigns a real value executing action A in state S. Negative reward represents a cost. The function directs an agent towards the goal location. Also defined as R: $S \times A$.
b	Belief state	The agent's knowledge or belief of the state of the environment. It is a probability distribution over all possible states S.
γ	Discount factor	A real value within the interval [0,1). An infinite sequence becomes finite ensures the algorithm converges to a final value.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

where α is a normalizing constant that makes belief state sum to 1. The subsequent belief state takes the summation for $s = 0, 1, \dots, S$ in an environment the agent is exploring. The agent does action according to its current belief state, not the actual state. This means that the optimal policy $\pi^*(b)$ maps belief states to actions. In fact, the action changes subsequent belief state when the agent observed the outcome of its action. Hence, action can be considered as one of the performance of the agent.

Given the current belief state b and action a, we can calculate the probability the agent would reach in the subsequent belief state b'. We do not know the subsequent observation yet, so the agent might reach in one of several possible belief states b'. The probability of observation o given that action a was performed in belief b is given by

$$P(o|a,b) = \sum_{s\prime} P(o|a,s',b)P(s'|a,b)$$
$$= \sum_{s'} P(o|s') \sum_{s'} P(s'|s,a)\, b(s). \quad (3.2)$$

To find the transition probability of mapping b to b' given action a as $T(b'|b,a)$, we get

$$T(b'|b,a) = T(b'|a,b) = \sum_{s\prime} T(b'|o,a,b)T(o|a,b) \quad (3.3)$$
$$= \sum_{o} T(b'|o,a,b) \sum_{s\prime} T(o|s') \sum_{o} T(s'|s,a)b(s). \quad (3.4)$$

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Equation (3.4) can be used as the transition model for the belief state. The reward function for belief states is

$$\rho(b) = \sum_{s} b(s)R(s). \quad (3.5)$$

The probability $T(b'|b,a)$ from Equation (3.3) and (3.4) and reward function $\rho(b)$ from Equation (3.5) can represent an observable MDP on the space of belief states. The optimal policy for this MDP, $\pi^*(b)$, is also the optimal policy for the original POMDP. Hence, POMDP in the physical state space can be generalized into an observable MDP on the corresponding belief-state space. This is because we assume the belief states are fully observable to the agent. Figure 3.1 illustrates the process of a general POMDP algorithm. The algorithm will stop after it reaches a terminating condition, usually a convergence

criterion or within a limited time. Though, in mobile robot, it is usually programmed to do other task after reaching the goal state, such as placing down objects.

3.2 Value function

Value function is one of the characteristic of Markov decision processes. Finding an optimal policy can be immediately transformed into an optimization problem in terms of value functions. This results in a less complex optimality equations resolution than exploring the whole set of policies. We can use the Bellman equation for the belief-space MDP to generate the value function, *V*:

$$V = \max_{a \in A} R(b,a) + \gamma \sum_{b' \in B} \tau(b,a,b') V(b') \tag{3.6}$$

$$= \max_{a \in A} R(b,a) + \gamma \sum_{o \in O} T(o|b,a) V(b^{a,o}). \tag{3.7}$$

In both the finite and infinite horizon case, the value function *V* can be modeled almost closely as the upper envelope of a finite set of linear functions, known as α-vectors. Now, the value function can be written as $V = \{\alpha_1, \ldots, \alpha_n\}$ to define over the full belief of the process. The value at a given belief can be computed as:

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

$$V(b) = \max_{\alpha \in V} b \cdot \alpha, \tag{3.8}$$

where $b \cdot \alpha = \sum_{s \in S} b(s) \cdot \alpha(s)$ is the standard inner product operation in vector space.

The Bellman equation from Equation (3.6) serves as an important aspect in value iteration algorithm to solve POMDPs. If there are *n* states, then there are *n* Bellman equations corresponding to each state. However, the Bellman equation is not linear, because the “max” operator is not a linear operator.

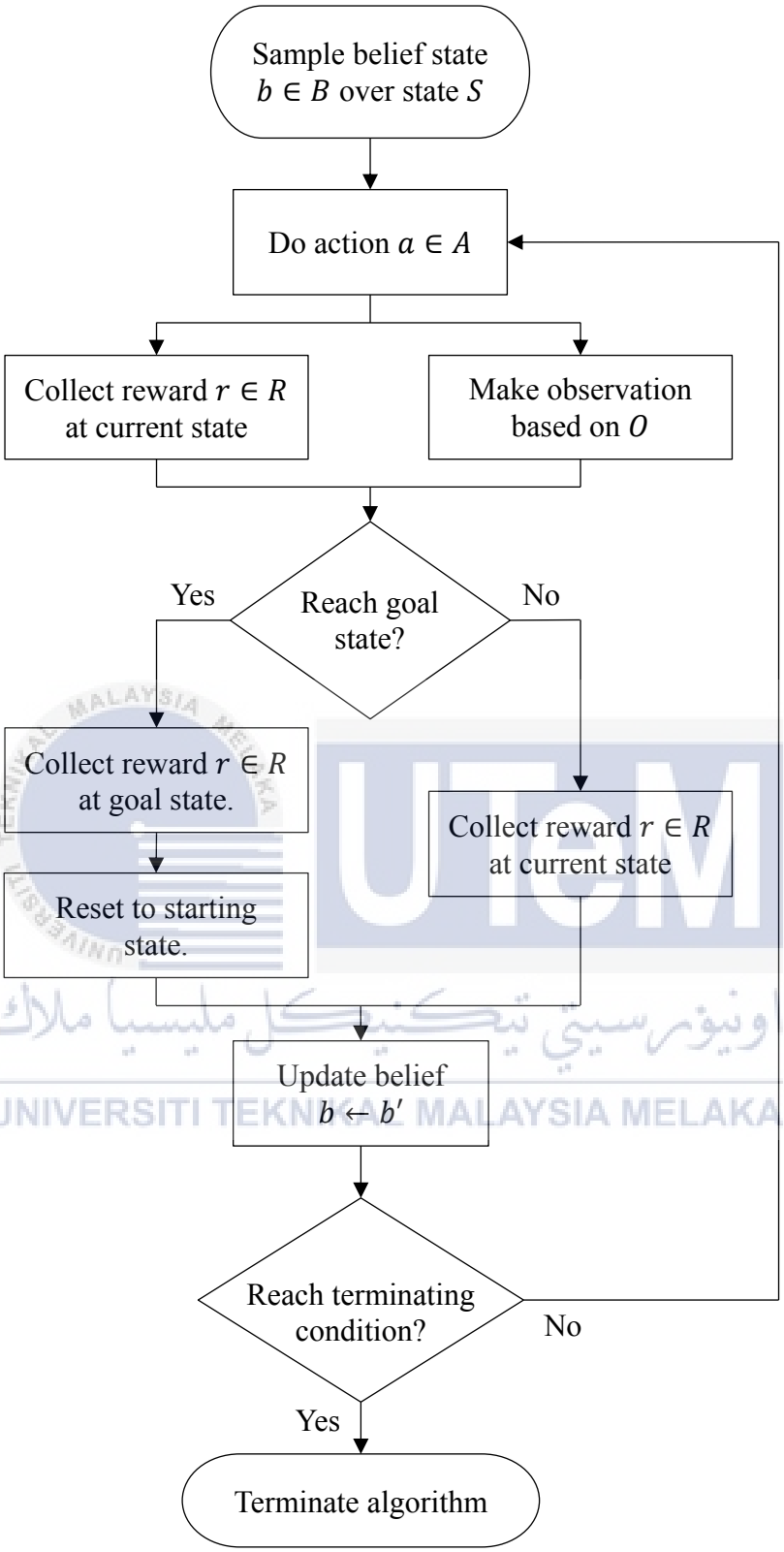


Figure 3.1: Flow chart of a general POMDP algorithm simulation.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

3.2.1 Belief-value mapping

The α-vector representation is particularly suitable for keeping a lower bound over the value function when it is updated incrementally. Because of the convergence property of the value function, the generated vectors will have higher values than the preceding vectors. So, the max operator will select the new vectors that dominate the preceding lower vector set *V*.

However, when the value iteration starts from an upper bound, the value of subsequent vectors will become lower than the currently existing vectors. These new vectors become useless, because the max operator of the value function selects the highest value. Thus, value iteration from the upper bound represented by α-vector is still an open problem [33].

3.2.2 Value iteration algorithm

One way to solve the non-linear equation is to use an iterative approach. Starting from an arbitrary value from the value function, we calculate a new value from the value function. Each time the new value is compared with the previous value, until we reach an equilibrium.

This iterative approach is updating the value function of the subsequent state from the previous state. The iteration step is a Bellman update of value function, given as

$$V \leftarrow \max_{a \in A} R(b,a) + \gamma \sum_{b' \in B} \tau(b,a,b')V(b'). \tag{3.9}$$

When the value function is performed iteratively, eventually we arrive to an optimum value function which is a unique solution of the Bellman equation. The value iteration algorithm can be considered as propagating information through the state space by updating the local value functions. The initial concept of the Value-Iteration algorithm is shown in Figure 3.2 [28].

