On Computing Optimal Thresholds for Sequential Hypothesis Testing Problem

Can Cui

MEng-Thesis

Department of Electrical and Computer Engineering

McGill University

Montreal,Quebec

2015-06-21

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Engineering

DEDICATION

To everyone who crossed my life and accompanied me during this journey.

ACKNOWLEDGEMENTS

First of all, I would like to express gratitude to my supervisor, Professor Aditya Mahajan. I am very grateful to him for admitting me into this Master program and guided me through my Master's research. He is not only a wonderful professor but also a good friend. His guidance has improved my research, my writing and my way of doing things. He is very knowledgeable but humble enough to discuss every detail of coding with me. He is strict on research but also friendly to discuss my holiday plans with me. Every week, during our meeting, I clear my confusion and learn new things. I learned a lot from him.

I also gratefully thank Professor Ioannis Psaromiligkos, Benoit Boulet, Hannah Michalska, Doina Precup, Richard Rose, Joelie Pineau and Michael Rabbat for the courses I've taken with them. These courses enlightened me on my research and also broadened my interest in other fields.

I like to thank my parents for supporting me to continue graduate school in McGill, even though we didn't agree on this decision at first place, they decided to believe in me and do whatever it takes to support me. And they always stand beside me whenever I come across obstacles. It's their belief that has supported me so far.

I also like to thank my boyfriend Yun Chen, my best friend Yu Zhang and Mengyao Zhang. The courage and accompany they provided, even in a long distance, helped me through many tough moments.

My lab mates in 502 are a group of people I am most close with during the last two years. I appreciate Jalal Arabneydi for his suggestion when I was doing a

very difficult part of my research. His simple and clear explanation was really great. I want to thank Mehnaz Mannan for giving me advice on graduate school and life in Montreal before I actually got here. I thank Jhelum Chakravorty and Prokopis Prokopiou for many discussions we had about the courses and research and many interesting past experience sharing talks. I also appreciate Shuang Gao for his help in both life and study, having a lab mates coming from China and talking Chinese is really good and we even shared the same desk for the first year of my graduate study. Although some of them already left lab 502, I will never forget this team we've made and many experiences we've shared together.

I also like to thank Ali Pakniyat, Sayani Seal, Hamed Layeghi, Mohammad Afshari and Dena Firoozi for all those ISS (Informal Systems Seminar) we've attended together. We also had a lot of fun in many other activities.

Lastly, a thank you to my friends Di Wu, Jing Zhu, Linwan Liu, Xiangyu Ren and Aida Nowzary for study mates for COMP 652, especially to Di Wu, we did a very interesting final project together. Another thank you to Chen Jiang from ECSE 500, for his suggestion in job hunting, to Wei Li from ECSE 501 who offered help in the course and life. Last but not least, to Jingsi Lv, my roommates Nanying Tao and Ke Sun, they are very good friends to me.

ABSTRACT

Sequential hypothesis testing problem, after firstly being successfully used in the second world war, has developed its own research field and been applied to many other fields. How to numerically solve this problem and develop an automatic decision strategy has become an academic topic since 1950's. For centralized sequential hypothesis testing, the threshold-based structure is well developed, and for decentralized sequential hypothesis testing, there has been a lot of recent progress in understanding the structure of optimal control strategies, but very little is known about computational methods in both cases. In this work, we looked at different methods to numerically find the optimal stopping rule for centralized and decentralized sequential hypothesis testing problem. We first implemented the classic Sondik's algorithm. Then, we seek approximation methods to simplify the process and improve the efficiency. To do this, we first introduced zeroth-order and first-order discretization methods to discretize the continuous state. After discretizing the state, we propose two main approaches to find the optimal threshold: one is value iteration, the other is policy evaluation combined with non-convex optimization function. For policy evaluation, we tried three approximation methods: Monte Carlo sampling, asymptotic expression and absorption probability of Markov chain. The performance of these methods are compared. Finally, for decentralized hypothesis testing problem, we choose one approximation method and compare the person-by-person optimal strategy with the optimal strategy based on global search.

ABRÉGÉ

Le problème de test d'hypothèse séquentielle, après avoir tout d'abord été utilisé avec succès dans la seconde guerre mondiale, a développé dans son propre domaine de recherche et a été appliquée à de nombreux autres domaines. Comment résoudre numériquement un problème de ce type et élaborer une stratégie automatique de décision est devenu un sujet académique depuis 1950. Pour les test d'hypothèse séquentielle centralisées, la structure à seuil est bien développé, et pour les tests d'hypothèse séquentielle décentralisées, il y a eu beaucoup de progrès récents dans la compréhension de la structure des stratégies de contrôle optimales, mais on sait très peu sur les méthodes de calcul dans les deux cas . Dans ce travail, nous avons examiné de différentes méthodes pour trouver numériquement la règle d'arrêt optimale pour le problème de test d'hypothèse séquentielle centralisée et décentralisée. Nous avons d'abord mis en œuvre l'algorithme classique de Sendik. Ensuite, nous cherchons des méthodes d'approximation pour simplifier le processus et améliorer l'efficacité. Pour ce faire, nous avons introduit des méthodes de discrétisation d'ordre zéro et de premier ordre pour discrétiser l'état continu. Après avoir discrétiser l'état, nous proposons deux approches principales pour trouver le seuil optimal: l'itération de valeur, et l'évaluation stratégique, combiné à une fonction d'optimisation non convexe. Pour l'évaluation stratégique, nous avons essayé trois méthodes d'approximation: l' échantillonnage Monte Carlo, l'expression asymptotique et la probabilité d'absorption de chaîne Markov. La performance de ces méthodes sont comparées. Enfin, pour le problème de test d'hypothèse décentralisée, nous choisissons une méthode d'approximation et nous comparons la stratégie optimale personne par personne avec la stratégie optimale en fonction de recherche globale.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

In our daily life when humans are decision makers, decisions can be made based on many factors: knowledge of the immediate circumstances; specialized knowledge; previous experiences about the effects of various actions. For many situations, this works well. However, when systems become more complex, the various component and their interactions become more difficult to reason about.

One aspect where automated decision making is needed is to ease the burden of human beings by designing models and developing tools to deal with complicated, uncertain processes or making better decisions than human beings. As an example consider inventory control in large company like Amazon, where interactions between customer demand, supply availability and resource limitations are complex. Even the most dedicated corporate manager will be overwhelmed by these. Operation research has provided successful tools to businesses for making these kinds of decisions.

Another aspect of automated decision making is motivated by problems where decisions are made, but it is not feasible or desirable to have a human available. One particular discipline of artificial intelligence has focused on automated decision making for problems like this, For example, a rover on the moon has to decide for itself on navigation problems based on its own observation and analysis. Communicating with earth-based controllers for instructions is improper for either financial or scientific reasons.

1.1 Sequential Decision Making

Sequential decision making problems are problems in which the decision maker needs to make a series of decisions in order to achieve the objective. After any decision he makes, an event will occur and change the current situation. For instance, a company trying to decide whether or not to market a new product might first decide to test the acceptance of the product using a consumer panel. Based on the results of the consumer panel, the company will then decide whether or not to proceed with further test marketing; after analyzing the results of the test marketing, the company will decide whether or not to produce the new product. The essence of sequential decision making is that decisions that are made now can have both immediate and long-term effects; the best choice to make now depends critically on future situations and how they will be faced.

To model the process of sequential decision making, several terms are introduced:

The Agent: In the context of this work, an agent is simply the system responsible for interacting with the world and making decisions. i.e. the decision maker. The agent "lives" in an environment. The *state* of the environment is a description of everything that might change from moment to moment.

The Environment: The environment is anything external to the agent. For the purpose of this thesis, I assume that the environment changes from state to state in response to the actions of the agent according to a fixed set of rules. The transition might be stochastic, that is, it is not necessary that the same transition occurs every time the agent takes a particular action in a particular state. However,

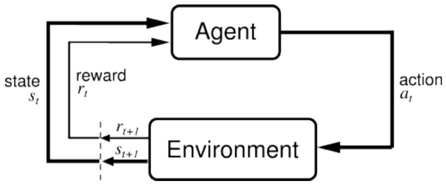

Figure 1–1: Components of a MDP system [1].

the probabilities that govern these stochastic transitions must remain constant over time.

Reward: To describe a sequential decision-making problem, it is not enough to specify the agent and the environment alone. The agent's actions need to serve some purpose. For each action the agent takes, there exists an associated reward (or cost), the final goal is to maximize the reward or minimize the cost.

Policies: A policy is an agent's prescription for behavior: a function that specifies the action that the agent will choose when in a given state.

The relationship of the above definition is shown in Figure1–1.

In a real world problem, the environment of a decision making process may not be deterministic. There are two major forms of uncertainty: the result of our decisions may not always have the same effects; and our perceptions of the systems being controlled are not always very accurate. We will introduce two useful frameworks in sequential decision making: Markov decision processes (MDP) and partially observable Markov decision processes (POMDP).

1.1.1 Markov decision process

Markov decision process, named after Andrey Markov, is a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker. In a MDP framework, agents move stochastically between states by executing actions and then receiving a reward. MDP is an extension of Markov chains, more precisely, it is a discrete time stochastic control process.

A Markov decision process can be described with a tuple $\langle \mathcal{X}, \mathcal{U}, P, C \rangle$, where,

- $\mathcal{X}$ is a finite set of states of the environment;
- $\mathcal{U}$ is a finite set of actions;
- The state transition function is

$$P(x, u, x') := \mathbb{P}(x_t = x' \mid x_{t-1} = x, u_{t-1} = u),$$

  giving for each environment state and agent action, a probability distribution over states. This is the probability of ending in state x', giving that the agent starts in state x and takes action u;
- $C : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the cost function, giving the expected immediate cost spent (or reward gained) by the agent for taking each action in each state, $C(x, u)$ is the expected cost for taking action u in state x.

At each time step, the process is in some state x, and the decision maker may choose any action u that is available in state x, this action will take the process to a new state x', and giving the decision maker a corresponding cost $C(x, u)$. The probability that the process goes into state x' from state x is determined by the state transition probability $P(x, u, x')$. The next state x' depends on the current state x and the action u. But given x and u, it is conditionally independent of all previous states and actions. The fact that the next state probabilities only depend upon the current state and action is the Markov property of the process.

The agent should act in such a way as to minimize some measure of the long-run costs spent. The most straightforward approach to achieve is to sum costs over the infinite lifetime of the agent, but discount them geometrically using a discount factor $0 < \gamma < 1$; the goal is to optimize

$$\sum_{t=0}^{\infty} \gamma^t C(x_t, u_t).$$

In here, costs received earlier have more value to the agent. Even though the infinite lifetime is considered, the discount factor γ ensures that the sum is finite.

A policy is a description of the behavior of an agent. There are two kinds of policies: stationary and non-stationary. Consider a stationary policy $g : \mathcal{X} \to \mathcal{U}$. The choice of action depend only on the state and is independent of the time step. The value function of a policy g is a map $V^g : \mathcal{X} \to C$, and $V^g(x)$ is the expected discounted sum of costs for starting in x and executing stationary policy g indefinitely. It is shown that there exists a stationary policy g^* [3], that is optimal for every starting state. The value function for this policy, V^{g^*} (also written as V^*) is defined by

$$V^*(x) = \min_{u \in \mathcal{U}} [C(x, u) + \gamma \sum_{x' \in \mathcal{X}} P(x, u, x') V^*(x')].$$

$V^*(x)$ is known as the Bellman equation and has a unique solution. The solution $g^*(x)$ is a greedy policy with respect to $V^*(x)$

$$g^*(x) = \arg\min_{u \in \mathcal{U}} [C(x, u) + \gamma \sum_{x' \in \mathcal{X}} P(x, u, x') V^*(x')].$$

For a completely observable Markov decision process, where the decision maker has access to the current state of the system at each decision point, we can use value iteration to solve the problem. For details of extensive and mathematically rigorous studies, see [4], [5]. Good starting references for MDPs are Puterman's textbook [6] and Bersekas' textbook [7].

1.1.2 Partially observable markov decision process

Although we can effectively solve MDP problems, the solutions have limited use and generally cannot be applied when the system does not permit access to the state directly. Now, we introduce a more general MDP model: partially observable Markov decision process. In a POMDP, we still assume that the system behaves in the same fashion as the MDP discussed previously: there are states, actions, costs and state transition based upon the current state-action pair. However, a set of observations is added to the model so that after each state transition of the system, one of these observations is produced by the system and is accessible to the decision maker. The observation produced is corrected with the current state, but does not generally allow us to completely determine the current state.

Formally, a discrete POMDP is specified as a tuple $\langle \mathcal{X}, \mathcal{U}, \mathcal{Y}, P, F, C \rangle$, where $\mathcal{X}, \mathcal{U}, P, C$ are defined as in MDP. In addition, it includes a finite set of observations $\mathcal{Y}$, and an observation function $F : \mathcal{X} \times \mathcal{U} \to \Pi(\mathcal{Y})$, where

$$F(x', u, y) := \mathbb{P}(y_t = y \mid x_t = x', u_t = u).$$

Since the system is partially observable, we maintain a distribution, known as the *belief state* to summarize its previous experiences. These distributions encode the agent's subjective probability about the current state of the world, and provide a basis for acting with incomplete information. It is proved that the belief state is a sufficient statistic for the past history, which means that given the agent's current belief state, no additional data about its past actions or observations would give any more information about the current state of the environment.