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

function Value-Iteration (*pomdp*, ε) **returns** a value function

inputs: *pomdp*, a POMDP with belief states b, actions a, transition model $T(b'|b,a)$, rewards $R(b)$, discount γ

ε, the maximum error allowed in the value functions of any state

local variables: V, V', value functions for belief b, initially zero

δ, the maximum change in the value function of any state in an iteration

repeat

$V \leftarrow V'$; $\delta \leftarrow 0$

for each belief b **in** B **do**

$V'[b] \leftarrow \max_{a \in A} R(b) + \gamma \sum_{b' \in B} T(b,a,b')V[b']$

if $|V'[b] - V[b]| > \delta$ **then** $\delta \leftarrow |V'[b] - V[b]|$

until $\delta < \varepsilon(1-\gamma)/\gamma$

return V

Figure 3.2: The value iteration algorithm for calculating value functions of states.

To illustrate the value iteration concept, a POMDP problem 4×3 grid is used to explain the concept. The problem consists of 11 states, 3 actions: up, left and right, and 2 observations: nothing and goal state. From Figure 3.3, state (2,2) is an obstacle. At state (4,3) there is a reward of +1, while at state (4,2) is a negative reward of -1. The agent starts at state (1,1) is going after for the reward +1, and must avoid negative reward -1 at all costs. Action of the agent is stochastic, it has a probability 0.8 heading to intended direction, and probability 0.1 moving to either one of two directions accidentally. After the agent reaches either +1 or -1 state, it will be reset to the start state.

Every state contains a value V_n, where n is the number of iteration. From iteration $n = 0$, algorithm starts with an arbitrary value (0 in this case) at all states, as shown in Figure 3.2. In the next iteration, value function of the adjacent states to the reward state is computed first. And in the subsequent iterations, the second nearest and adjacent states to the reward state are computed, and so on, as illustrated in Figure 3.3. Value V is computed from Equation (3.7), is assigned into every state. The value at each grid will be improved iteratively. At a sufficient nth iteration, value V arrives to the optimum value V^*, one of the

criterion that stops the algorithm. Figure 3.5 shows the optimum value when the algorithm meets stopping criterion [28].

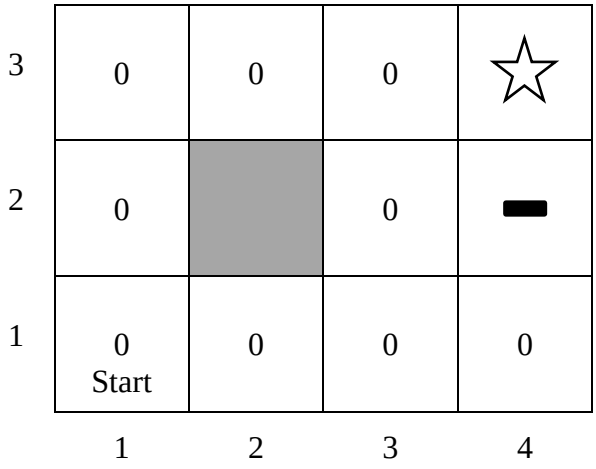


Figure 3.3: 4×3 grid world. All states start with value 0.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

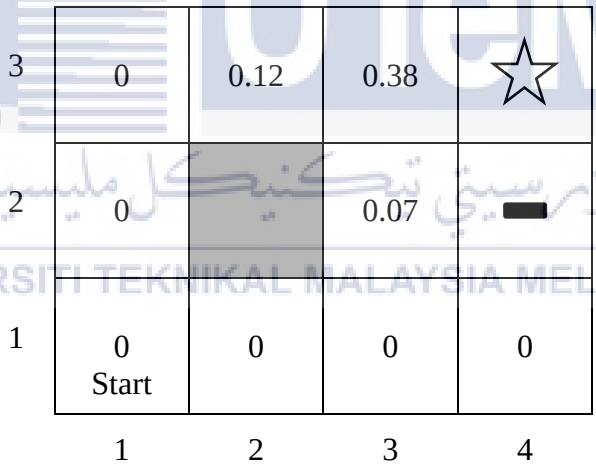


Figure 3.4: Value of the states after the first few iterations. In the next iteration, states (1,3) and (3,1), along with the updated states previously, will be updated at once. While the other states that are not updated, which are (2,2), (1,1), (2,1) and (4,1) will remain until subsequent iterations.

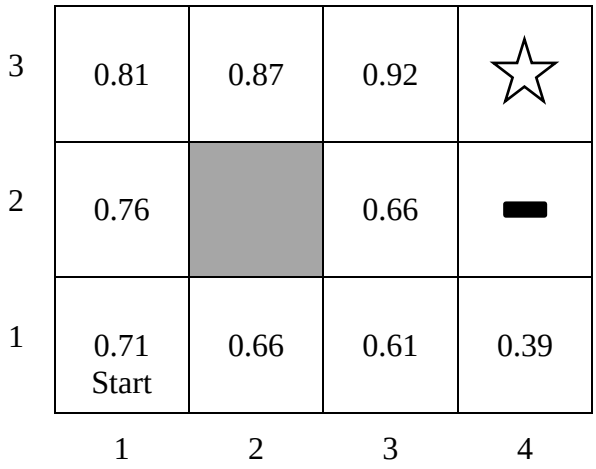


Figure 3.5: Optimum value V^* at each state when the algorithm reaches a stopping criterion.

3.2.3 Convergence of value iteration

The value iteration will eventually arrive to a unique set of solution of the Bellman equation. The error of value functions over the states can define when to terminate the algorithm so the algorithm does not have to run forever.

The convergence of value iteration is basically the notion of a contraction. Contraction is a function of an argument that when given two different input values, the outcome values of the argument will get closer together to a fixed point by a constant factor. After the argument reaches the fixed point, repeated application of a contraction on the argument will always stay at the same fixed point.

The error of value functions is the difference between the value function at the ith iteration over the belief states V_i and the original value function V. The error bound can be related to the size of the Bellman update on any value iteration. The optimal policy can only be computed when the error is less than a specified value. For each iteration, the error is reduced by a factor of γ. Thus, the convergence is fast given that γ is small enough. If $\gamma \approx 1$, however, the number of iterations needed to run to reach the error bound becomes exponentially larger. The difference of the value function, δ in any state of an iteration is the terminating condition, which is mentioned in Figure 3.2, given as

$$\delta < \varepsilon(1-\gamma)/\gamma.$$

3.2.4 Piecewise Linear and Convex Function

An interesting characteristic of the POMDPs is the piecewise linear and convex (PWLC) function [28]. Let the value functions of the belief space be the vectors α in a hyperplane of belief space. Then, we can note that the vectors vary linearly with the belief b in a hyperplane of the belief space, which is

$$V(b) = \max_{\alpha \in A} \sum_{s \in S} b(s)\alpha(s) \tag{3.10}$$

$$= \max_{\alpha \in A} b \cdot \alpha. \tag{3.11}$$

In Figure 3.6, the line segments of vectors α_0, α_1 and α_2 are drawn in bold, which are the dominating vectors. While α_3 is dominated by other vectors and it does not influence the value function. Hence, vector α_3 can be eliminated to save memory space in the agent. The 'max' function works this out by selecting the maximum vector α from all vectors A generated in the current belief space.

Using the PWLC function, the optimal value function is PWLC in a finite horizon problem. There are several advantages using the PWLC function:

- The optimal value function can be calculated using smaller number of iterations.
- The optimal value function can be represented by a finite number of vectors.
- The optimal policy can be computed in a shorter period.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

3.3 The PBVI algorithm

In a large belief-state space, depending on the value iteration algorithm that computes every belief state is not very efficient. This problem can be reduced by using the PBVI algorithm. The PBVI algorithm is able to approximately solve the value function on a large POMDPs rapidly. The size of the vectors must be limited when performing value iteration over the belief state space.

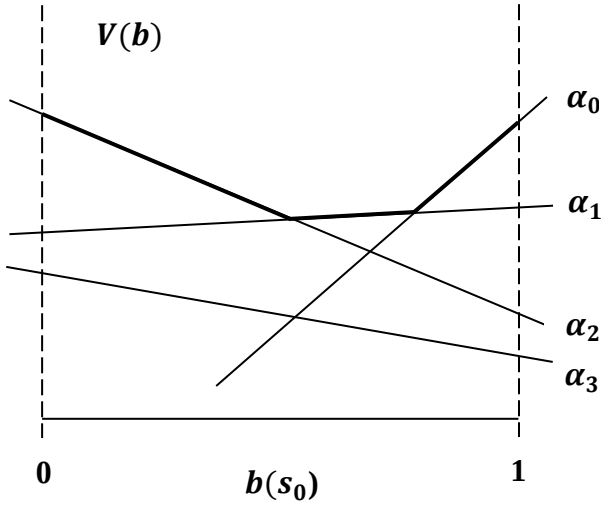


Figure 3.6: PWLC. In this PWLC example, there are 4 vectors of parameters α_i in the belief space hyperplane. The optimal value function is the line segment drawn in bold [27].