Let $\pi(x)$ denote the probability assigned to state x by belief state π:

$$\pi_t(x) = \mathbb{P}(x_t = x \mid Y_{1:t}, U_{1:t-1}).$$

After the agent takes an action u and receives an observation y, the update of belief state to $\pi_{t+1}(x')$ is given by

$$\begin{aligned}
\pi_{t+1}(x') &= \mathbb{P}(x_{t+1} = x' \mid Y_{1:t+1} = y_{1:t+1}, U_{1:t} = u_{1:t}) \\
&= \mathbb{P}(x' \mid y, u, \pi_t) \\
&= \frac{\mathbb{P}(y \mid x', u, \pi_t) \cdot \mathbb{P}(x' \mid u, \pi_t)}{\mathbb{P}(y \mid u, \pi_t)} \\
&= \frac{\mathbb{P}(y \mid x', u) \cdot \sum_{x \in \mathcal{X}} [\mathbb{P}(x' \mid u, \pi_t, x) \cdot \mathbb{P}(x \mid u, \pi_t)]}{\mathbb{P}(y \mid u, \pi_t)} \\
&= \frac{F(x', u, y) \cdot \sum_{x \in \mathcal{X}} [P(x, u, x') \cdot \pi_t(x)]}{\mathbb{P}(y \mid u, \pi_t)} \\
&=: \phi(\pi_t, u, y)(x).
\end{aligned}$$

The denominator $\mathbb{P}(y \mid u, \pi_t)$ can be treated as a normalizing factor, independent of x, which makes π_t sum to 1.

The optimal policy for a POMDP is one that chooses an action that minimizes the expected future discounted cumulative cost:

$$g^*(\pi) = \operatorname*{arg\,min}_{u \in \mathcal{U}} \mathbb{E}[\sum_{t=0}^{T} \gamma^t C_t(x_t, u_t) \mid \pi].$$

The value function of a policy is the expected discounted sum of rewards for executing g starting from belief π_0,

$$V^*(\pi) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t C_t \mid \pi_0],$$

where C_t is the cost spent at time step t, γ is the discount factor.

Computing the optimal policy is difficult. There are two main obstacles. The first one is the *curse of dimension*. In a problem with n states, the corresponding belief space then has n dimensions. The second obstacle is the *curse of history*. Many decision making tasks require the agent to take many actions before it can reaches its goal, resulting in a long time horizon for planning. The complexity of planning grows exponentially with the time horizon.

In the infinite-horizon discounted case, for any initial belief π, we want to execute the policy g that minimizes V^*, defined by:

$$V^*(\pi) = \max_u [\sum_{x \in \mathcal{X}} C(x, u) + \gamma \sum_{y \in \mathcal{Y}} \mathbb{P}(y \mid \pi, u) V^*(\phi(\pi, u, y))],$$

where $\phi(\pi, u, y)$ is the new belief state found by after taking action u in belief state π and observing y. And the optimal policy g is a the one that minimize the value function.

1.2 Decentralized Sequential Decision Making

Sequential decision making theory provides analytic and computational techniques for centralized decision making in stochastic systems with noisy observations. However, this assumption of a centralized decision maker is not true in many modern control applications such as networked control systems, communication and queuing networks, sensor networks, and smart grids. In these applications, decisions are made by multiple decision makers who have access to different information, this kind of problem is called *decentralized sequential decision making*.

The technique from centralized sequential decision making cannot be directly applied to decentralized problems. Two general solution approaches that indirectly use techniques from centralized sequential decision making have been used to solve decentralized sequential decision making problems: 1) person-by-person approach which views the problem from each individual decision maker; and 2) the global search approach which views the problem as a team collaboration.

The person-by-person approach investigates the decentralized problem from the viewpoint of one decision maker (DM), for DM^i: 1) fix the initial strategy of all other DMs except DM^i; and 2) use centralized sequential decision making to design best-response strategy of DM^i. By iteratively proceed these steps until no DM can improve performance by changing its strategy, we can identify the person-by-person strategy for all DMs. In this thesis, we call the procedure that finds a person-by-person optimal strategy as *orthogonal search*.

The global search approach investigates the decentralized sequential decision making problem from the view of a team, where all decision makers collaborate to achieve a common goal. Effectively, we are solving a centralized planning problem: 1) model this centralized planning problem as a multi-stage, open-loop stochastic control problem, at each time, we design the control law for all DMs; and 2) use centralized sequential decision making to obtain a dynamic programming decomposition. Each step of the dynamic program is a functional optimization problem, rather than a parameter optimization problem as in centralized dynamic programming. In this thesis, we call the procedure to find a global optimal strategy as *direct search*.

1.3 Sequential Hypothesis Testing

One classic and widely studied problem of sequential decision making is sequential hypothesis testing. Sequential hypothesis testing was first formulated by Wald [8] for efficient testing of anti-aircraft guns in World War II. Since then, the theory of sequential hypothesis testing has been applied to various applications. A sequential test of a statistical hypothesis means any statistical test procedure which gives a specific rule, at any stage of the experiment, the decision maker needs to make one of the following three decisions: 1) to accept the hypothesis being tested (usually called the *null hypothesis*), 2) to reject the null hypothesis, 3) to continue the experiment by making an additional observation. Thus, the test procedure is carried out sequentially and is terminated when either the first or the second decision is made. In a sequential test, the number of observations is not predetermined but is a random variable.

In a sequential test, there are two kinds of errors. We may reject the null hypothesis when it is true (also called missed detection), or we may accept the null hypothesis when some alternative hypothesis is true (also called false alarm). There are cost incurred when we make a wrong decision and cost incurred when we make an additional observation. The goal is to design an optimal stopping rule that minimize the total cost.

In a centralized sequential hypothesis testing, there is only one decision maker while in a decentralized sequential hypothesis testing there are two or more decision makers.

1.3.1 Application examples

There are many real world problems in which decisions have uncertain outcomes and uncertain perception of the current state of the system. We list some of the problems below.

Machine Maintenance: A machine used in certain industry have a myriad of internal components, each of which must operate on the product before it is finished. All of the components affect the tolerances and general quality of the parts being produced. Obviously, the state of internal components is not directly observable, we can have a general predictability based on the operating hours of the component, but how much it is worn or when it may stop performing is a very non-deterministic behavior. We want to replace the worn components before they produce defective parts, but to access the real state of one component requires the cost of disassembling the machine and a loss or profit while the whole machine is not working.

The task is to design a maintenance schedule: when to manufacture items, when to dismantle the machine and inspect the components and (or) replace it when fails. The quality of the items being produced provide an indirect, probabilistic observation about the internal components. There is cost associated with inspecting or replacing components. Also there is a loss in revenue if the replacement is being delayed and the machine starts producing defective items. We need to choose the best schedule that minimize the total cost. This Machine maintenance problem is studied by Richard D. Smallwood and Edward J. Sondik [3]. This inspection, maintenance and repair problem has a broader application than just manufacturing machines.

Networking Troubleshooting: There is a computer network consists of several switching computers and cables connecting computers. At any moment of time, the switching computer is up or down and lightly or heavily loaded, and the cables can be transmitting or not transmitting information. An electrical engineer need to design an equipment that plugs into the network to monitor and correct faults in a specific section of the network.

It is not possible to know certainly the state of the network at any time, but the equipment can have access to various signals and alarms emitted by the computers. It can also send query signals to get feedback on the status of the switching computer. For example, if a computer fails to respond to a query, it is possible that the computer is overloaded, or it has gone down, or one of the incoming cables stopped transmitting data. There are probabilistic distributions on the reasons given a trouble situation.

In spite of the unsure and inaccurate information, the monitoring equipment must keep the network running because there will be revenue loss if the network is down. It can send query and reset signals to the computer or call for a technician to examine or repair the computers or cables. And there are different costs associated with these actions. To make the optimal decision, the system need to be modeled and algorithms need to be designed to find the strategy. Similar supply restoration in power systems [9] is studied.

Other Applications: Apart from the examples stated above, there are many other applications to imply sequential decision making in real world situations. Including medical diagnosis [10]; cost control in accounting [11]; teaching strategy [12]; moving target search [13];and elevator control [14].

1.4 Contributions

The primary contribution of this thesis is in providing numerical methods to solve sequential hypothesis testing problem. To achieve this, we implemented some existing algorithms like Sondik's enumeration method and witness algorithm. Then, we seek approximation method by discretizing the continuous belief space into equally distributed discrete states. Based on the discrete model, we also implemented existing method like value iteration. Then, we propose a new technique using the absorption probability of Markov chain. In the latter part, we also extend the centralized sequential hypothesis testing problem to decentralized sequential hypothesis testing problem. For decentralized sequential hypothesis testing problem, we simulated the orthogonal search method proposed in [15] and compare it to our direct search method.

Overall, this work not only provides numerical evaluations about existing methods but also suggests new insights to solve sequential hypothesis testing problem. The numerical evaluations and discussion of different approaches will be useful to practitioners developing sequential hypothesis testing systems and the insights may lead to new theoretical developments as well.

The content of Chapter 3 has been submitted to the 54th IEEE Conference on Decision and Control: C.Cui and A. Mahajan, “On computing optimal thresholds in decentralized sequential hypothesis testing”, submitted to 54th IEEE Conference on Decision and Control, Osaka, Japan, December 15-18, 2015.

1.5 General Guidelines

We now present the overall organization structure of the thesis. In Chapter 2, we investigate the centralized sequential hypothesis testing problem. This chapter has four parts: in the first part, we formally present the centralized model and its dynamic programming decomposition; in the second part, we try to solve a continuous state POMDP problem using two methods: Sondik's enumeration method (also written as α-vector method) and mathematically asymptotic expressions; in the third part, we use grid-based discretization to approximate the continuous state space and then introduce three approximation methods: value iteration, direct search with absorption probability, direct search with Monte Carlo simulation; in the last part, we present the numerical results in a cast study and conclude this chapter. In Chapter 3, we investigate the decentralized sequential hypothesis testing problem. This chapter also has four parts: in the first part, we present the decentralized model and its dynamic programming decomposition; in the second part, we use orthogonal search to identify a person-by-person optimal strategy; in the third part, we use direct search to identify another strategy; in the last part, we present the numerical results in a cast study and conclude this chapter. Lastly, we conclude in Chapter 4.

CHAPTER 2
Centralized Sequential Hypothesis Testing

In this chapter, we will focus on a centralized sequential hypothesis testing problem, we will formally present the model and problem first, then provide numerical methods to solve the problem, including the exact algorithm and approximate algorithms. At last, we use a case study to conclude this chapter.

2.1 Model

The problem is stated as below: A decision maker (DM) makes series of i.i.d. observations which may be distributed according to PMF f_0 or f_1. Let the random variable H denote the value of the true hypothesis, let Y_t denote the observation at time t. The DM need to differentiate between two hypothesis:

$$h_0 : Y_t \sim f_0 \quad \text{and} \quad h_1 : Y_t \sim f_1.$$

The a prior probability $\mathbb{P}(H = h_0) = p$. At each time $t < T$, the DM has three choices: 1) to stop observing and declare h_0 is true; 2) to stop observing and declare h_1 is true; 3) to continue the experiment by making another observation. At time T, option 3 is unavailable. Thus, such procedure is carried out sequentially.

When making decisions, there are two kinds of errors: we may accept h_0 when h_1 is true or we may accept h_1 when h_0 is true. Let u denote the final decision, the cost of making wrong decision is denoted as $\ell(u, H)$. There is also cost of taking per step observation, denoted by c. The goal of the DM is to find a optimal strategy to minimize the expected cost.

In order to solve the problem stated above, we can build the model as:

$$\begin{aligned}
\text{State} \quad &: \quad X_t = H \in \{h_0, h_1\};\\
\text{Observation} \quad &: \quad \text{Under } H = h_0, Y_t \sim f_0,\\
&\quad\quad \text{Under } H = h_1, Y_t \sim f_1;\\
\text{Action} \quad &: \quad \text{For } t < T : U \in \{h_0, h_1, \mathsf{C}\},\\
&\quad\quad \text{For } t = T : U \in \{h_0, h_1\};\\
\text{Cost} \quad &: \quad u_t \in \{\mathsf{C}\}, C_t(u_t, X_t) = c,\\
&\quad\quad u_t \in \{h_0, h_1\}, C_t(u_t, X_t) = \ell(u_t, H);
\end{aligned}$$

It can be seen that the sequential hypothesis testing problem is a POMDP problem since we do not have access to the state. Define a belief state:

$$\pi_t(i) = \mathbb{P}(H = h_i \mid Y_{1:t}).$$

In this case, $p_t = \mathbb{P}(H = h_0 \mid Y_{1:t}) = \pi_t(0)$. The update of p_t follows Bayes rule:

$$p_{t+1} = \varphi(p_t, y_t) = \frac{p_t f_0(y_t)}{p_t f_0(y_t) + (1 - p_t) f_1(y_t)}.$$

The dynamic programming can be developed as:

$$V_T(p) = \min\{p \cdot \ell(h_0, h_0) + (1 - p) \cdot \ell(h_1, h_0),\ p \cdot \ell(h_0, h_1) + (1 - p) \cdot \ell(h_1, h_1)\};$$

and for t = T-1, . . . , 1,

$$V_t(p) = \min\{p \cdot \ell(h_0, h_0) + (1-p) \cdot \ell(h_1, h_0),\ p \cdot \ell(h_0, h_1) + (1-p) \cdot \ell(h_1, h_1),$$
$$c + \mathbb{E}[V_{t+1}(\varphi(p, Y_{t+1})) \mid p_t = p]\}.$$

2.2 Structural Properties

The results presented in this section closely follow the presentation in [16], which in turn was based on [17].

Definition 2.2.1 *Define* $W_T(p) = \infty$, $W_t(p) = c + \mathbb{E}[V_{t+1}(\varphi(p, Y_{t+1})) \mid p_t = p]$.

Definition 2.2.2 *Define* $L_i(p) = p\ell(h_i, h_0) + (1-p)\ell(h_i, h_1), \quad i \in \{0, 1\}$.

Theorem 2.2.1 *:* $V_t(p)$ *and* $W_t(p)$ *are* $\forall$ *t, concave in p,* $\forall$ *p, increasing in t.*

Proof of concavity in p:

It is proved by backward induction.

Basis: $V_T(p)$ is minimum of two linear functions, therefore, $V_T(p)$ is concave. $W_T(p)$ is a constant, therefore, $W_T(p)$ is concave.

Hypothesis: Suppose $V_{t+1}(p)$ and $W_{t+1}(p)$ are concave in p.

Induction: From the properties of convex functions, sum of concave functions is concave.

$$W_t(p) = c + \int_y [pf_0(y) + (1-p)f_1(y)]V_{t+1}(\frac{pf_0(y)}{pf_0(y) + (1-p)f_1(y)})dy.$$

Therefore, $W_t(p)$ is concave in p. Then, $V_t(p)$, which is the minimum of three functions, two linear in p and one concave in p, is also concave in p.

Proof of increasing in t:

It is proved by backward induction.

Basis: By construction, $W_{T-1}(p) \leq W_T(p)$. Moreover,

$$V_{T-1}(p) = \min\{W_{T-1}(p), L_0(p), L_1(p)\} \leq \min\{L_0(p), L_1(p)\} = V_T(p).$$

Hypothesis: Suppose $V_{t+1}(p) \leq V_{t+2}(p)$ and $W_{t+1}(p) \leq W_{t+2}(p)$.

Induction:

$$W_t(p) = c + \mathbb{E}[V_{t+1}(\varphi(p, Y_t)) \mid p_t = p] \leq c + \mathbb{E}[V_{t+2}(\varphi(p, Y_t)) \mid p_{t+1} = p] = W_{t+1}(p),$$

and

$$V_t(p) = \min\{W_t(p), L_0(p), L_1(p)\} \leq \min\{W_{t+1}(p), L_0(p), L_1(p)\} = V_{t+1}(p).$$

Definition 2.2.3 *Define the stopping set* $S_t(h) = \{p \in [0,1] : g_t(p) = h\}, h \in \{h_0, h_1\}$

Theorem 2.2.2 *For all* t *and* $h \in \{h_0, h_1\}$*, the set* $S_t(h)$ *is convex.*

Proof: To show that $S_t(h_0)$ is convex, it suffices to show that:

For any $p^{(0)}$, $p^{(1)} \in S_t(h_0)$ and $\lambda \in [0, 1]$,

the belief state $p^{(\lambda)} = (1-\lambda)p^{(0)} + \lambda p^{(1)}$ is in $S_t(h_0)$.

Since $p^{(i)} \in S_t(h_0)$, i=0,1:

$$L_0(p^{(i)}) \leq \min\{L_1(p^{(1)}), W_t(p^{(i)})\}.$$

Since $L_i(p)$ is linear in p, i=0,1:

$$(1-\lambda)L_i(p^{(0)}) + \lambda L_i(p^{(1)}) \leq L_i(p^{(i)}).$$

Since $W_t(p)$ is concave in p:

$$(1-\lambda)W_t(p^{(0)}) + \lambda W_t(p^{(1)}) \leq W_t(p^{(\lambda)}).$$

Combining the above, we have:

$$L_0(p^{(\lambda)}) \leq \min\{L_1(p^{(\lambda)}), W_t(p^{(\lambda)})\}.$$

Hence, $p^{(\lambda)} \in S_t(h_0)$. Consequently, $S_t(h_0)$ is convex.

Assumption (A1):

$$\ell(h_0, h_0) \leq c \leq \ell(h_0, h_1) \quad \text{and} \quad \ell(h_1, h_1) \leq c \leq \ell(h_1, h_0).$$

We assume that when the decision maker makes correct decision, there is no punishment. And the cost of making wrong decisions should be bigger than the cost of making an additional observation, otherwise the decision makers might just choose to reach any decision at the first time step.

Theorem 2.2.3 *Under (A1):* $0 \in S_t(h_1)$ *and* $1 \in S_t(h_0)$

Proof: $L_0(0) = \ell(h_0, h_1)$,$L_1(0) = \ell(h_1, h_1)$, and $W_t(0) \geq c$. Therefore, we have:

$$L_1(0) \leq \min\{L_0(0), W_t(0)\} \Rightarrow 0 \in S_t(1).$$

From similar proof, we have $1 \in S_t(0)$.

Now, define the upper and lower bound:

$$\alpha_t = \max\{p \in [0,1] : g_t(p) = h_1\},$$

$$\beta_t = \min\{p \in [0,1] : g_t(p) = h_0\}.$$

Then, under (A1), the optimal control law has a threshold property as the following:

$$g_t(p) = \begin{cases} h_1, & \text{if} \quad p \leq \alpha_t, \\ \mathsf{C}, & \text{if} \quad \alpha_t < p < \beta_t, \\ h_0, & \text{if} \quad \beta_t \leq p. \end{cases}$$

For a infinite horizon sequential hypothesis testing problem, $T \to \infty$, which means that the continuation alternative is always available.

Theorem 2.2.4 *For the infinite horizon problem, an optimal stopping rule exists, is time-invariant, and is given by the solution to the following fixed point equation*

$$V(p) = \min\{L_0(p), L_1(p), W(p)\},$$

where $W(p) = c + \int_y [pf_0(y) + (1-p)f_1(y)]V(\varphi(p,y))dy$.

The proof follows from standard results on non-negative dynamic programming [18].

After introducing the model, in the rest of this chapter, we will talk about how to numerically solve a sequential hypothesis testing problem. The exact solution using α-vectors is provided first, then we try to reduce the complexity in two ways: 1) use asymptotic expression; 2) discretize the continuous state space.

2.3 Exact Solution: α-vector

After knowing the problem and the threshold property of the solution, the next step is to find how to solve the problem numerically. In this section, we will review some of the classic dynamic programming based algorithms. Firstly, Sondik's algorithm [19, 20] is introduced in detail, then we discuss other algorithms that try to

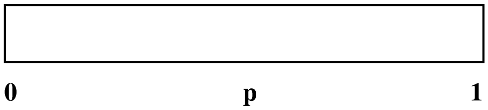

Figure 2–1: Belief space for two state POMDP.

improve the efficiency of Sondik's algorithm. The reader is referred to the survey by Monahan [21], White [22] and Lovejoy [23].

2.3.1 Sondik/Monahan's enumeration algorithm

It is proved in the previous section that a POMDP problem can be transformed to a MDP by defining the belief state as: $\pi_t(i) = \mathbb{P}(H = h_i \mid y_{1:t}, u_{1:t-1}), i \in [0, n]$. (Note that in is section, we generalize the problem to multi-dimensional case, we have more than two hypothesis). It has also been shown that this belief state satisfies the Markov property: $\mathbb{P}(\pi_{t+1} \mid \pi_{1:t}, u_{1:t}) = \mathbb{P}(\pi_{t+1} \mid \pi_t, u_t)$. Thus, dynamic programming can be employed in this problem,

$$V_T(\pi_T) = \min_{u_T}(\mathbb{E}(\ell(u_T, H_T))),$$
$$V_t(\pi_t) = \min_{u_t}\{\sum_{i=1}^{n} \ell(u_t, h_i)\pi_t(i) + \mathbb{E}(V_{t+1}(\pi_{t+1}) \mid \pi_t, u_t)\}.$$

For every belief state, there is one value function and correspondingly one optimal control strategy. Since the belief state is continuous between $[0, 1]$, there is uncountable infinite number of value functions. For a two state POMDP we can represent the belief state with a single number. Figure 2–1 shows how we represent the belief space. Since a belief state is a probability distribution, the sum of all probabilities must sum to 1. With a two state POMDP, if we are given the probability

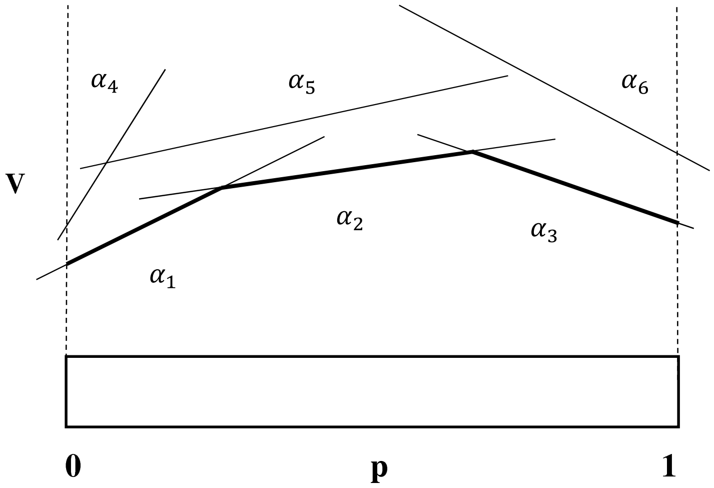

Figure 2–2: Illustration of a value function represented by α-vectors.

for being in one of the state as being p, then we know that the probability of being in the other state must be $1 - p$.

An important characteristic in here is that the finite horizon value function is piece-wise linear and concave (PWLC) for every horizon length because we are minimizing cost [19]. This indicates that for each iteration of value iteration, we only need to find a finite number of linear segments. From Figure 2–2, we can see that the value function is the lower surface of finite number of linear segments. These linear segments will completely specify the value function and we can represent each hyper-plane with a vector of numbers. These vectors are called *α-vectors*. Knowing α-vector will result in knowing the control action which has generated it. To represent the value function in the form of α-vector:

$$\begin{aligned} V_T(\pi_T) &= \min_{u_T}(\mathbb{E}(\ell(u_T, H))) \\ &= \min_{u_T}\{\sum_{i=0}^{n}\ell(u_1, h_i)\pi_T(i), \sum_{i=0}^{n}\ell(u_2, h_i)\pi_T(i), \ldots, \sum_{i=0}^{n}\ell(u_m, h_i)\pi_T(i)\} \\ &= \min_{u_T}\{\alpha(u_1)\pi_T, \alpha(u_2)\pi_T, \ldots, \alpha(u_m)\pi_T\}, \end{aligned}$$

where,

$$\pi_T = \begin{bmatrix} \pi_T(1) \\ \pi_T(2) \\ \vdots \\ \pi_T(n) \end{bmatrix} \Rightarrow V_T(\pi_T) = \min_{u}\{\alpha(u_1)\pi_T, \alpha(u_2)\pi_T, \ldots, \alpha(u_m)\pi_T\}. \tag{2.1}$$

It is obvious that the value function at the terminal step is piece-wise linear and concave. Assume that at step $t+1$, the set of all α-vectors, denoted as A_{t+1}, are known, then,

$$V_{t+1}(\pi_{t+1}) = \min_{u_{t+1}}\{\alpha_{t+1}(u_{t+1}) \cdot \pi_{t+1}\}.$$

Now, the value function at time t should be:

$$V_t(\pi_t) = \min_{u_t}\{\sum_{i=1}^{n}\ell(u_t, h_i)\pi_t(i) + \mathbb{E}(V_{t+1}(\pi_{t+1}) \mid \pi_t, u_t)\}, \tag{2.2}$$

$$\pi_{t+1}(h_{t+1}) = \frac{\mathbb{P}(y_{t+1} \mid h_{t+1}) \cdot \mathbb{P}(h_{t+1} \mid \pi_t, u_t)}{\mathbb{P}(y_{t+1} \mid \pi_t, u_t)}. \tag{2.3}$$

By substituting (2.3) into (2.2), we get the result:

$$\begin{aligned}
\mathbb{E}(V_{t+1}(\pi_{t+1}) \mid \pi_t, u_t) &= \sum_{y_{t+1}} \mathbb{P}(y_{t+1} \mid \pi_t, u_t) \cdot V_{t+1} \left(\begin{bmatrix} \dfrac{\mathbb{P}(y_{t+1} \mid h_{t+1} = h_1) \cdot \mathbb{P}(h_{t+1} = h_1 \mid \pi_t, u_t)}{\mathbb{P}(y_{t+1} \mid \pi_t, u_t)} \\ \vdots \\ \dfrac{\mathbb{P}(y_{t+1} \mid h_{t+1} = h_n) \cdot \mathbb{P}(h_{t+1} = h_s \mid \pi_t, u_t)}{\mathbb{P}(y_{t+1} \mid \pi_t, u_t)} \end{bmatrix} \right) \\
&= \sum_{y_{t+1}} V_{t+1} \left(\begin{bmatrix} \mathbb{P}(y_{t+1} \mid h_{t+1} = h_1) \cdot \mathbb{P}(h_{t+1} = h_1 \mid \pi_t, u_t) \\ \vdots \\ \mathbb{P}(y_{t+1} \mid h_{t+1} = h_n) \cdot \mathbb{P}(h_{t+1} = h_s \mid \pi_t, u_t) \end{bmatrix} \right).
\end{aligned} \tag{2.4}$$

Since V_{t+1} is piece-wise linear, $\mathbb{P}(y_{t+1} \mid \pi_t, u_t)$ in 2.4 will cancel out. Also, the effect of observation is just $\mathbb{P}(y_{t+1} \mid h_{t+1})$, which is independent of π_t, u_t. So, the summation is a weighted average:

$$\mathbb{P}(h_{t+1} \mid \pi_t, u_t) = \sum_{h_t} \mathbb{P}(h_{t+1} \mid h_t, u_t) \cdot \pi_t(h_t). \tag{2.5}$$

Substituting equation 2.5 into 2.4:

$$\begin{aligned}
&\sum_{y_{t+1}} V_{t+1} \left(\begin{bmatrix} \mathbb{P}(y_{t+1} \mid h_{t+1} = h_1) \cdot \sum_{h_t} \mathbb{P}(h_{t+1} = 1 \mid h_t, u_t) \cdot \pi_t(h_t) \\ \vdots \\ \mathbb{P}(y_{t+1} \mid h_{t+1} = h_n) \cdot \sum_{h_t} \mathbb{P}(h_{t+1} = s \mid h_t, u_t) \cdot \pi_t(h_t) \end{bmatrix} \right) \\
&= \sum_i \alpha'_{i,t}(u_t) \cdot \pi_{i,t} = \alpha'_t(u_t) \cdot \pi_t.
\end{aligned} \tag{2.6}$$

(2.6) is a piece-wise linear function in π_t and the number of α-vectors is finite. For each α-vector, there is a control action associated with it.