The PBVI algorithm selects several points over the belief-state space in a finite horizon, then solves the selected points by initializing a separate α-vector to update the value function. PBVI also exhibits the PWLC property of the optimal value function. The algorithm first computes an initial set of belief points by applying *backup* operation from Equation (3.15). Then, new belief points are generated and these points are used to find a new solution for the expanded set.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

The value function update procedure is manipulated to compute a value at a certain belief point b after a Bellman backup over the given value function V. After a complete Bellman backup process, an optimal α-vector can be computed on the belief state.

By using the notation $R(b,a) = r_a \cdot b$, we can rewrite Equation (3.9) as,

$$V'(b) = \max_{a \in A} r_a \cdot b + \gamma \sum_{o \in O} P(o|b,a) V(b^{a,o}) \tag{3.12}$$

$$= \max_{a \in A} r_a \cdot b + \gamma \sum_{o \in O: T(o|b,a) > 0} \max_{\alpha \in V} b \cdot \alpha^{a,o}, \tag{3.13}$$

where

$$\alpha^{a,o}(s) = \sum_{s' \in S} \alpha(s') O(a,s',o) T(s,a,s'). \tag{3.14}$$

We can make a compact backup operation, denoted as $backup(V, b)$ that generates a new α-vector for a specific belief b:

$$backup(V, b) = \underset{\alpha_a^b : a \in A, \alpha \in V}{\operatorname{argmax}} \; b \cdot \alpha_a^b, \tag{3.15}$$

$$\alpha_a^b = r_a + \gamma \sum_{o \in O} \underset{\alpha^{a,o} : \alpha \in V}{\operatorname{argmax}} \, b \cdot \alpha^{a,o}. \tag{3.16}$$

The dominated α-vectors are implicitly pruned twice at each argmax operation. The value $\alpha^{a,o}$ is independent of b, so it can be reused for backups over other belief points in the set of belief states B. The complexity of computing $\alpha^{a,o}(s)$ is $Z(|A|\times|O|\times|V|\times|S|^2)$. Computing α_a^b requires computation of all relevant $\alpha^{a,o}$ and the summation is only $Z(|S|\times|O|)$. The $backup$ operation for α_a^b requires another $Z(|S|)$ operations for the inner product. Hence, the total number of complexity of the point-based backup requires $Z(|A|\times|O|\times|V|\times|S|^2 + |A|\times|S|\times|O|)$. Complexity analysis is important in determining the performance of an algorithm, but it is beyond the scope of this thesis.

In POMDPs, the optimal policy with respect to the value function V defined over the belief space is given as,

$$\pi_V(b) = \underset{a}{\operatorname{argmax}} \, R(b, a) + \gamma \sum_{o \in O} P(o|b, a) V(b^{a,o}). \tag{3.17}$$

Calculating $\pi_V(b)$ for the current belief state b requires computing all $|A|\times|O|$, with a cost of $|S|^2$ for each updated b. Then, calculating the value at each updated b requires $|S|\times|V|$ operations using the α-vector representation.

A generic PBVI algorithm is approached using the two ideas above, which are limiting the value function size at a finite, reachable belief subset, and optimizing the value function using point-based technique. PBVI algorithm is shown in Figure 3.7 [31].

The algorithm requires input an initial belief point set B_{init}, an initial value function V_0, the number of desired expansions N, and the planning horizon T. The initial belief point B_{init} is usually the initial belief b_0. Initial value function V_0 is set to a lower bound on V^*, which will be explained in the next section. The algorithm uses $backup$ operation described in Equation (3.15). An operation $expand$ is introduced for the moment for which it selects belief points at random.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

function PBVI (B_{init}, V_0, N, T) **returns** a value function

inputs: B_{init}, V_0, N and T

local variables: V, V_0, value function in PBVI

B, belief state

$B = B_{init}$

$V = V_0$

for N expansions **do**

for T iterations **do**

$V = backup(V, b)$

$B_{new} = expand(V, b)$

$B = B \cup B_{new}$

return V

Figure 3.7: Point-based value iteration (PBVI) algorithm.

3.3.1 Initial value function

To calculate a value function in Equation (3.12), we need to initiate some initial value function to be updated. Ideally, choosing an initial value function as close as possible to the optimal value function V^* can reduce the number of iterations before convergence. The value function is required to be a lower bound on V^*. Finding the lower bound in value function is given by,

$$R_{min} = \min_{s \in S, a \in A} R(s, a), \tag{3.18}$$

$$\alpha_{min}(s) = \frac{R_{min}}{1 - \gamma}, \tag{3.19}$$

$$V_0 = \{\alpha_{min}\}. \tag{3.20}$$

This method always computes the minimum reward in every step, and relies on the convergence of $\sum_{i=[0,\infty)} \gamma^i$ to $1/(1-\gamma)$. Solution from this step will always be a lower bound on the exact solution. So, failing to compute an α-vector can only lower the value function.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

3.3.2 Parameters affecting PBVI algorithm

The parameters of the PBVI algorithm influence the resulting value function. They are a compromise between the computational power and the accuracy of the point-based approximation.

One of the parameter is the number of belief points in the set of belief space B. The size of the belief points affects the time taken to update the belief points. The approximation of the value function also depends on the number of belief points. So, a balance is needed to achieve between accurate approximation and the computation time of a value function update. The number of belief points in B is controlled by the algorithm to achieve any one of the performance criterion.

Another parameter of the algorithm is the number of point-based backups in each value function update. It is not necessary to compute all backups in B. Increasing the number of backups increases the computational burden, however, the value function can reach to V^* more quickly. The third parameter is the removal of dominated vectors. When value functions are constantly updated, they will take up memory space quickly. The agent may also take more time to compute all the vectors included dominated ones. Hence, it is important to keep a limited number of vectors in the value function.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

3.4 Perseus algorithm

In PBVI algorithm, it focuses on the smart expansion of belief state B, because B influences the size of value function V in PBVI. However, PBVI algorithm requires significant effort when expanding B, which is inefficient in large state spaces.

Perseus algorithm is introduced to improve belief point collection and selection from the original PBVI algorithm [32], [33]. First, the Perseus algorithm randomly explore the belief space of an environment and collect a set of reachable belief points B, which remains fixed throughout the algorithm. The value function V is set to initial value function

V_0 as in Equation (3.20). Perseus then computes the set of non-improved points B consisting b whose new value V' is still lower than V_0.

The algorithm performs the $backup$ operation iteratively until the value function converges. At each iteration, the algorithm starts by setting new belief $B' = B$ and a new empty value function $V' = \phi$. Perseus selects a belief point b from the new belief space B' to compute a new vector $\alpha' = backup(V, b)$ using Equation (3.15), which becomes

$$\alpha' = \underset{\alpha_a^b : a \in A, \alpha \in V}{\operatorname{argmax}} \; b \cdot \alpha_a^b. \tag{3.21}$$

If $b \cdot \alpha' \geq V_0(b)$, then α' is added into V'. If $b \cdot \alpha' \leq V_0(b)$, then the previously computed α is added into V'. When α improves the value of belief points b from B', all the points b are removed from B', and the set B' shrinks. The iteration ends when all belief points b from belief space B' are improved, which makes $B' = \phi$ one of termination condition. Figure 3.8 [33] is the full procedure, and Figure 3.9 [34] is the flow chart of the algorithm.

3.5 Experimental setup

Several well-known POMDP problems that are described here are 1d [29], part-painting, tiger-grid and hallway2 [30], [31].

A simple POMDP problem known as 1d problem is shown in Figure 3.10. In 1d problem has four states, where state 2 is the goal state. An agent is in one of the states. It has two actions, left and right that move it one state in any of the direction. When the agent moves into a wall, it stays in the state it was in. When the agent moves into the goal state, no matter what action it takes, it is moved with equal probability into state 0, 1 or 3 and receives reward +1. The problem is easy if the agent can observe which state it is in, but is more difficult where it can only observe whether it is currently at the goal state. Actions are deterministic, which means the agent will move to the direction it intends to.

The second POMDP problem is part-painting problem. A part needs to be painted and shipped if it is not flawed, or rejected if it is flawed. There are three state variables in this problem: flawed, blemished or painted. The full state space is the cross product of the

function Perseus

$B \leftarrow RandomExplore(n)$

$V \leftarrow PerseusUpdate(B, \phi)$

function RandomExplore(n) **returns** B

$B \leftarrow \phi$

$b \leftarrow b_0$

repeat

Choose $a \in A$ randomly

Choose $o \in O$ following the $P(o|b, a)$ distribution

Add $b^{a,o}$ to B

$b \leftarrow b^{a,o}$

until $|B| = n$

function PerseusUpdate(B, V) **returns** value function

repeat

$B' \leftarrow B$

$V' \leftarrow \phi$

while $B' \neq \phi$ **do**

Choose $b \in B'$ *// Choose an arbitrary point in B to improve*

$\alpha \leftarrow backup(V, b)$

if $b \cdot \alpha' \geq V(b)$ **then**

$B' \leftarrow \{b \in B' : b \cdot \alpha < V(b)\}$

$\alpha_{old} \leftarrow \alpha$

else

$B' \leftarrow B' - \{b\}$

$\alpha_{old} \leftarrow \text{argmax}_{\alpha \in V}\, b \cdot V$

$V' \leftarrow V' \cup \{\alpha_{old}\}$

$V \leftarrow V'$

until V has converged

Figure 3.8: Perseus algorithm.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

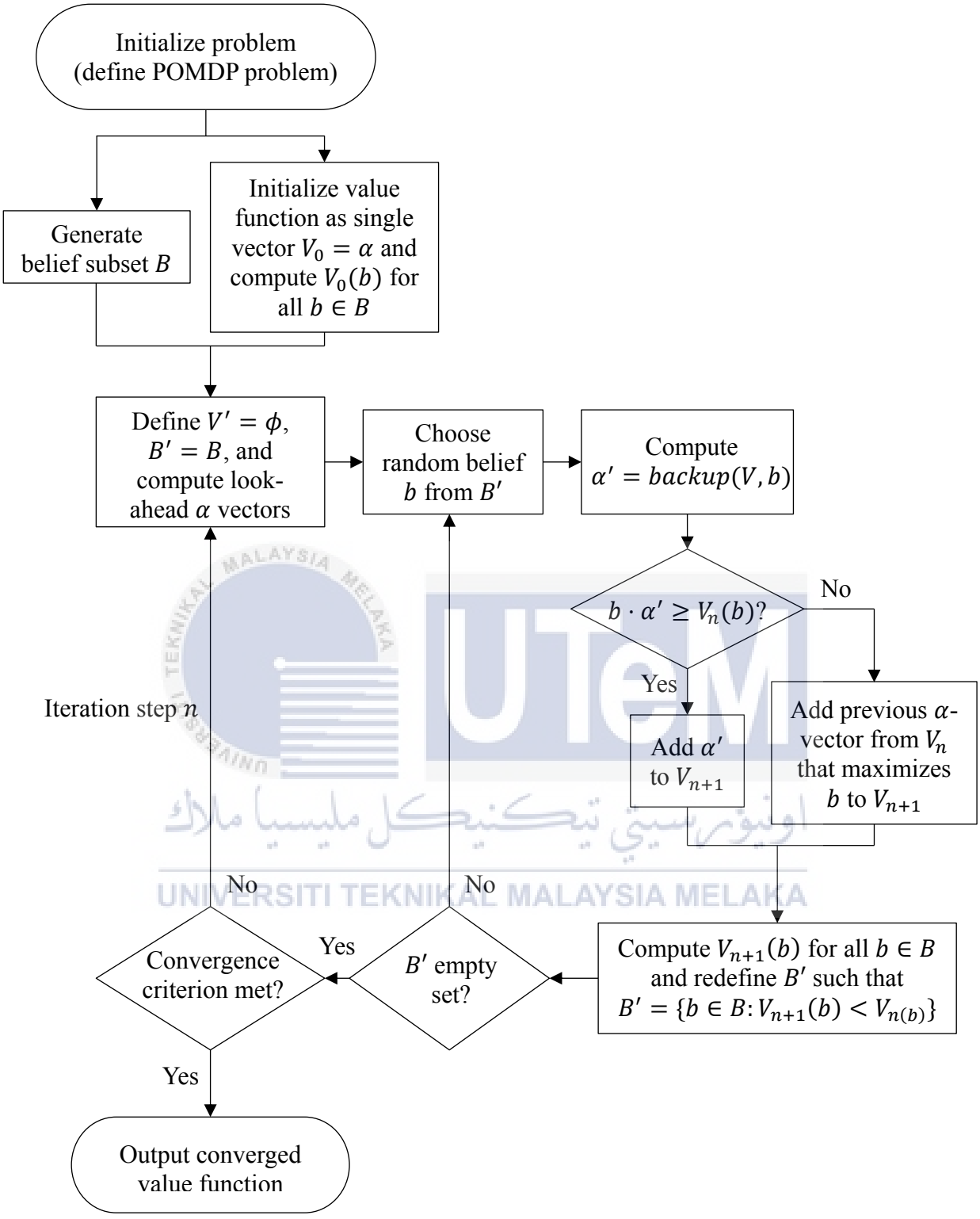


Figure 3.9: Flow chart of Perseus algorithm.

three variables which totals to 8 states. However, we assume that the part start in one of the two states, so four states can be discarded, since it is impossible for a part to reach that state from the two possible starting states. The only two possible starting states are either the part is flawed and blemished or the part is not flawed and not blemished. In both cases, the part starts as not painted. There are four possible actions: inspect, paint, ship or reject the part. The action for this problem is stochastic. There is a probability 0.9 of painting successfully and 0.1 of painting unsuccessfully. Shipping a part get reward +1 while rejecting a part get cost -1, and the state being reset to correspond to a new part starts at any one of the two possible starting states with equal probability. Inspecting a part does not change its state, it observes the part to give two observations: blemished and unblemished. A part is inspected successfully with probability 0.75 and incorrect observation with probability 0.25.

The following two POMDP problems (hallway2 and tiger-grid) are based upon the same framework. Then, any deviations from the basic framework is discussed for each problem. The basic framework is that there is an agent wandering around a map. The map of the agent is assumed to be discretized so that there are a finite number of locations the agent could be. At each location, the agent identifies 4 orientations: forward, backward, left and right. The actual state of the map is the number of possible orientations the agent can have in a location. There is a single goal location in a map, which is the objective of the agent. Since the objective of the agent is to achieve the goal, the goal location requires only one state, because the orientation of the agent is not concerned when it achieves the goal. The agent will be reset randomly to any other location other than the goal location after achieves the goal.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

The agent has a finite number of actions, and the actions are stochastic. The actions of the agent are the movements: forward, turn around, turn left, turn right and no operation (stay in place). These actions are relative to the current orientation of the agent. The no operation action always succeeds in leaving the state unchanged. The agent moves forward successfully with probability 0.8, turn left or right successfully with probability 0.7 and turn around successfully with probability 0.6.

Moreover, the agent is equipped with noisy sensors which and they are only capable of sensing adjacent wall. It can observe 4 directions: forward, backward, left and right, which are relative to the current orientation of the agent. The sensors detect a wall

correctly with probability 0.9 and mistakenly detect a wall with probability 0.05. The sensors are installed independently. The probability for an observation is computed by multiplying the four individual probabilities. Thus, there are 16 possible observations in each state by observing 2 possible values: wall or no wall, over 4 independent sensors. The goal state gives a reward of +1 and other states give zero reward. An extra observation is included to observe the goal. This observation is possible in the goal state and occurs deterministically.

The third POMDP problem is the tiger-grid problem (also known as 33 States). This problem differs from the basic framework in two ways. First, the agent starts from and resets to two specific belief states (state 3 and 4) uniformly as shown in Figure 3.11. Second, two upper corner states (state 0 and 7) are cost -1 and state 4 is reward +1.

The fourth POMDP problem is the hallway2 problem (also known as 89 States). This problem conforms exactly to the configuration of the basic framework laid out previously. The environment map is shown in Figure 3.12.

There are two approximate POMDP solving software applied in this thesis. Third party Matlab functions is used to implement the Perseus randomized point-based approximate value iteration algorithm for POMDP [32]. The software takes as input a POMDP problem specification in POMDP file format [30]. The parameters to simulate all four POMDP problems are shown in Table 3.2.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Computation time, denoted by time from here onwards, is obtained by subtracting Starting time from the first iteration by CPU time taken at subsequent iterations. The outcome value is obtained by retrieving SumV array data. The outcome reward is obtained by executing function SampleRewards for each iteration. At each iteration, average reward is computed. The outcome number of vectors is contributed to the computation of value at each step. The resulting value, reward and number of vectors are plotted against time as loglog graph.

The second software is an approximate POMDP planning (APPL) toolkit that implements C++ environment for SARSOP algorithm [19]. It takes as input a POMDP model in the POMDP or POMDPX file format and produce a policy file. Then, simulation is performed based on the policy file and repeat for 1,000 times. Computation time for both tiger-grid and hallway2 problems is limited to 30 minutes.

In this thesis, the operation for SARSOP consists of two parts: solver and simulator. The solver uses SARSOP algorithm to compute a policy for the POMDP model. The algorithm stops at termination conditions that can be an error for the desired output, a time limit or a user input Ctrl-C. In this thesis, the time limit of 30 minutes is set for the problem. When the termination condition is reached, the algorithm will stop and produce the policy file. The simulator performs simulation runs based on the policy file and computes the expected reward.