(2.2) can be rewritten as minimization over m different possible actions. Each value is associated with one control action:

$$\ell(u, H) \cdot \pi_t + \sum_y V_{t+1}(\phi(\pi_t, u, y)). \tag{2.7}$$

If we expand the summation over y, then each term is a weighted value of $V_{t+1}(\pi_{t+1})$ given (π_t, u, y). Also, knowing (π_t, u, y) is sufficient to indicate index $\ell(\pi_t, u, y)$ which specifies in what region the next belief state lies. Summation over y has q terms, each of them has different observation and for each of them $\ell(\pi_t, u, y)$ can be different. Therefore, (2.7) depends on $\ell(\pi_t, u, y)$. From (2.5), we see that (2.7) is linear in π_t and a function of control action. Therefore, expected for control action u can be written as:

$$\begin{aligned}
&\alpha_t^{\ell(\pi,y,u)}(u) \cdot \pi_t \triangleq \ell(u, H) \cdot \pi_t + \sum_y V_{t+1}(\phi(\pi_t, u, y)), \\
&V_t(\pi_t) = \min_{u_t}\{\alpha_t^{\ell(\pi_t,y,u_t)}(u) \cdot \pi_t\}.
\end{aligned} \tag{2.8}$$

In the worst case scenario, (2.7) may have N (size of all α-vectors in time $t+1$) different α-vectors. Since minimization is over m actions. $N \cdot m$ α-vectors is generated at time t. We know that at $t = T$, $N = m$, therefore, for the worst case, there will be m^{T-t+1} α-vectors. Size of α-vectors grows exponentially in time and this is why assumption of finite horizon is needed.

2.3.2 Pruning algorithms

We have discussed how to represent a piece-wise linear and concave function with a set of α-vectors, $\mathcal{A}$, but there are a number of issues that will arise concerning this representation in the algorithmic approach to the single step of value iteration. As shown in Figure 2–2, there are *useless* and *dominant* vectors in the representation of a value function, i.e. α_4,α_5 and α_6. One can even add an arbitrary number of vectors without changing the representation. In fact, it is shown [24] that any PWLC value function does have minimal representation. The term *parsimonious set* is used when referring to the unique minimal set of vectors, and we can do a reduction or *pruning* procedure to compute this set.

Given a set of vectors, $\mathcal{A}$, representing a value function over information space, we can define a partition of the information space where the partition has a finite number of elements, one for each α in $\mathcal{A}$. Additionally, each $\alpha \in \mathcal{A}$ has a set or region of belief states, $R(\alpha) \subseteq \Pi(X)$, where it dominates, which is,

$$R(\alpha) = \{\pi \in \Pi \mid \pi \cdot \alpha < \pi \cdot \tilde{\alpha},\ \forall \tilde{\alpha} \in \mathcal{A} - \{\alpha\}\}. \tag{2.9}$$

Note that because of the strict inequality in this definition, some belief states can be in the region of none of the vectors in A, which makes it not quite a true partition of the information space. The set of points which are not in any region define the borders of the partition and are points where more than one vector gives the same minimal value. Figure 2–3 shows a value function over information space with the partition it imposes on the information space along the horizontal axis.

How to generate a finite set of points so that we can construct all of the vectors in value function is the heart of the matter. The idea is simply generate all possible vectors that one can construct, this was proposed by Monahan in 1982 [21], also mentioned by Sondik in 1971. To construct a vector requires selection an action and a vector in value function for each observation. Among the large number of vectors, many are not useful, since they are completely dominated by other vectors over the

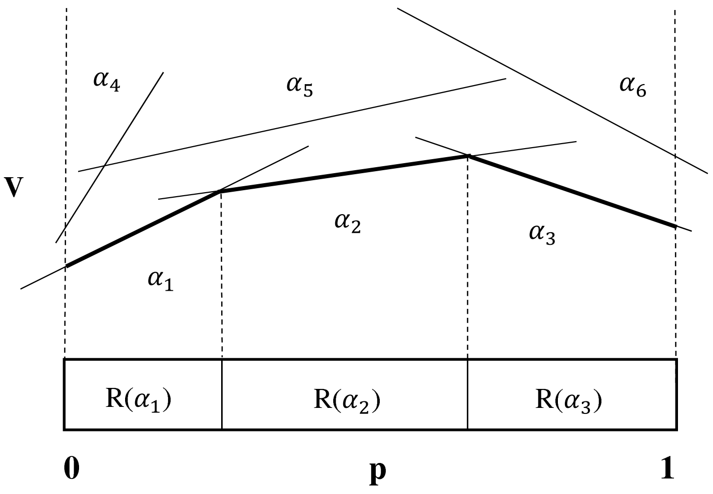

Figure 2–3: An example of the partition.

entire belief space. Although we can eliminate the useless ones at the expense of some computing time using linear programming, enumerating all the vectors takes a long time even for some small problems. As a side note, it is possible to construct a problem where each of those vectors enumerated is useful over some portion of belief space. The complexity of linear programming is polynomial, however, the number of linear programs to be solved is exponential in horizon (as shown in the last part of Section 2.3.1). Thus, in the worst case, the complexity grows exponentially in the horizon.

If we were able to generate only the necessary vectors, many problems may have a tractable solution: Cheng proposed the linear support algorithm [25], *linear support* comes from the idea that dominate vectors provide support to the value function. Cheng's procedure incrementally generates the set of support vectors by searching for the vertex that witnesses the missing support vector providing the greatest improvement to the value function, this vertex is called the *witness*. Although the algorithm generates only the necessary support vectors, it may take exponential amount of time to find a vertex that witnesses a missing support vector since a support region may have an exponential number of vertices and consequently the number of vertices in the current approximation can be exponential [24].

The next approach, Littman, et al. [24], attacked the problem differently. This algorithm, called *Witness* algorithm, uses the same structure as Sondik and Cheng; Unlike Sondik's algorithm, it does not worry about all the actions all the time. It concentrates on finding the best value function for each of the actions separately. Once a witness is found, the support conditional plan that maximizes the expected total return for that witness is easily computed by a one-step lookahead.

To avoid generating useless α-vectors, the linear support and witness algorithm introduced special search procedures to determine belief states that witness the support vectors. However, compared with Sondik's and Monahan's algorithms, they are conceptually difficult to implement. Zhang and Liu [26] proposed a new algorithm called incremental pruning that achieves both simplicity and computational efficiency. Later on, Zhang and Zhang [27] improved the running time of incremental pruning by point-based method. The point-based version provides substantial time saving in practice, however the number of α-vectors may still grow exponentially.

Apart from the value iteration algorithms mentioned above, Sondik [28], Hansen [29], and Meuleau et al. [30] also proposed *policy iteration* algorithms that conduct an iterative search directly within the space of polices. In practice, because the running time of a policy evaluation step is negligible compared to value iteration, the policy iteration algorithm usually runs faster.

2.4 Asymptotic Results

By representing the value function using α-vectors, although, we can precisely represent the value function, however, the number of α-vector grows exponentially with time. Even though we can use pruning algorithms to get rid of useless α-vectors, the complexity is still very high. One way to get around is to use some asymptotic results to directly compute the value function instead of doing dynamic programming. These results were originally presented in [31], we summarize them by closely following the presentation of [32].

For a Markov process that starts from an arbitrary belief state p_0, then at time step t,

$$p_t = \mathbb{P}(H = h_0 \mid Y_{1:t}) = \frac{\mathbb{P}(H = h_0, Y_{1:t})}{\mathbb{P}(Y_{1:t})}, \tag{2.10}$$

then,

$$(1 - p_t) = \mathbb{P}(H = h_1 \mid Y_{1:t}) = \frac{\mathbb{P}(H = h_1, Y_{1:t})}{\mathbb{P}(Y_{1:t})}, \tag{2.11}$$

where,

$$\mathbb{P}(H = h_0, Y_{1:t}) = \mathbb{P}(H = h_0) \cdot \mathbb{P}(Y_{1:t} \mid H = h_0) = p_0 \cdot f_0(y_0) \cdot f_0(y_1) \cdots f_0(y_t).$$

The same apply for $\mathbb{P}(H = h_1, Y_{1:t})$.

Use α to denote the lower threshold and β to denote the upper threshold. When we choose to continue, p_t must satisfy $\alpha < p_t < \beta$, therefore,

$$\frac{\alpha}{1-\alpha} < \frac{p_t}{1-p_t} < \frac{\beta}{1-\beta}. \tag{2.12}$$

As we defined in (2.10) and (2.11), (2.12) can be rewritten as:

$$\frac{\alpha}{1-\alpha} \cdot \frac{1-p_t}{p_t} < \frac{f_0(y_1) \cdot f_0(y_2) \cdots f_0(y_t)}{f_1(y_1) \cdot f_1(y_2) \cdots f_1(y_t)} < \frac{\beta}{1-\beta} \cdot \frac{1-p_t}{p_t}.$$

To simplify the expression, we define

$$\frac{\alpha}{1-\alpha} \cdot \frac{1-p_t}{p_t} = \frac{1}{B} \quad \text{and} \quad \frac{\beta}{1-\beta} \cdot \frac{1-p_t}{p_t} = \frac{1}{A},$$

then,

$$A < \frac{f_1(y_1) \cdot f_1(y_2) \cdots f_1(y_t)}{f_0(y_1) \cdot f_0(y_2) \cdots f_0(y_t)} < B. \tag{2.13}$$

If we stop and declare h_0, it satisfies that

$$f_1(y_1) \cdot f_1(y_2) \cdots f_1(y_t) \leq A \cdot f_0(y_1) \cdot f_0(y_2) \cdots f_0(y_t), \tag{2.14}$$

which makes

$$\mathbb{P}(U = h_0 \mid H = h_1) \leq A \cdot \mathbb{P}(U = h_0 \mid H = h_0). \tag{2.15}$$

If we stop and declare h_1, it satisfies that

$$f_1(y_1) \cdot f_1(y_2) \cdots f_1(y_t) \geq B \cdot f_0(y_1) \cdot f_0(y_2) \cdots f_0(y_t), \tag{2.16}$$

which makes

$$\mathbb{P}(U = h_1 \mid H = h_1) \geq B \cdot \mathbb{P}(U = h_1 \mid H = h_0). \tag{2.17}$$

It is known that $\mathbb{P}(u = h_0 \mid H) + \mathbb{P}(u = h_1 \mid H) = 1$. If the final value of the likelihood ratio in relation (2.13) when decision h_0 is chosen is generally close to the lower limit A, then the two sides of the inequality (2.14) will be approximately equal. If the final value of this likelihood ratio when decision h_1 is chosen is generally close to the upper limit A, then the two sides of the inequality (2.16) will also be approximately equal. Hence, as an approximation, we can regard the relation (2.14) and (2.16) as equality and can write the following relations:

$$\mathbb{P}(U = h_0 \mid H = h_0) \approx \frac{B-1}{B-A}, \quad \mathbb{P}(U = h_0 \mid H = h_1) \approx \frac{A(B-1)}{B-A}. \tag{2.18}$$

If we take log of equation (2.13), we get

$$\log A < \log\left[\frac{f_1(y_1)}{f_0(y_1)}\right] + \log\left[\frac{f_1(y_2)}{f_0(y_2)}\right] + \cdots + \log\left[\frac{f_1(y_n)}{f_0(y_n)}\right] < \log B.$$

We define the random variable Z_i as follows:

$$Z_i = \log\frac{f_1(Y_i)}{f_0(Y_i)}.$$

Furthermore, we shall let $a = \log A < 0$ and $b = \log B > 0$. Then the sequential probability ratio test defined by relation (2.13) specified that sampling should be continued whenever the following relation is satisfied:

$$a < \sum_{i=1}^{n} Z_i < b.$$

When either value $H = h_i, (i = 0, 1)$ is given, the random variables $Y_1, Y_2, \ldots$ are independent and identically distributed. Hence the random variable $Z_1, Z_2, \ldots$ also have the same properties. It is proved by Wald that if $Z_1, Z_2, \ldots$ is a sequence of independent and identically distributed random variables such that $\mathbb{E}(Z_i) = m$. For any sequential procedure of which $\mathbb{E}(N) < \infty$, the following relations must be satisfied:

$$\mathbb{E}(Z_1 + \cdots + Z_N) = m\mathbb{E}(N).$$

Let Z denote a random variable having the common distribution of each of the random variables $Z_1, Z_2, \ldots$. Then it can be seen from above that the following relation must be satisfied for $i = 0, 1$:

$$\begin{aligned} &\mathbb{E}(Z \mid H = h_i) \cdot \mathbb{E}(N \mid H = h_i) \\ &= \mathbb{E}(Z_1 + \cdots + Z_N \mid H = h_i) \\ &= \sum_{j=1}^{2} \mathbb{E}(Z_1 + \cdots + Z_N \mid U = h_j, H = h_i) \cdot \mathbb{P}(U = h_j \mid H = h_i). \end{aligned} \tag{2.19}$$

If we ignore the difference between the value of the terminating sum $Z_1 + \cdots + Z_N$ and either the boundary value a or b, we can approximate the final conditional expectations in equation (2.19) by the following simple values:

$$\begin{aligned} \mathbb{E}(Z_1 + \cdots + Z_N \mid U = h_0, H = h_i) &\approx a, \\ \mathbb{E}(Z_1 + \cdots + Z_N \mid U = h_1, H = h_i) &\approx b. \end{aligned} \tag{2.20}$$

Also, we can approximate the probabilities $\mathbb{P}(U = h_j \mid H = h_i)$ by applying equation (2.18). Since $A = e^a$ and $B = e^b$, the results obtained from equation (2.19) and (2.20) are

$$\begin{aligned} \mathbb{E}(N \mid H = h_0) &\approx \frac{a(e^b - 1) + b(1 - e^a)}{(e^b - e^a) \cdot \mathbb{E}(Z \mid H = h_0)}, \\ \mathbb{E}(N \mid H = h_1) &\approx \frac{a(e^{a+b} - e^a) + b(e^b - e^{a+b})}{(e^b - e^a) \cdot \mathbb{E}(Z \mid H = h_1)}. \end{aligned} \tag{2.21}$$

The procedure for determining the values of a and b that minimize the value function is, in general, quite complicated. However, when the sampling cost c is small, we can make the following approximations. If the sampling cost is small, the optimal procedure will typically involve taking many observations. In such a case, the bound a and b of the optimal procedure will be far apart, which makes $-a$ and b both very large number. Therefore, we can have the following approximations:

$$\mathbb{P}(U = h_1 \mid H = h_0) \approx e^{-b}, \quad \mathbb{P}(U = h_0 \mid H = h_1) \approx e^{a}.$$
$$\mathbb{E}(N \mid H = h_0) \approx \frac{a}{\mathbb{E}(Z \mid H = h_0)}, \quad \mathbb{E}(N \mid H = h_1) \approx \frac{b}{\mathbb{E}(Z \mid H = h_1)}.$$

These approximations are verified in our simulation when sampling cost c is small.

For a special case of Gaussian noise, when the detectors' observations are described by $y_i(t) = h + w_i(t)$, where $w_i(t)$ are zero mean white Gaussian noise sequences with variance σ; From statistical sequential analysis [8], it is mentioned that the average number of observations required to reach a decision is approximately

$$\mathbb{E}(N \mid h_0) = -2\sigma \left[\epsilon_1 \cdot \log\frac{1-\epsilon_2}{\epsilon_1} + (1-\epsilon_1) \cdot \log\frac{\epsilon_2}{1-\epsilon_1}\right],$$
$$\mathbb{E}(N \mid h_1) = -2\sigma \left[(1-\epsilon_2) \cdot \log\frac{1-\epsilon_2}{\epsilon_1} + \epsilon_2 \cdot \log\frac{\epsilon_2}{1-\epsilon_1}\right].$$

where,

ϵ_1 is the probability of error type 1, that is, the probability of deciding $H = h_1$ when $H = h_0$;

ϵ_2 is the type 2 error, that is, the probability of deciding $H = h_0$ when $H = h_1$;

σ is the variance for Gaussian noise sequences $\{w(t)\}$ as defined in $y(t) = h + w(t)$.

2.5 Method 1: Grid-based Discretization

Although the asymptotic expression is computationally easy, it's application requires special property of the problem, such as $c \ll \min\{\ell(u,h) : u,h \in \{h_0,h_1\}\}$ or the noise needs to be Gaussian. Since the complexity comes from the fact that the belief space is a continuous domain, another way to get around the continuous nature of the belief space is to discretize it, i.e. we select a finite grid points and store the corresponding values. The value of all other belief states are interpolated from the values of the grid points.

Generally, grid-based algorithms [23, 33, 34, 35] vary depending on the regularity of the grid, the resolution of the grid and the interpolation technique. Lovejoy [23] proposed a *fixed-resolution regular grid* that selects grid points equally spaced. This is an efficient interpolation approach based on triangulation concept, however, as the resolution increases, the number of grid points grows exponentially with the size of state space. As an improvement, Hauskrecht [34] proposed *variable-resolution non-regular* grids which allows to increase resolution in poor accuracy areas by adding grid points that are not necessarily equally spaced. However, because grid points are unevenly spaced, interpolation techniques are more computationally intensive. Recently, Zhou and Hansen [35] proposed a *variable-resolution regular* grid that achieves both increasing resolution in only necessary areas and fast interpolation.

In this section, we first introduce two discretization approaches and then implement Lovejoy's grid-based algorithm using these discretization methods separately.

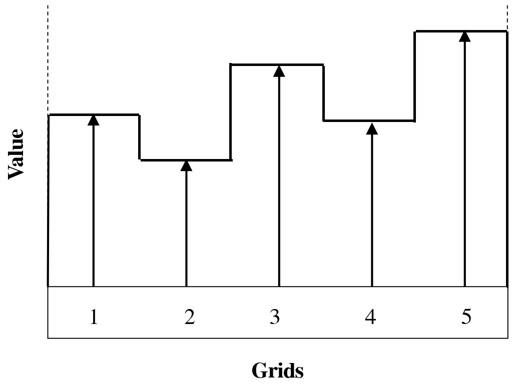

Figure 2–4: Illustration of zeroth-order value function approximation.

2.5.1 Zeroth and first order discretization

In general, orders of approximation refer to formal or informal terms for how precise an approximation is. A zeroth-order approximation of a function does a piece-wise constant approximation while the first-order approximation does a piece-wise linear approximation. An illustration of zeroth-order and first-order approximation of value function with 5 grids are shown in Figure 2–4 and Figure 2–5. To discretize the continuous belief space for a POMDP, we follow the method proposed by Lovejoy [23], which is to divide the belief space into equally spaced grid points set $S_m = \{0, \frac{1}{m}, \ldots, 1\}$, if one wants to discretize the space [0, 1] into m parts. When the grids are selected, each point s_i will be the *a prior* probability p_t as we described earlier in the model, after taking observation y_t, the posterior probability is calculated using Bayes rule:

$$p_{t+1} = \frac{p_t f_0(y_t)}{p_t f_0(y_t) + (1 - p_t) f_1(y_t)}.$$

p_{t+1} may fall into some point on the belief space, but not necessarily one of the selected grid points. In the approximate of value function, we need to calculate the

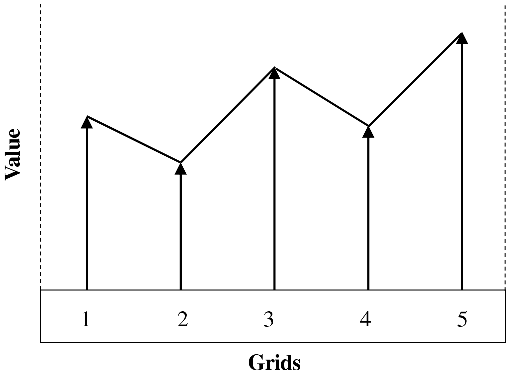

Figure 2–5: Illustration of first-order value function approximation.

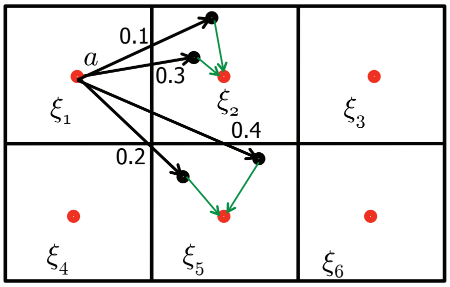

Figure 2–6: Illustration of zeroth-order discretization [2].

transition probability $P_{ij} = \mathbb{P}(p_+ = g_j \mid p = g_i, y)$. However, in this case, $p_{t+1} \neq p_+$, that's where the zeroth-order hold and first-order hold approximation come in.

Zeroth-order Approximation:

For zeroth-order approximation we do a deterministic transition onto the nearest vertex, that is, we approximate p_{t+1} to the nearest grid of it. An illustration of zeroth-order approximation is shown in Figure 2–6.

In Figure 2–6, $\{\xi_1, \cdots, \xi_6\}$ are the selected grid points. Action a takes grid ξ_1 to points in region of ξ_2 and ξ_5, therefore, we approximate these points to ξ_2 and ξ_5

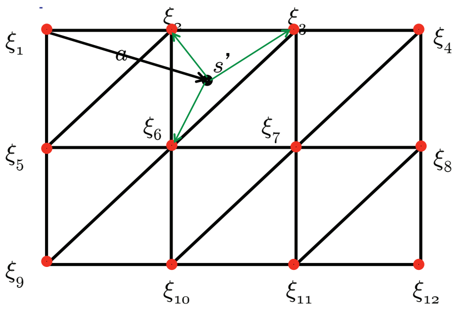

Figure 2–7: Illustration of first-order discretization [2].

using zeroth-order approximation, the transition probability between grids will be:

$$\mathbb{P}(\xi_2 \mid \xi_1, a) = 0.1 + 0.3 = 0.4;$$

$$\mathbb{P}(\xi_5 \mid \xi_1, a) = 0.4 + 0.2 = 0.6;$$

The transition probabilities between all other ξ_i are defined in the same way.

First-order Approximation:

For first-order approximation, we do a stochastic transition onto neighboring vertices, that is, we do a linear programming between p_{t+1} and n nearest neighbors of it based on the distance between p_{t+1} and its neighbors. Compared with zeroth-order approximation, this method gives more precise approximation. An illustration of first-order approximation is shown in Figure 2–7.

In Figure 2–7, $\{\xi_1, \cdots, \xi_{12}\}$ are the selected grid points. Action a takes the grid ξ_1 to s', which falls in the region of ξ_2, ξ_3 and ξ_6. Interpolation between grids is needed to calculate the transition probability.

$$\mathbb{P}(\xi_2 \mid \xi_1, a) = p_A,$$
$$\mathbb{P}(\xi_3 \mid \xi_1, a) = p_B,$$
$$\mathbb{P}(\xi_6 \mid \xi_1, a) = p_C,$$

such that

$$s' = p_A \xi_2 + p_B \xi_3 + p_C \xi_6.$$

An example: Here, we give an detailed illustration for zeroth-order and first-order discretization by a numerical example. The parameters are given as,

$$f_0 = [0.25 \quad 0.75], \quad f_1 = [0.6 \quad 0.4].$$

We discretize the belief space $[0, 1]$ into $m = 5$ equal space parts.

1. Zeroth-order

Middle point of each section is taken as grid points, therefore, for section i the grid point will be $p_i = \frac{2i-1}{2N}$, then different observation y take p_i to different posterior probability points. If observation y_i takes p_+ to the interval $[\frac{j-1}{N}, \frac{j}{N}]$, we increase the probability of transferring to section j by the probability of observing y_i. Therefore:

$$\begin{aligned} P_{ij} &= \sum_y \mathbb{1}\left(p_+ = j \mid p = i, y\right) \cdot \mathbb{P}(y \mid p = i) \\ &= \sum_y \mathbb{1}\left(\frac{j-1}{N} \le \frac{\frac{2i-1}{2N} \cdot \mathbb{P}(y \mid h_0)}{\frac{2i-1}{2N} \cdot \mathbb{P}(y \mid h_0) + (1 - \frac{2i-1}{2N}) \cdot \mathbb{P}(y \mid h_1)} \le \frac{j}{N}\right) \cdot \mathbb{P}(y \mid p = i). \end{aligned}$$

Taking $i = 3$ for further example, $p_i = 0.5$,

$$p_+^1 = \frac{0.5 \cdot 0.25}{0.5 \cdot 0.25 + (1 - 0.5) \cdot 0.6} = 0.294, \quad p_+^2 = \frac{0.5 \cdot 0.75}{0.5 \cdot 0.75 + (1 - 0.5) \cdot 0.4} = 0.652.$$

$p_+^1 = 0.294$ falls in $j = 2$ and $p_+^2 = 0.652$ falls in $j = 4$, therefore: $P_{31} = 0$,$P_{33} = 0$,$P_{35} = 0$, as no posterior probability falls in these intervals.

$$P_{32} = \mathbb{P}(Y = y_1 \mid p = i) = \mathbb{P}y_1 \mid h_0) \cdot \mathbb{P}(h_0 \mid p_i) + \mathbb{P}(y_1 \mid h_1) \cdot \mathbb{P}(h_1 \mid p_i) = 0.425,$$

$$P_{34} = \mathbb{P}(Y = y_2 \mid p = i) = \mathbb{P}(y_2 \mid h_0) \cdot \mathbb{P}(h_0 \mid p_i) + \mathbb{P}(y_2 \mid h_1) \cdot \mathbb{P}(h_1 \mid p_i) = 0.575.$$

For the reader's reference, the probability matrix is provided in Table 2–1.