Perseus algorithm is implemented in Matlab, and SARSOP algorithm is implemented in C++. The experiments are performed on a PC with a 1.8 GHz Intel processor and 2 GB memory.

3.6 Summary

The POMDP model is a powerful framework developed for an agent to determine the decisions that leads to the main objectives while obtaining the highest reward possible in a partially observable environment. The agent generates a belief state over the state space based on an action made previously and current observation of the state. The value function is a special characteristic of Markov decision processes that reduces the complexity of optimality equations resolution. Point-based value iteration (PBVI) algorithm approximately solve the value function on a large POMDP rapidly by solving several belief points over the belief-state space. Perseus algorithm improves PBVI algorithm by collecting a set of reachable belief points, and is remained constant throughout the process. Perseus algorithm is the choice to run on several famous POMDP problems because of its fast computation speed.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Table 3.2: Parameters of all four POMDP problems for Perseus simulation.

Type of problems	Belief point	Discount factor	Maximum step	Computation time
1d	1000	0.75	251	200 s
Part-painting	1000	0.75	100	200 s
Tiger-grid	10000	0.95	100	1800 s
Hallway2	10000	0.95	251	1800 s

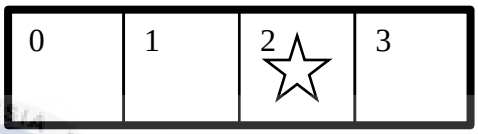


Figure 3.10: A simple POMDP problem – 1d.

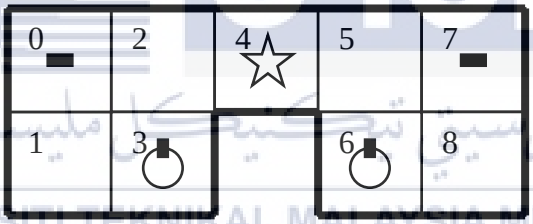


Figure 3.11: The tiger-grid problem.

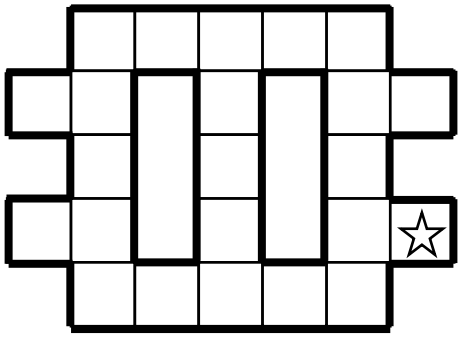


Figure 3.12: The hallway2 problem.

CHAPTER 4

RESULT AND DISCUSSION

The path planning task of an autonomous mobile robot is a comprehensive and a real-world problem which is a challenge using the POMDP model. Four known POMDP problems are studied and the results are plotted. The results are divided into three parts. The simulation setup is presented in Section 3.5. First, the number of action is varied with fixed number of state and observation. 1d and part-painting problem are compared in the first part. Second, the number of state is varied with a fixed number of action and observation. Tiger-grid and hallway2 problem are compared in the second part. Third, the performance of Perseus algorithm is compared with SARSOP algorithm.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

The result that will be studied are the value, reward and number of vectors. All results are plotted against time.

Value is calculated using iterative approach from the value to select the optimal action in each belief state. Equation (3.8) is used to compute the value over the full belief of the process. The equilibrium value is the optimum value function which is a unique solution of the Bellman equation. Equation (3.9) is used to compute Bellman update of value function. Equation (3.12) and (3.13) are used to calculate the value at selected belief points. In value - time graphs, time to reach optimal value and change of value is studied. Change of value is the gradient of the graph.

Reward is a real value offered to the agent after executing an action in a belief state. The reward here refers to the expected discounted reward. The agent reaches the optimal reward which is to collect the highest reward available in the problem. Reward for belief

states is calculated from Equation (3.5). Number of vectors is the number of point-based backups in each value function update. The number of vectors is computed from Equation (3.16). Equation (3.11) relates value function with number of vectors.

4.1 Comparison between 1d and part-painting problem

The following two problems are relatively small POMDP problem. They are compared with varying number of action, but same number of state and observation. The parameters of the problems are shown in Table 4.1. Results are plotted in Figure 4.1, Figure 4.2 and Figure 4.3.

Table 4.1: Parameters of 1d and part-painting problem.

Problem	State	Action	Observation
1d	4	2	2
Part-painting	4	4	2

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

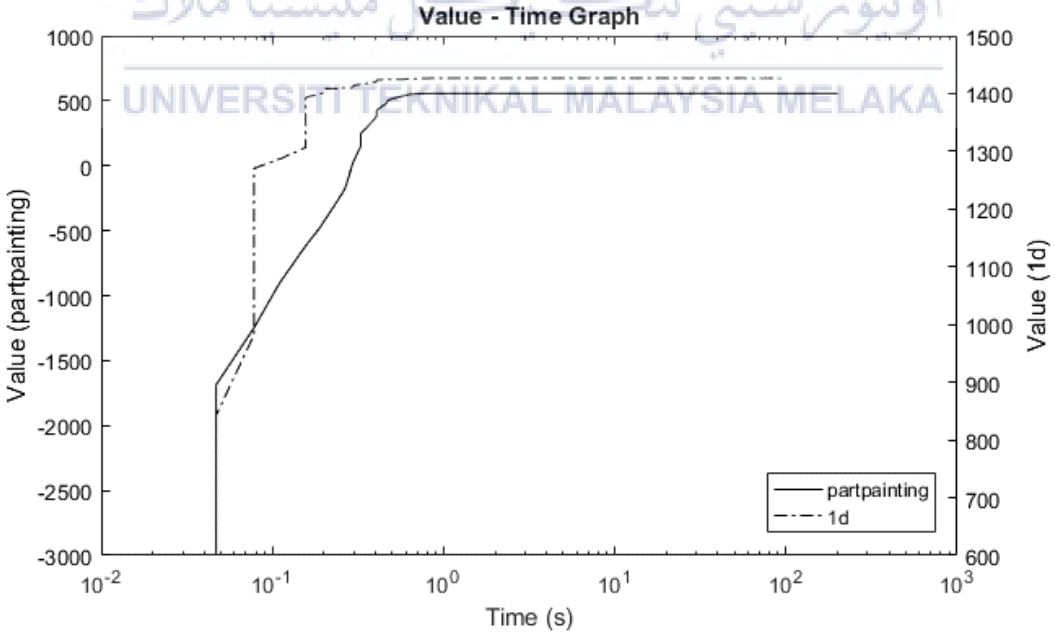


Figure 4.1: Value - time graph comparing part-painting and 1d problem.

Part-painting problem reaches optimal value at 0.8 s for value 556. 1d problem reaches optimal value at 0.6 s for value 1424. Change of value in 1d problem is greater than that in part-painting problem. When the number of action is lower, change of value is greater. In 1d problem, when number of action is lower, the agent has lower number of belief state to consider. So, the value rise faster among these particular belief states compared to the part-painting problem.

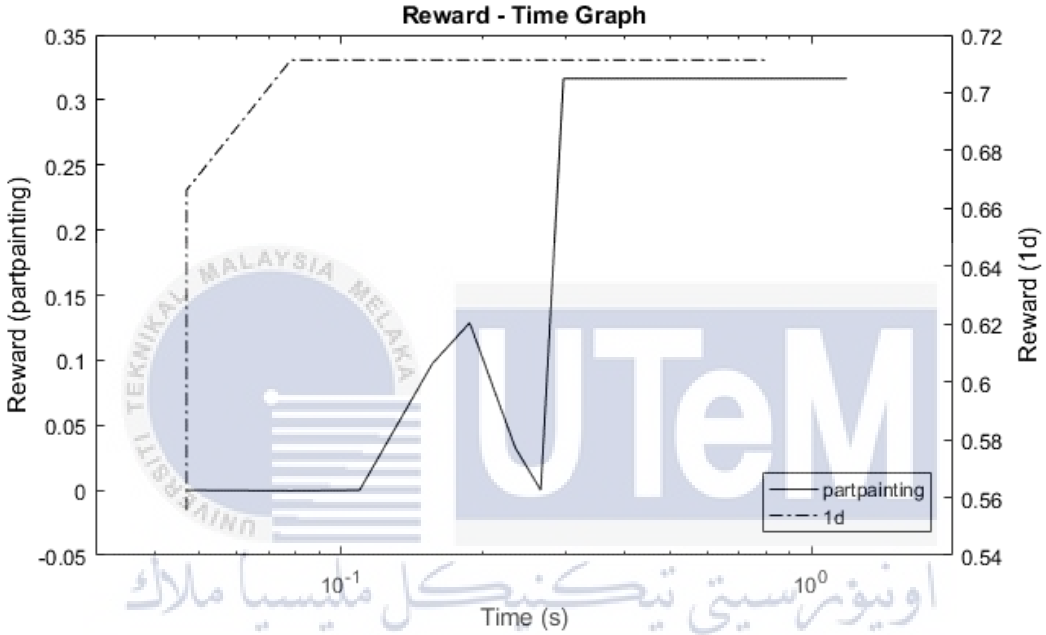


Figure 4.2: Reward - time graph comparing part-painting and 1d problem.