Table 2–1: Zeroth order transition matrix for N = 5.

1.0000	0	0	0	0
0.4950	0	0.5050	0	0
0	0.4250	0	0.5750	0
0	0	0.3550	0	0.6450
0	0	0	0.2850	0.7150

2. First-order

In first-order approximation, if we discretize the belief space into N parts, we get $N + 1$ grids, and $p_i = \frac{i-1}{N}$ for section i. And if the posterior probability p_+ falls into section j, the previous P_{ij} in zeroth-order approximation will be divided by a parameter λ to P_{ij} and $P_{i(j+1)}$. λ is obtained by solving an interpolation function, such that $p_+ = \lambda \cdot p_j + (1 - \lambda) \cdot p_{j+1}$.

Taking $i = 3$ for further illustration, $p_i = 0.4$:

$$p_+^1 = \frac{0.4 \cdot 0.25}{0.4 \cdot 0.25 + (1 - 0.4) \cdot 0.6} = 0.217, \quad p_+^2 = \frac{0.4 \cdot 0.75}{0.4 \cdot 0.75 + (1 - 0.4) \cdot 0.4} = 0.555.$$

$p_+^1 = 0.217$ falls between $j = 2$ and $j = 3$, $p_+^2 = 0.555$ falls between $j = 3$ and $j = 4$.

To solve the interpolation, $0.217 = 0.2 \cdot \lambda + 0.4 \cdot (1 - \lambda) \Rightarrow \lambda = 0.915$, thus $P_{32} = 0.46 \cdot 0.915 = 0.42$. $p_+^2 = 0.555$ falls between $j = 3$ and $j = 4$. $0.555 = 0.4 \cdot \lambda + 0.6 \cdot (1 - \lambda) \Rightarrow \lambda = 0.222$, thus $P_{34} = 0.54 \cdot (1 - 0.222) = 0.42$ and $P_{33} = 0.46 \cdot 0.085 + 0.54 \cdot 0.222 = 0.16$.The transition probability between other grids is calculated following the same approach.

For the reader's reference, the probability matrix is provided in Table 2–2.

Table 2–2: First order transition matrix for N = 5.

1.0000	0	0	0	0	0
0.2800	0.4400	0.2800	0	0	0
0	0.4200	0.1600	0.4200	0	0
0	0.0300	0.3600	0.1900	0.4200	0
0	0	0	0.2800	0.4400	0.2800
0	0	0	0	0	1.0000

2.5.2 Dynamic program based on discretization

As we defined earlier, the dynamic program of value function for POMDP problem is given as:

$$V_t(p) = \min\{L_0(p), L_1(p), W_t(p)\},$$

where

$$\begin{aligned}
L_0(p) &= p\ell(h_0, h_0) + (1-p)\ell(h_1, h_0), \\
L_1(p) &= p\ell(h_0, h_1) + (1-p)\ell(h_1, h_1), \\
W_t(p) &= c + \int_y [pf_0(y) + (1-p)f_1(y)]V_{t+1}(\varphi(p, y))dy.
\end{aligned}$$

In grid based approximation, the latter part of $W(p)$ can be calculated based on the value function from previous time step $V_{t+1}(j)$ and the transition probability P_{ij}, where $j = \varphi(i, y)$. Thus the value function will become:

$$V_t(p) = \min\{L_0(p), L_1(p), c + \sum_{j \in S_m} P_{pj} V_{t+1}(j)\}, p \in S_m.$$

All we need is to pick select grid points, compute the transition matrix, approximate the value function and perform value iteration or policy iteration on the approximated value function.

2.6 Method 2: Absorption Probability of Markov Chain

In each time step of the dynamic programming, the value function is expressed as the minimum cost with respect to the optimal action among all three alternatives. However, if we change the view and look at the problem in another perspective, whenever a decision maker stop and make a decision, the value function is actually the sum of the expected loss of making wrong decision and the cost of taking all observations along.

Let N denote the stopping time when the decision maker decides to stop,

$$N = \min\{t \in \mathbb{Z} > 0 : U \in \{h_0, h_1\}\}.$$

Let $\ell(U, H)$ denote the loss of deciding U is true when the true hypothesis is H, and we set $\ell(h_0, h_0) = \ell(h_1, h_1) = 0$. Thus, for any strategy $g \in \mathcal{G}$, the value function can be written as

$$J(g) = \mathbb{E}[c \cdot N + \ell(U, H)]. \tag{2.22}$$

For any belief state p in $[0,1]$, (2.22) can be expanded as:

$$\begin{aligned} J(p,g) =& p \cdot [c \cdot \mathbb{E}[N \mid h_0, g] + \ell(h_1, h_0) \cdot \mathbb{P}(u = h_1 \mid h_0, g)] \\ &+ (1-p) \cdot [c \cdot \mathbb{E}[N \mid h_1, g] + \ell(h_0, h_1) \cdot \mathbb{P}(u = h_0 \mid h_1, g)]. \end{aligned} \tag{2.23}$$

Define

$$\xi_k(u, g) = \mathbb{P}(U = u \mid H = h_k, g), \ u \in \{h_0, h_1\}. \tag{2.24}$$

If we could compute for p, $\xi_k(u, g)$, $u \in \{h_0, h_1\}$ and $\mathbb{E}[N \mid H, g]$, then the value function at p given the strategy g is determined.

2.6.1 Preliminaries: absorption probability in Markov chain

Now consider an arbitrary absorbing Markov chain. Re-order the states so that the transient states come first. If there are r absorbing states and t transient states, the transition matrix can be written in the following canonical form:

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix}.$$

Where **I** is an $r \times r$ identity matrix, **0** is an $r \times t$ zero matrix, **R** is a nonzero $t \times r$ matrix, and **Q** is an $t \times t$ matrix. The first t states are transient and the last r states are absorbing both row-wise and column-wise.

It is proved that in an absorbing Markov chain, the probability that the process will be absorbed at last is 1 [36]. Now, we define the fundamental matrix

$$F = (I - Q)^{-1}.$$

Given that the chain starts in state s_i, the expected number of steps before the chain is absorbed is given as

$$t = F\mathbf{1},$$

where $\mathbf{1}$ is the column vector with all entries as 1.

Let b_{ij} be the probability that an absorbing chain will be absorbed in the absorbing state r_j if it starts in the transient state t_i. Let B be the matrix with entries b_{ij}. Then B is an $t \times r$ matrix and

$$B = FR.$$

2.6.2 Approximately computing ξ_k using absorption probability

Recall that after discretization, the space between [0, 1] is discretized to $S_m = \{0, \frac{1}{m}, \ldots, 1\}$. We can approximate the [0, 1]-valued Markov process $\{p_t\}_{t=1}^{\infty}$ by the S_m-valued Markov chain. The Markov process starts in one state and moves successively to another. Each move is called a *step.* If the chain is currently in state s_i, then it moves to state s_j at the next step with a probability denoted by P_{ij} (called *transition probability*), and this probability does not depend upon which states the chain was in before the current state. An initial probability distribution, defined on S_m, specifies the starting state. In our case, we assume uniform distribution.

We consider three approximations that make different assumptions on probability distribution of observation Y. Denote the corresponding transition probabilities by P_0, P_1 and P_*. For $P_k, k \in \{0, 1\}$, we assume that $Y \sim f_k$, for P_* we assume that $Y_{t+1} \sim q(\cdot \mid p_t)$, where $q(y_{t+1} \mid p_t) := p_t \cdot f_0(y_{t+1}) + (1 - p_t) \cdot f_1(y_{t+1})$. Using the first-order discretization method in Section 2.5.1, we can compute the transition probability matrix P_{ij}. Note that the transition probabilities $P_k, k \in \{0,1\}$, approximate the evolution of $\{p_t\}_{t=1}^{\infty}$ process when hypothesis $H = h_k$ is true.

For $H = h_k, k \in \{0,1\}$, given any threshold based strategy $g = \langle \alpha, \beta \rangle$ such that $\alpha, \beta \in S_m$, define sets $A_0, A_1 \subset S_m$ as:

$$A_0 = \{\beta, \beta + \frac{1}{m}, \ldots, 1\},$$

and

$$A_1 = \{0, \frac{1}{m}, \ldots, \alpha\}.$$

Then A_0 and A_1 are two absorbing states and every state in set $\{\alpha + \frac{1}{m}, \ldots, \beta - \frac{1}{m}\}$ is a transient state. Note that $\xi_k(h_0, g)$ corresponds to the event that the Markov process $\{p_t\}_{t=1}^{\infty}$ goes above the threshold β before it goes below the threshold α. This event is approximated by the event that the Markov chain with transition probability P_k that starts in p (which is assumed to belong to S_m) gets absorbed in the set A_0 before it is absorbed in the set A_1. A similar interpretation holds for $\xi_k(h_1, g)$.

Re-order P_k to $\hat{P}_k$ and follow the computation process described in Section 2.6.1, denote $B_k = (I - Q_k)^{-1} R_k$ and $T_k = (I - Q_k)^{-1}\mathbf{1}$. Then, we know that for any transient state s and $b \in \{0,1\}$, $[B_k]_{sb}$ is the probability that the Markov process starting in state s is absorbed in the set A_b. Use $\langle \alpha, \beta \rangle$ to represent threshold-based strategy. Thus,

$$\xi_k(h_b, \langle\alpha, \beta\rangle) \approx [B_k]_{s^*b},\ b \in \{0, 1\}, \tag{2.25}$$

$$\mathbb{E}(N \mid h_k, \langle\alpha, \beta\rangle) \approx [T_k]_{s^*b},\ b \in \{0, 1\}, \tag{2.26}$$

where s^* denotes the index of s in transient set $\{\alpha + \frac{1}{m}, \ldots, \beta - \frac{1}{m}\}$.

2.6.3 Direct search based on discretization

In the previous sections, by computing absorption probability, we can approximate methods for approximating $\mathbb{E}(N \mid H, g)$ and $\xi_k(U, g)$ for any discrete state p in set $\{\alpha + \frac{1}{m}, \ldots, \beta - \frac{1}{m}\}$. The cost at p is given as:

$$\begin{aligned} J(p, g) = & p \cdot [c \cdot \mathbb{E}[N \mid h_0, g] + \ell(h_1, h_0) \cdot \xi_0(h_1, g)] \\ & + (1 - p) \cdot [c \cdot \mathbb{E}[N \mid h_1, g] + \ell(h_0, h_1) \cdot \xi_1(h_0, g)]. \end{aligned} \tag{2.27}$$

Thus, we can come up with another approach to find the optimal strategy. The main idea of the approach is to approximately compute the performance of a strategy $g = \langle\alpha, \beta\rangle$, and optimize over g. For this reason, we call this approach *direct search*.

From previous knowledge, we know that the value function is not linear. Then the problem turns into an non-linear optimization problem. Since the relationship between value function and threshold cannot be expressed in analytic equations, we cannot take derivatives. In principle, such non-convex optimization problems can be solved using derivative-free methods that do not use numerical or analytic gradients. In this thesis, after reading some documentation, we choose one of the simplest algorithm: *fminsearch* in Matlab [37]. This step can be replaced by more sophisticated algorithms to obtain better results. However, it is not possible to guarantee that such algorithm will converge to globally optimal solution.

fminsearch uses the Nelder-Mead simplex algorithm as described in Lagarias et al [38]. This is a direct search method that does not use numerical or analytic gradients. If n is the length of x, a simplex in n-dimensional space is characterized by the $n+1$ distinct vectors that are its vertices. In two-space, a simplex is a triangle; in three-space, it is a pyramid. At each step of the search, a new point in or near the current simplex is generated. The function value at the new point is compared with the function's values at the vertices of the simplex and, usually, one of the vertices is replaced by the new point, giving a new simplex. This step is repeated until the diameter of the simplex is less than the specified tolerance [8].

The form of *fminsearch* is $x =$ fminsearch(*fun*,x_0,*options*), the function starts at the point x_0 and returns a value x that is a local minimizer of the function described in *fun*. x_0 can be a scalar, vector, or matrix. *fun* is a function handle. Additional optimization parameters can be specified in *options*.

To do this, we first write a function with $g = \langle \alpha, \beta \rangle$ as input and J as output using different approximate methods. $J(p, g)$ is computed following (2.27)). To reduce the dependence of the numerical results on the choice of a *a priori* probability p, we pick multiple values of p in a finite set in [0, 1], here we average over all p to get a mean value J. If $J(p, g)$ is computed exactly, then such an averaging will not affect the result of the optimization algorithm because the optimal strategy g does not depend on the choice of p. Then, this function can be used as *fun* in *fminsearch*, providing a start point x_0 and optimization parameters for *fminsearch*, the function will automatically search for optimal threshold that gives minimum value.

In the input of *fminsearch*, the parameter *fun* is the function we tell *fminsearch* to try a value and optimize on. Since this function cannot be expressed in equations, we first write the function in a Matlab *.m* file, then use the Matlab function handler [43] *@Fun* to pass it in. We need to do this for two separate cases, for each case, the pseudo code is shown in Algorithm 1.

Algorithm 1 Nonlinear minimum value search function

```
Require: $g = \langle \alpha, \beta \rangle$
Ensure: value
function MYFUN($g = \langle \alpha, \beta \rangle$)
    Specify $f_0, f_1$ and $\ell(U, H), c$
    Discretize [0, 1] into $S_m = \{0, \frac{1}{m}, \ldots, 1\}$
    if $\alpha < \beta$ then
        for $p \in S_m$ do
            Compute $\xi_0(U, g)$ and $\xi_1(U, g)$
            Compute $\mathbb{E}[N \mid H = h_0, g]$ and $\mathbb{E}[N \mid H = h_1, g]$
            Compute $J(p, g)$ follows equation (2.27)
        end for
        value = $\frac{1}{m} \sum_{i=1}^{m} V(p)$
    else
        value = $10^6$
    end if
end function
```

2.7 Method 3: Monte Carlo Simulation

An early variant of the Monte Carlo method can be seen in the Buffon's needle experiment, in which the circumference ratio π can be estimated by dropping needles on a floor made of parallel and equidistant strips. In the 1930s, Enrico Fermi first experimented with the Monte Carlo method while studying neutron diffusion, but did not publish anything on it [39]. In the 1950s they were used at Los Alamos for early work relating to the development of the hydrogen bomb, and became popularized in the fields of physics, physical chemistry, and operations research. The Rand Corporation and the U.S. Air Force were two of the major organizations responsible for funding and disseminating information on Monte Carlo methods during this time, and they began to find a wide application in many different fields.

The principle behind Monte Carlo simulation is that the behavior of a statistic in random samples can be assessed by the empirical process of actually drawing lots of random samples and observing this behavior. The strategy for doing this is to create an artificial *world*, or *pseudo-population*, which resembles the real world in all relevant respects. This pseudo-population consists of mathematical procedures for generating sets of numbers that resemble samples of data drawn from the true population. We then use this pseudo-population to conduct multiple trials of the statistical procedure of interest to investigate how that procedure behaves across samples.

The basic Monte Carlo procedure is as follows:

1. Specify the pseudo-population in symbolic terms in such a way that it can be used to generate samples. This usually means developing a computer algorithm to generate data in a specified manner.
2. Sample from the pseudo-population in ways reflective of the statistical situation of interest, for example, with the same sampling strategy, sample size and so forth.
3. Calculate $\hat{\theta}$ in the pseudo-sample and store it in a vector, $\hat{\theta}_{.}$

4. Repeat steps 2 and 3 t times, where t is the number of *trials* or samples.
5. Construct a relative frequency distribution of the resulting $\hat{\theta}_t$ values, which is the Monte Carlo estimate of the sampling distribution of $\hat{\theta}$ under the conditions specified by the pseudo-population and the sampling procedures.

2.7.1 Approximately compute ξ_k using MC

Clearly, Monte Carlo simulation is a very simple concept as it follows naturally from the conception of what a sampling distribution is. In our problem, $\xi_k(u, g)$ can be viewed as a conditioned distribution of a decision u given the hypothesis h_k and $\mathbb{E}(N \mid H, g)$ is the sampling length of a Monte Carlo simulation until a decision is made. Therefore, we can us Monte Carlo simulation as an approximation method. The complicated aspects of the technique are (a) writing the computer code to simulate the data conditions desired and (b) interpreting the estimated sampling distribution.

Given $g = \langle \alpha, \beta \rangle$, design the simulation procedure for each sampling process as:

1. Generate sample observation y_1 and y_2 according to PMF f_0 (or f_1).
2. Given an prior belief state (in this case is each grid point p), evolve p using Bayes rule, if $p_+ < \alpha$, stop and output $u_s = 1$ (which means deciding $u_s = h_1$), else if $p_+ > \beta$, stop and output $u_s = 0$ (which means deciding $u_s = h_0$), else continue.
3. In the mean time, keep track of the number of samples τ_s needed to reach a decision.
4. Repeat step 1, 2, and 3 for S times, where S is the sampling size.

After doing the above procedure,

$$\xi_0(h_1, g) = \frac{1}{|S|}\sum_s (u_s \mid h_0),$$
$$\mathbb{E}(N \mid h_0, g) = \frac{1}{|S|}\sum_s (\tau_s \mid h_0).$$

Note that by evaluating the variance of sampled results, we can have an idea of how precise the simulation is compared to true value. When we estimate mean $\mu = \mathbb{E}(X)$ of a distribution by collecting n i.i.d. samples from the distribution, $X_1, \cdots, X_n$ and the sample mean

$$\bar{X}(n) = \frac{1}{n}\sum_{j=1}^{n} x_j.$$

Let $\sigma^2 = Var(X)$ denote the variance of the distribution

$$Var(\bar{X}(n)) = \frac{\sigma^2}{n}.$$

However, in practice, we would not know the value of σ^2; We instead use an estimate for it, the sample variance $S^2(n)$:

$$S^2(n) = \frac{1}{n-1}\sum_{j=1}^{n}(x_j - \bar{X}_n)^2.$$

It can be shown that $S^2(n) \to \sigma^2$ with probability 1 as $n \to \infty$ and $\mathbb{E}(S^2(n)) = \sigma^2$. So, we use $S(n)$ in place of σ when constructing our confidence intervals.

2.7.2 Direct search based on MC

The idea is the same as Section 2.6.3, only that when we are approximating $\xi_k(U, g)$ and $\mathbb{E}(N \mid H, g)$, we use Monte Carlo simulation this time.

2.8 Case Study

In this section, we present some numerical results of the methods described above and conclude this chapter.

Consider the following sequential hypothesis testing problem: There are two hypothesis h_0 and h_1, and two observations y_1 and y_2. Given hypothesis h_k, observations Y are distributed according to PMF f_k.

$$f_0 = [0.25 \quad 0.75], \quad f_1 = [0.6 \quad 0.4].$$

The cost are given as:

$$\ell(U = h_0, H = h_1) = \ell(U = h_1, H = h_0) = 20,$$
$$\ell(U = h_0, H = h_0) = \ell(U = h_1, H = h_1) = 0.$$

The cost of making an observation c is specified in different cases.

2.8.1 Evolution of α-vectors

In this part, we will show how to use α-vectors to represent value function and find the threshold. To reduce the complexity of constructing α-vectors, we set $c = 1$ in this example.

With the parameters given as above, we implemented the witness algorithm by:

1. Construct a new set of α-vectors A_{t+1} from A_t computed in the previous step;
2. Find the parsimonious set of α-vectors that represent the value function and only keep track of those α-vectors;
3. Calculated the upper and lower bound $g_t = \langle \alpha_t, \beta_t \rangle$ of the decision policy;

4. Keep doing step 1, 2, 3 until the upper and lower bound remain steady.

Evolution of α-vectors until $T = 6$ is shown in Figure 2–8, where α is the lower bound and β is the upper bound. As shown in Figure 2–8, in each iteration, the number of α-vectors increase and the lower and upper bound change accordingly. Eventually, at some step, the number of α-vector and the threshold will remain the same.

2.8.2 Approximate $\xi_k(u, g)$ and $\mathbb{E}(N \mid H)$

In this part, we arbitrarily pick $g = \langle \alpha, \beta \rangle = \langle 0.003, 0.997 \rangle$ and compute $\xi_k(u, g)$ and $\mathbb{E}(N \mid H)$ using the asymptotic expression, absorption probability of Markov chain and Monte Carlo simulation. For the discretization, we choose $m = 1000$ and for the Monte Carlo simulation, we set $s = 10^6$ samples. These results do not depend on the choice of c.

Asymptotic Approximation Result:

For the asymptotic approximation, we just need to plug the numbers in the asymptotic expression,

$$\xi_0(h_1, g) \approx \frac{\alpha(1-p)}{(1-\alpha)p} = \frac{1}{B} = 0.007,$$
$$\xi_1(h_0, g) \approx \frac{(1-\beta)p}{\beta(1-p)} = A = 0.0013.$$

For the expected observations needed, we can compute the KL-divergence as:

$$\mathbb{E}[Z \mid H = h_0] = \mathbb{E}\left[\log \frac{f_2(Y)}{f_1(Y)} \mid H = h_0\right] = \sum_y \left[\log \frac{f_2(y)}{f_1(y)}\right] \cdot f_1(y).$$

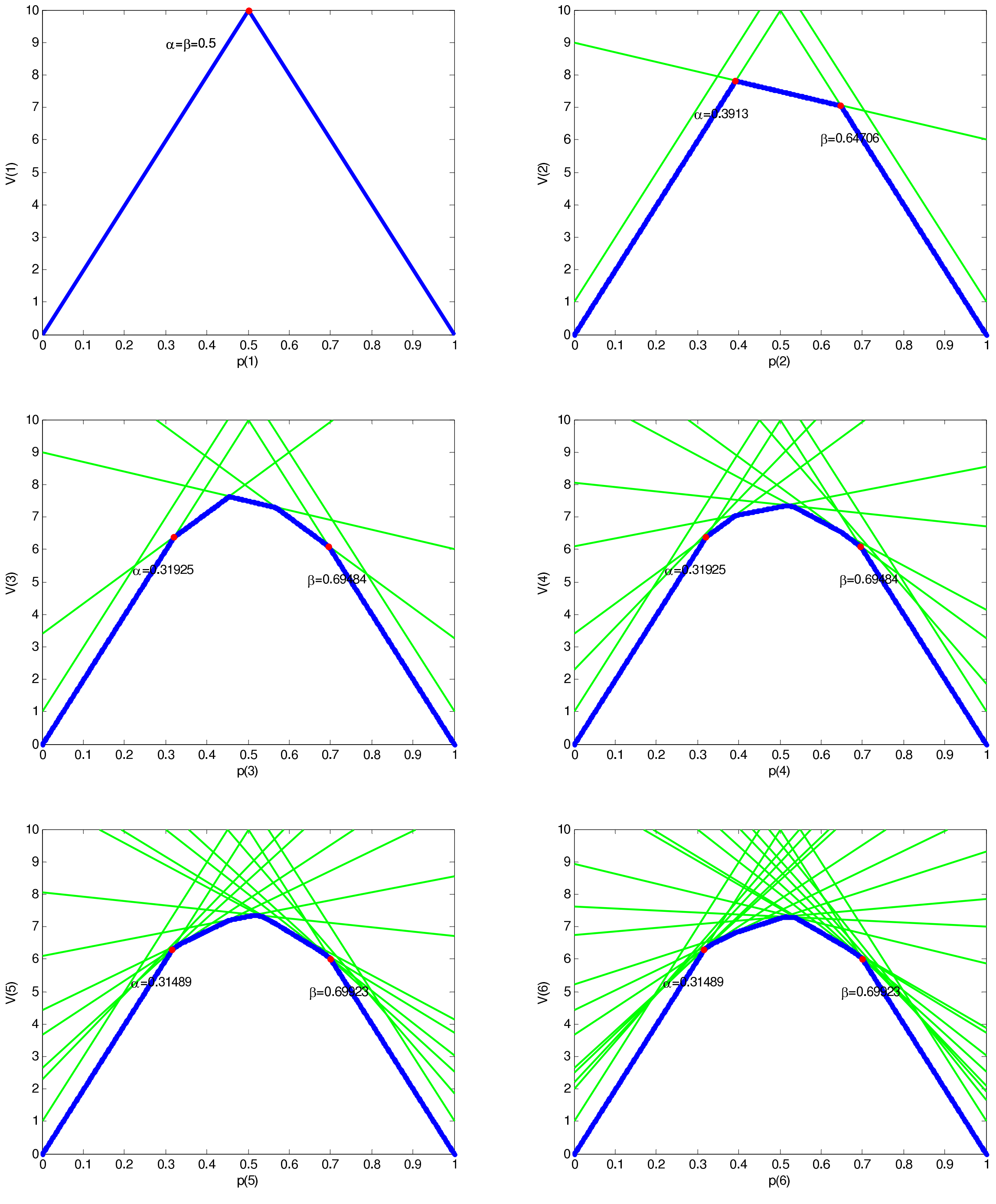

Figure 2–8: Value function represented by α-vectors over six time steps.