Part-painting problem achieves optimal reward at 0.29 s which is later than 1d problem which achieves optimal reward at 0.078 s. For part-painting problem, there is a sudden drop of reward beginning at 0.19 s. This is because of the cost in the problem. 1d problem does not have cost, while part-painting problem has several costs -1. In addition, in part-painting problem more actions tend to bring the agent to discover other possible states to collect the best possible rewards. However, during the discovery, the agent might encounter costs that plummet the previous accumulated reward.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

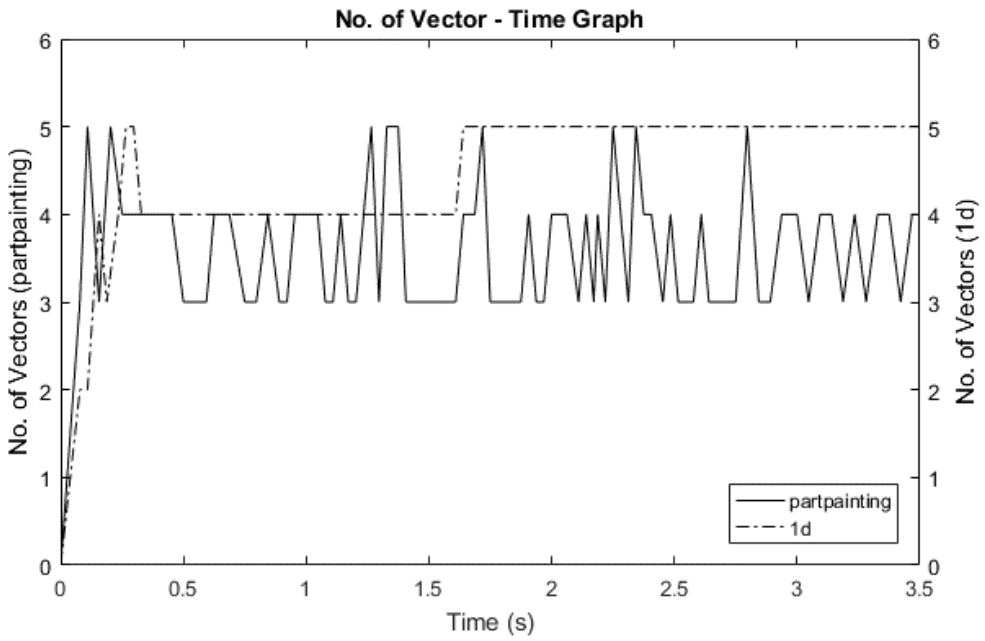


Figure 4.3: No. of vector - time graph comapring part-painting and 1d problem.

In part-painting problem, number of vectors range from 3 to 5 vectors, but it mostly stays at 3 and 4 vectors. In 1d problem, initially number of vectors was at 4, then it rises to 5 vectors. When having higher number of vectors, greater computational burden is needed to compute value function in 1d problem. Although computational burden is greater, the value function of 1d problem reach optimal value V^* more quickly, as shown in Figure 4.1.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Higher number of actions will bring the agent to explore other state. Under certain states, the computed value function may become the lower bound in its belief space hyperplane. So, the vector associated to the value function become dominated and is pruned to reduce complexity. Sometimes the states require lower amount of value function to achieve the optimal value. On the other hand, certain states computed more value functions at the upper bound to achieve the optimal value. Hence, the number of vectors in part-painting problem fluctuates when the agent is exploring other states. From Figure 4.1, it explains the optimal value in part-painting problem is lower than that in 1d problem due to the lower number of vectors in certain states.

4.2 Comparison between tiger-grid and hallway2 problem

The following two problems are large POMDP problem. The two problems are compared with varying number of state but same number of action and observation. The parameters of the problems are shown in Table. Results are plotted in Figure 4.4, Figure 4.5 and Figure 4.6.

Table 4.2: Parameters of tiger-grid and hallway2 problem.

Problem	State	Action	Observation
tiger-grid	36	5	17
hallway2	92	5	17

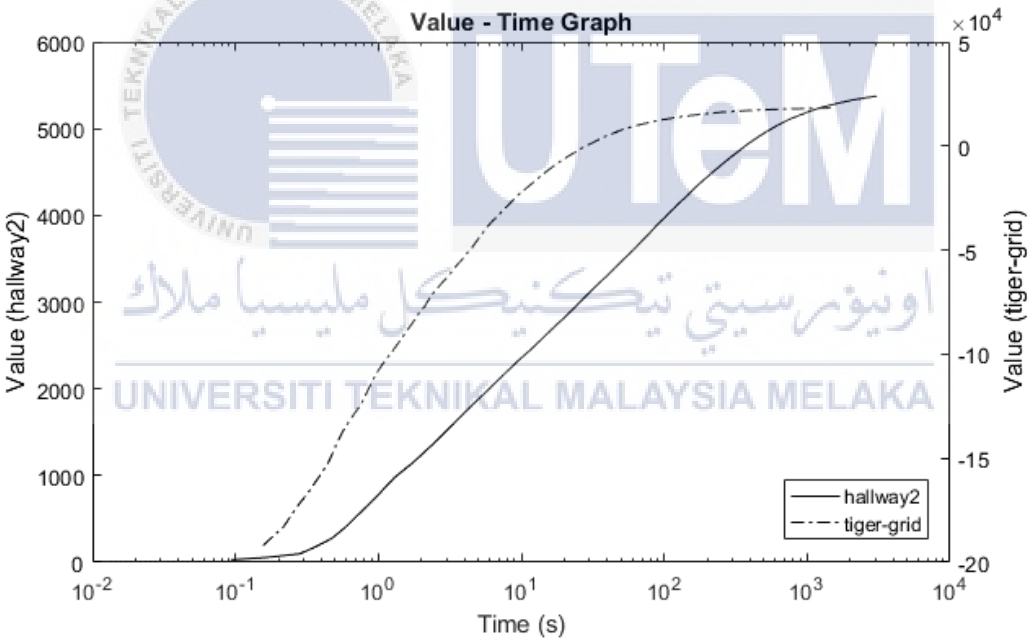


Figure 4.4: Value - time graph comparing hallway2 and tiger-grid problem.

Hallway2 problem takes significantly longer time to reach the optimal value. The result has been trimmed for a better view. The value is noticed to converge at 3081 s at value 5375. Tiger-grid problem reaches optimal value at 174 s at value 15111. Change of value in tiger-grid problem is greater than that in hallway2 problem. In tiger-grid problem,

the agent has lower number of belief state to consider, so the value rise faster among these belief states compared to the hallway2 problem.

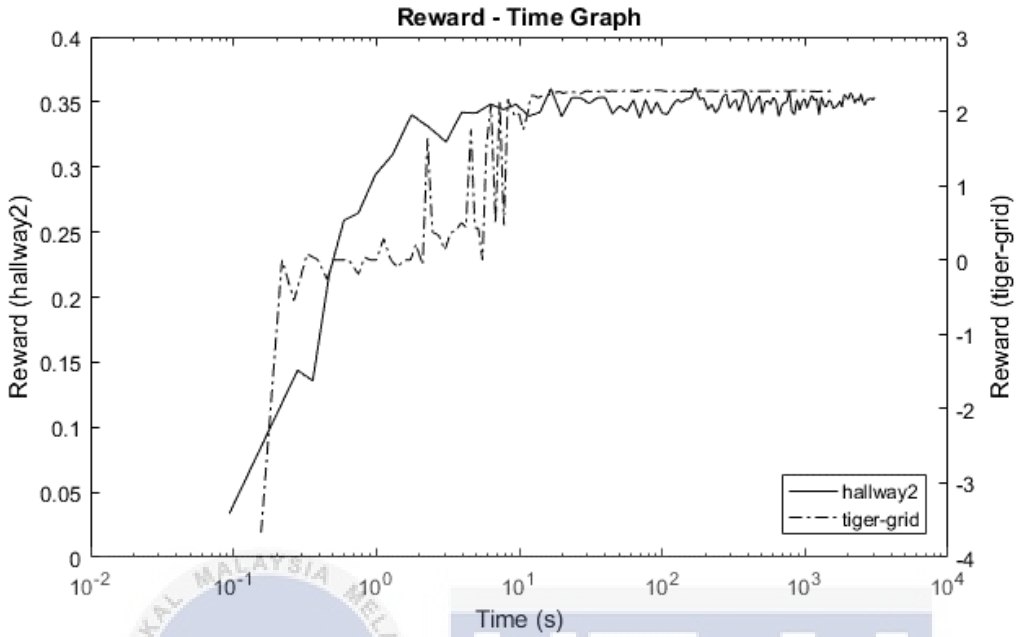


Figure 4.5: Reward - time graph comparing hallway2 and tiger-grid problem.

Hallway2 problem achieves optimal reward at 23 s and then it fluctuates slightly around reward 0.35. Tiger-grid problem achieves optimal reward at 12 s for reward 2.27 after significant fluctuations. Tiger-grid problem achieves optimal reward earlier than that in hallway2 problem. This is also because of the cost in the problem. Hallway2 problem does not have cost, while tiger-grid problem has two states having cost -1.

Initially, hallway2 problem rises to the optimal reward without much fluctuations, but tiger-grid problem fluctuates significantly because the agent in tiger-grid problem accidentally enters the cost states. Then, hallway2 problem fluctuates around the optimal reward, because the reward would change while the agent is exploring other states. For tiger-grid problem, the agent has lesser state to discover, so the reward changes slightly later in time.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

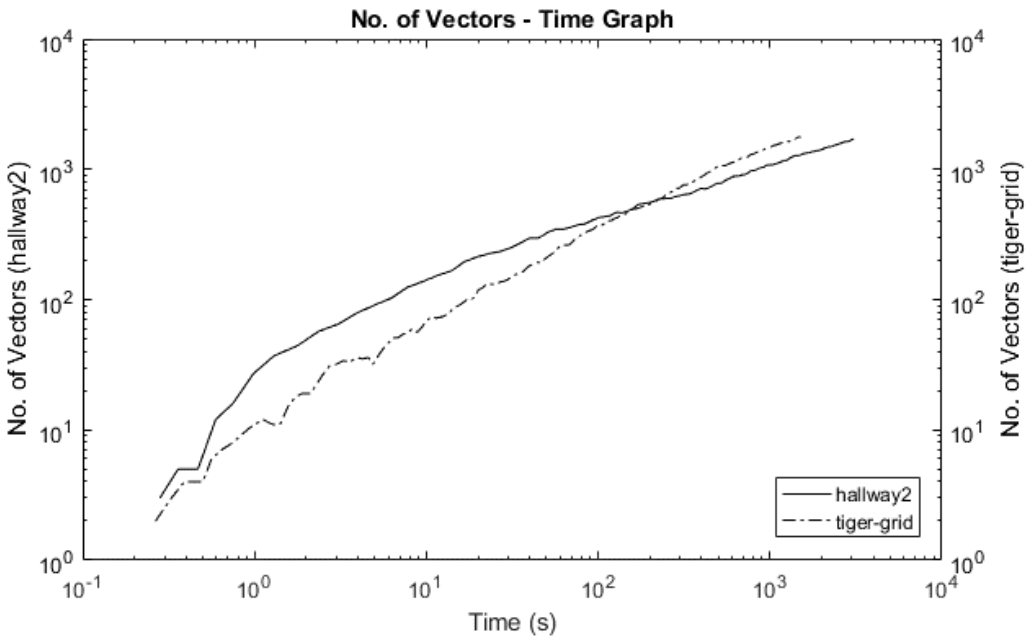


Figure 4.6: No. of vectors - time graph comparing hallway2 and tiger-grid problem.

Initially, number of vectors of hallway2 problem is higher than that of tiger-grid problem. Beginning 228 s, number of vectors of tiger-grid problem has surpassed hallway2 problem. Number of vectors of tiger-grid problem increases at a faster rate so the value function of the problem reaches the optimal value V^* more quickly, as shown in Figure 4.4.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

4.3 Comparison of Perseus and SARSOP algorithms

Both Perseus and SARSOP are extensions to PBVI algorithm to improve its performance in terms of belief point collection, reward and value function. Table 4.3 compares the performance of Perseus and SARSOP algorithms.

In this section, two POMDP problems: tiger-grid and hallway2 are compared. The other two problems: 1d and part-painting are not studied due to small state spaces and convergence is too fast. There are three performance criteria that are compared: expected reward, number of vector and belief point. The value function is not compared in the sense that SARSOP presents value function differently in lower and upper bound value, where

the optimal value would be in between the bounds. Moreover, the expected reward is a more important criterion to study when comparing POMDP solving algorithms.

Table 4.3: Performance comparison of Perseus and SARSOP.

POMDP problem	Algorithm	Expected Reward	Number of vector	Belief point
Tiger-grid	Perseus	2.27	1775	10000
	SARSOP	2.28	2347	5384
Hallway2	Perseus	0.35	1716	10000
	SARSOP	0.497	496	3304
	SARSOP (2 hrs)	0.503	705	4615

In tiger-grid problem, SARSOP performed slightly better than Perseus in terms of expected reward and number of belief points, though SARSOP requires more vectors. In hallway2 problem, SARSOP performed better than Perseus in all three performance criteria. Another simulation is also performed under the same environment in [19] by using a PC with 2.66 GHz and 2 GB memory for 2 hours to evaluate the performance. SARSOP tends to achieve better expected reward because it samples belief points near to the reachable belief space, in turn collects better rewards. In Perseus, user must assign belief points before the algorithmic operation, so it is fixed at 10,000 points for both problems. SARSOP assigns reachable belief points during algorithmic operation, so it will increase the belief points near to the reachable belief space to obtain more desirable results. Using Perseus will not encounter belief point explosion, while SARSOP will become computational intractable if the state space is scaled up.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

4.4 Summary

Higher number of actions makes the value to increase more quickly to the optimal value. Lower number of states helps the value converges faster to the optimal. More actions will bring the agent to discover other states for better rewards. However, more states cause the reward to fluctuate when the agent is exploring all the states. When the agent is exploring other states, number of vectors changes in respective states for the agent to reach the optimal value faster. Even though Perseus algorithm did not perform well as SARSOP, but Perseus algorithm is suitable to handle bigger states.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

The path planning method using Perseus algorithm to solve mobile robot navigation in partially observable environment is studied. Perseus algorithm is chosen because this method has been able to solve approximately large POMDP rapidly. The first objective is achieved by studying the Perseus algorithm. The algorithm selects several points over the belief-state space and compute several vectors that represents the value functions. It limits the value function size and optimizing the value function. Several important variables affecting the performance of the algorithm are the belief state space, number of point-based backups and the number of vectors in the belief space hyperplane.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

The second objective is achieved by obtaining results from the POMDP problems using Perseus algorithm and compared with SARSOP algorithm. The outcomes studied are the value, reward and number of vectors. High number of actions and low number of states help value converges to the optimum more quickly, and help collect better reward. Perseus algorithm can be scaled up to perform in larger problems better than SARSOP algorithm.

The recommendation for this project is to scale up Perseus algorithm to path planning in continuous state space so that the algorithm can simulate in an actual environment. Additionally, an algorithm needs to be shaped specially for robot navigation. The algorithm must be modified to suit the requirements demanded by specific application. In this project, the application is to transport object from a source to a destination in an enclosed environment. Thus, the modified algorithm only will have the greatest performance for this specific purpose only.

REFERENCES

[1] S. Candido, J. Davidson, and S. Hutchinson, "Exploiting domain knowledge in planning for uncertain robot systems modeled as POMDPs," Proc. - IEEE Int. Conf. Robot. Autom., pp. 3596–3603, 2010.

[2] H. I. Ibekwe and A. K. Kamrani, "Navigation for autonomous robots in partially observable facilities," World Autom. Congr. (WAC), 2014, pp. 1–5, 2014.

[3] S. Zhang, M. Sridharan, and C. Washington, "Active visual planning for mobile robot teams using hierarchical pomdps," IEEE Trans. Robot., vol. 29, no. 4, pp. 975–985, 2013.

[4] F. F. Carvalho, R. C. Cavalcante, M. A. M. Vieira, L. Chaimowicz, M. F. M. Campos, and M. Gerais, "A multi-robot exploration approach based on distributed graph coloring," 2013.

[5] S. Seuken and S. Zilberstein, "Improved Memory-Bounded Dynamic Programming for Decentralized POMDPs," Proc. Twenty-Third Conf. Uncertain. Artif. Intell., pp. 344–351, 2007.

[6] M. Cap, P. Novak, A. Kleiner, and M. Selecky, "Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots," IEEE Trans. Autom. Sci. Eng., vol. 12, no. 3, pp. 835–849, 2015.

[7] Q. Li, Z. Sun, S. Chen, and Y. Liu, "A method of camera selection based on partially observable Markov decision process model in camera networks," Am. Control Conf., pp. 3833–3839, 2013.

[8] B. Wang and S. Qin, "Multi-robot environment exploration based on label maps building via recognition of frontiers," Process. 2014 Int. Conf. Multisens. Fusion Inf. Integr. Intell. Syst. MFI 2014, no. 60875072, 2014.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

[9] W. Adiprawita, A. S. Ahmad, J. Sembiring, and B. R. Trilaksono, "Reinforcement learning with heuristic to solve POMDP problem in mobile robot path planning," Proc. 2011 Int. Conf. Electr. Eng. Informatics, ICEEI 2011, no. July, 2011.

[10] G. Theocharous, K. Murphy, and L. Kaelbling, "Representing hierarchical POMDPs as DBNs for multi-scale robot localization," Int. Conf. Robot. Autom., no. April, pp. 1045–1051, 2004.

[11] D. Grady, M. Moll, and L. E. Kavraki, "Automated model approximation for robotic navigation with POMDPs," Proc. - IEEE Int. Conf. Robot. Autom., pp. 78–84, 2013.

[12] S. Zhang, "Active Visual Sensing and Collaboration on Mobile Robots using Hierarchical POMDPs," Aamas, pp. 181–188, 2012.

[13] Y.-W. Chen and W.-Y. Chiu, "Optimal Robot Path Planning System by Using a Neural Network-Based Approach," Proc. 2015 Int. Autom. Control Conf., pp. 85–90, 2015.

[14] S. Li, X. Xu, and L. Zuo, "Dynamic Path Planning of a Mobile Robot with Improved Q-Learning algorithm," no. August, pp. 409–414, 2015.

[15] S. Zhang, M. Sridharan, and J. L. Wyatt, "Mixed Logical Inference and Probabilistic Planning for Robots in Unreliable Worlds," IEEE Trans. Robot., vol. 31, no. 3, pp. 699–713, 2015.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

[16] Q. Li, Z. Sun, S. Chen, and Y. Liu, "A method of camera selection based on partially observable Markov decision process model in camera networks," Am. Control Conf., pp. 3833–3839, 2013.

[17] X. R. Cao, D. X. Wang, and L. Qiu, "Partial-information state-based optimization of partially observable Markov decision processes and the separation principle," IEEE Trans. Automat. Contr., vol. 59, no. 4, pp. 921–936, 2014.

[18] H. Bai, D. Hsu, and W. S. Lee, "Integrated perception and planning in the continuous space: A POMDP approach," Int. J. Rob. Res., vol. 33, no. 9, pp. 1288–1302, 2014.

[19] H. Kurniawati, D. Hsu, and W. S. Lee, “SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces,” Proc. Robot. Sci. Syst. IV, p. w/o page numbers, 2008.

[20] L. MacDermed and C. L. Isbell, “Point Based Value Iteration with Optimal Belief Compression for Dec-POMDPs,” Nips, pp. 1–9, 2013.

[21] L. Tomás, Autonomous robot vehicles, Springer Science & Business Media, 2012.

[22] M. L. Cummings, J. P. How, A. Whitten, and O. Toupet, “The impact of human-automation collaboration in decentralized multiple unmanned vehicle control,” in Proceedings of the IEEE, 2012, vol. 100, no. 3, pp. 660–671.

[23] A. E. Nicholson and M. J. Flores, “Combining state and transition models with dynamic Bayesian networks,” Ecol. Modell., vol. 222, no. 3, pp. 555–566, 2011.

[24] I. A. Sucan and L. E. Kavraki, “Accounting for uncertainty in simultaneous task and motion planning using task motion multigraphs,” in Proceedings - IEEE International Conference on Robotics and Automation, 2012, pp. 4822–4828.

[25] I. A. Şucan, M. Moll, and L. Kavraki, “The open motion planning library,” IEEE Robot. Autom. Mag., vol. 19, no. 4, pp. 72–82, 2012.

[26] J. Pineau, G. Gordon, and S. Thrun, “Point-based value iteration: An anytime algorithm for POMDPs,” IJCAI Int. Jt. Conf. Artif. Intell., pp. 1025–1030, 2003.

[27] Sigaud, Olivier, and Olivier Buffet, eds. Markov decision processes in artificial intelligence. John Wiley & Sons, 2013.

[28] Russell, Stuart. "Artificial Intelligence: A Modern Approach Author: Stuart Russell." Peter Norvig, Publisher: Prentice Hall Pa 2009.

[29] A. R. Cassandra, L. P. Kaelbling, and M. L. L. B.-C. A. Littman, “Acting optimaly in partially observable stochastic domains,” *Aaai*, no. April, pp. 1023–1023, 1994.

[30] M. Littman, A. Cassandra, and L. Kaelbling, “Learning policies for partially observable environments: Scaling up,” *Icml*, no. February 1970, pp. 1–59, 1995.

[31] J. Pineau, G. Gordon, and S. Thrun, “Anytime Point-Based Approximations for Large POMDPs.,” *Jounal Artif. Intell. Res.*, vol. 27, pp. 335–380, 2006.

[32] M. T. J. Spaan and N. Vlassis, “Perseus: Randomized point-based value iteration for POMDPs,” *J. Artif. Intell. Res.*, vol. 24, pp. 195–220, 2005.

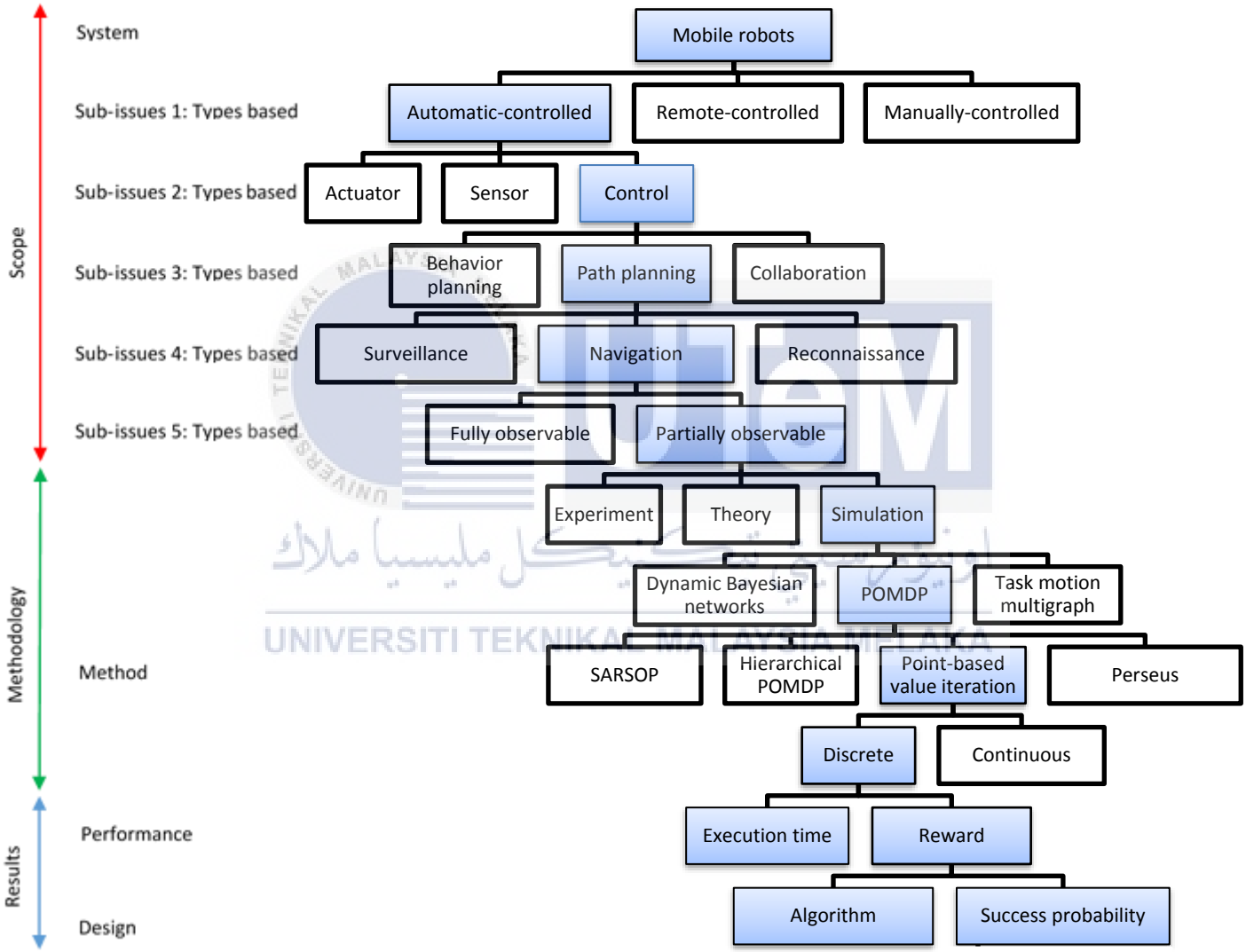
[33] G. Shani, J. Pineau, and R. Kaplow, “A survey of point-based POMDP solvers,” *Auton. Agent. Multi. Agent. Syst.*, vol. 27, no. 1, pp. 1–51, 2013.

[34] T. R. Halbert, “An Improved Algorithm for Sequential Information-Gathering Decisions in Design under Uncertainty,” 2015.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

APPENDIX

K-chart



UNIVERSITI TEKNIKAL MALAYSIA MELAKA