Thus,

$$\begin{aligned}\mathbb{E}[Z \mid H = h_0] &= \left(\log \frac{0.6}{0.25}\right) \cdot 0.25 + \left(\log \frac{0.4}{0.75}\right) \cdot 0.75 = -0.2526,\\ \mathbb{E}[Z \mid H = h_1] &= \left(\log \frac{0.6}{0.25}\right) \cdot 0.6 + \left(\log \frac{0.4}{0.75}\right) \cdot 0.4 = 0.2738.\end{aligned}$$

Since $a = \log A = -6.6454$, $b = -\log B = 4.9618$, then:

$$\begin{aligned}\mathbb{E}[N \mid H = h_0] &\approx \frac{a}{\mathbb{E}[Z \mid H = h_0]} = 26.308,\\ \mathbb{E}[N \mid H = h_1] &\approx \frac{b}{\mathbb{E}[Z \mid H = h_1]} = 18.122.\end{aligned}$$

Method 2: Absorption of Markov Chain:

Follow Section 2.6, compute the transition matrix P_k using first-order discretization, and compute B_0, T_0 and B_1, T_1 accordingly, then find the result with the index of $p = 0.3$, the results are:

$$\begin{aligned}&\xi_0(h_1, g) \approx 0.0041, \quad \mathbb{E}(N \mid H = h_0) \approx 28.101;\\ &\xi_1(h_0, g) \approx 0.0008, \quad \mathbb{E}(N \mid H = h_1) \approx 20.199.\end{aligned}$$

Method 3: Monte Carlo Simulation:

Follow Section 2.7, the approximated value using Monte Carlo simulation are:

$$\begin{aligned}&\xi_0(h_1, g) \approx 0.005, \quad \mathbb{E}(N \mid H = h_0) \approx 27.150;\\ &\xi_1(h_0, g) \approx 0.0013, \quad \mathbb{E}(N \mid H = h_1) \approx 19.439.\end{aligned} \tag{2.28}$$

Also, the variance for simulation result is calculated accordingly, $\text{Var}_1 = 4.888 \times 10^{-4}$ for hypothesis $H = h_0$, $\text{Var}_2 = 1.349 \times 10^{-4}$ for hypothesis $H = h_1$. Therefore,

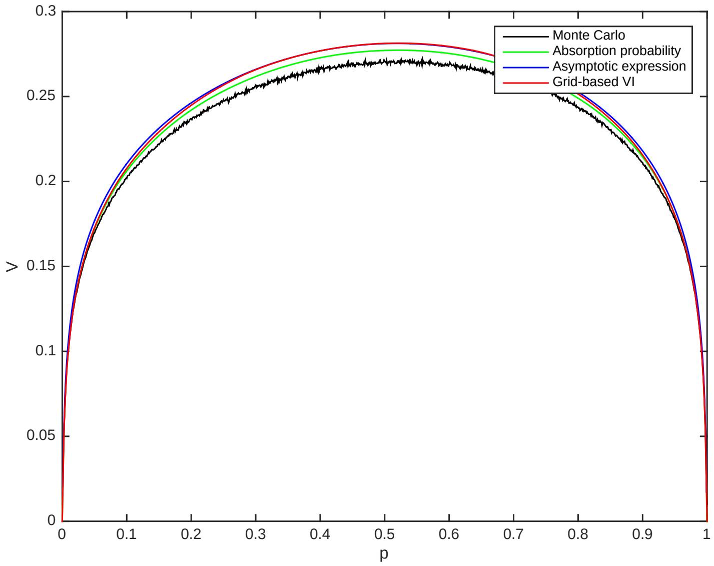

Figure 2–9: Comparison of approximated value functions.

combined with (2.28), a more accurate to present the result should be:

$$\xi_0(h_1, g) \approx 0.005 \pm 0.0022, \quad \xi_1(h_0, g) \approx 0.0013 \pm 0.0012.$$

In order to have an idea of which approximation method has better performance in this case, we can compute $\xi_k(U, g)$, $\mathbb{E}(N \mid H)$ for each p in set S_m then plug the approximation result in (2.27). Thus, the value function is constructed, we can compare approximated value function by three methods to the value function by grid-based value iteration method in Section 2.5. The result is shown in Figure 2–9.

2.8.3 Finding optimal strategy

In this part, we will show the threshold-based strategy computed by the methods described in this chapter. For value function represented by α-vectors, the threshold is computed by finding the minimum and maximum cross point of every pair of α-vectors. For grid-based discretization method, the threshold is computed by solving the dynamic program using value iteration. For approximation method shown in Section 2.8.2, the threshold is found by the non-linear optimization function *fminsearch*.

Now, consider the infinite horizon case of the sequential hypothesis testing for one decision maker. Method 2 and Method 3 are already in infinite horizon scenario. For α-vector and grid-based discretization, we can use standard technique in infinite horizon dynamic programming to discount the problem. Note that, for grid-based discretization, we only present the results of first-order discretization here.

In case 1, set $c = 1$, the result is provided in Table 2–3. In case 2, change the value of c to 0.1. the result is provided in Table 2–4. In case 3, we change the c to 0.01 and show the result in Table 2–5.

Table 2–3: Thresholds obtained by different methods when $c = 1$.

Method	Thresholds	min cost	run time
α-vector	[0.3139 0.7018]	4.5022	5745sec
Asymptotic	[0.1544 0.8349]	6.3409	0.2 sec
Method 1	[0.3050 0.7060]	4.4961	0.4 sec
Method 2	[0.3045 0.7046]	4.4961	0.49 sec
Method 3	[0.3089 0.7014]	4.9674	255 sec

2.9 Discussion

In this chapter, we investigated centralized sequential hypothesis testing problem. On computing the value function, we present mainly three ways, the first one is to represent the value function on a continuous state MDP using α-vectors;

Table 2–4: Thresholds obtained by different methods when $c = 0.1$.

Method	Thresholds	min cost	run time
Asymptotic	[0.0180 0.9806]	1.5103	0.03 sec
Method 1	[0.03 0.9710]	1.4578	1.115 sec
Method 2	[0.0304 0.9698]	1.4553	5.0 sec
Method 3	[0.0285 0.9708]	1.4547	298 sec

Table 2–5: Thresholds obtained by different methods when $c = 0.01$.

Method	Thresholds	min cost	run time
Asymptotic	[0.0018 0.9980]	0.2387	0.2 sec
Method 1	[0.003 0.998]	0.2416	1.24 sec
Method 2	[0.0035 0.9965]	0.2389	4.8 sec
Method 3	[0.0024 0.9976]	0.2346	356 sec

the second one is to discretize the continuous space [0, 1] into discrete states and approximate the value function using different methods. Apart from the standard value iteration of dynamic program, we propose another three approximate methods: asymptotic expression, absorption probability of Markov chain and Monte Carlo simulation. On finding the optimal threshold-based stopping rule of the problem, we propose two ways, the first way is to use dynamic program and find the lower and upper bound of each time step until it remains steady; the second way is to evaluate an arbitrary strategy and use non-convex optimization (we call direct search) to find the strategy that minimize the value function. As shown in the case study, the discretized value function is a good approximation of the exact one. Among three approximation methods, asymptotic expression is computationally fast but has limited applications; Monte Carlo simulation is generally applicable but takes much longer time; Absorption probability is fast and can be used in all situation. Thus, we will focus on Method 1 and Method 2 as the approximation method in the next chapter.

CHAPTER 3
Decentralized Sequential Hypothesis Testing

In last chapter, we investigated centralized sequential hypothesis testing problem, however, in many of the modern applications, multiple nodes observe noisy information about the system state. Communicating all the observations to a single node is often too expensive and impractical. Thus, decisions need to be made in a decentralized manner by decision makers that share a common objective. Such decentralized problem are often investigated using team decision theory.

Decentralized sequential hypothesis testing has received considerable attention in the literature [15, 40, 41, 42]. The main emphasis is on identifying qualitative properties of optimal decision strategies. In particular, in identifying belief state (or sufficient statistics) of the data available at the decision maker and in establishing the structure of optimal decision rules, e.g., showing that the threshold-based strategies (similar in spirit to Wald's sequential likelihood ratio test [1]) are optimal. We summarize some of these results below.

A model in which multiple sensors make independent decisions that are coupled through a common loss function was studied in [15, 40]. It was shown that at each time instant, optimal policies for decision makers are described by two thresholds requires solution of two coupled sets of dynamic programming equations. A model in which multiple sensors make individual decisions and a sensor can signal its decision to others was studied in [42], it is proved that at each time, an optimal strategy of the peripheral sensor is characterized by at most four thresholds and an optimal strategy of the coordinating sensor is characterized by two thresholds. A model in which multiple sensors communicate their decisions to a fusion center and the fusion center makes the final stopping decision was studied in [41], where sensors can only use current observation and all past transmissions of all sensors to decide what message to send to the fusion center. The optimal decision rules in this case are also characterized by two thresholds.

3.1 Model

Consider a decentralized version of Wald's sequential hypothesis problem investigated in [15][40]. For ease of exposition, we assume that there are two decision makers, DM^1 and DM^2; the result generalize to multiple decision makers in a natural manner.

At time t, the DM^i, $i \in \{1, 2\}$, takes observations $Y_t^i \in \mathcal{Y}^i$. We assume that given the hypothesis $H = h_k$, $k \in \{1, 2\}$: (i) the observations $\{Y_t^1\}_{t=1}^{\infty}$ and $\{Y_t^2\}_{t=1}^{\infty}$ are conditionally independent; and (ii) the observations $\{Y_t^i\}_{t=1}^{\infty}$ are i.i.d with PMF f_k^i.

Each decision maker has to decide which hypothesis is true based on its own observations; there is no communication between the two decision makers. That is, $\mathrm{DM}^i, i \in \{1, 2\}$, takes a decision $U_t^i \in \{h_0, h_1, \mathsf{C}\}$ at time t according to

$$U_t^i = g_t^i(Y_{1:t}^i),$$

where $Y_{1:t}^i := (Y_1^i \cdots Y_t^i)$.

The decision $U_t^i = h_0$ (or $U_t^i = h_1$) means that DM^i decides to stop and declare h_0 (or h_1) as the true hypothesis and makes no further observations. The decision $U_t^i = \mathsf{C}$ means that DM^i decides to take an additional observation.

Let N^i denote the stopping time when DM^i decides to stop, i.e.,

$$N^i = \min\{t \in \mathbb{Z} > 0 : U_t^i \in \{h_0, h_1\}\}. \tag{3.1}$$

We denote the terminal decision $U_{N^i}^i$ by U^i.

There are two kinds of costs: (i) cost c^i for each observation at DM^i, and (ii) a stopping cost $\ell(U^1, U^2, H)$, which satisfies the following assumptions:

(A1) $\ell(U^1, U^2, H)$ cannot be decomposed as $\ell(U^1, H) + \ell(U^2, H)$, otherwise, the problem decomposes into two independent standard Wald problems.

(A2) For any $m, n \in \{h_0, h_1\}$, $m \neq n$,

$$\begin{aligned} &\ell(m, m, n) \geqslant \ell(n, m, n) \geqslant c^i \geqslant \ell(n, n, n); \\ &\ell(m, m, n) \geqslant \ell(m, n, n) \geqslant c^i \geqslant \ell(n, n, n); \end{aligned} \tag{3.2}$$

All inequalities in equation (3.2) imply that at most one mistake is less costly than at least one mistake.

Let $\mathcal{G}^i$ denote the set of all strategies for DM^i. Then for any choice $(g^1, g^2) \in \mathcal{G}^1 \times \mathcal{G}^2$, the total cost is

$$J(g^1, g^2; p) = \mathbb{E}[c^1 N^1 + c^2 N^2 + \ell(U^1, U^2, H)]. \tag{3.3}$$

We are interested in the following optimization problem:

Problem 1 *Given the observation PMFs f_0^i, f_1^i, the observation cost c^i, and the loss function ℓ, find a strategy (g^1, g^2) that minimizes $J(g^1, g^2; p)$ given by* (3.3).

Note that in Problem 1, we are seeking globally optimal decision strategies. For team problems, a weaker solution concept is that of person-by-person optimality (PBPO), defined below.

Definition 3.1.1 (person-by-person optimality) *A strategy (g^1, g^2) is called person-by-person optimal (PBPO) if*

$$J(g^1, g^2; p) \leq J(g^1, \tilde{g}^2; p), \ \forall \ \tilde{g}^2 \in \mathcal{G}^2,$$

and

$$J(g^1, g^2; p) \leq J(\tilde{g}^1, g^2; p), \ \forall \ \tilde{g}^1 \in \mathcal{G}^1.$$

This gives rise to the following relaxation of Problem 1.

Problem 2 *Given the prior probability p, the observation PMFs f_0^i, f_1^i, the observation cost c^i, and the loss function ℓ, find a strategy (g^1, g^2) that is person-by-person optimal.*

We start by summarizing the results of [15] (which were derived for the finite-horizon setup) and generalizing them to infinite horizon.

3.2 Structure of Optimal Strategy

For any $i \in \{1, 2\}$, let $-i$ denote the other decision maker. Define the belief state

$$\pi_t^i := \mathbb{P}(H = h_0 \mid y_{1:t}^i).$$

And define

$$q^i(y^i_{t+1} \mid \pi^i_t) := \pi^i_t f^i_0(y^i_{t+1}) + (1 - \pi^i_t) f^i_1(y^i_{t+1}), \tag{3.4}$$

$$\phi^i(\pi^i_t, y^i_{t+1}) := \pi^i_t f^i_0(y^i_{t+1}) / q^i(y^i_{t+1} \mid \pi^i_t). \tag{3.5}$$

Using Bayes' rule, the update of the belief state is given by

$$\pi^i_{t+1} = \phi^i(\pi^i_t, y^i_{t+1}). \tag{3.6}$$

It was shown in [15] that $\{\pi^i_t\}_{t=1}^{\infty}$ is an belief state process for DM^i. In particular:

Proposition 3.2.1 ([15]) *For any $i \in \{1, 2\}$ and any choice of strategy g^{-i} of DM^{-i}, there is no loss of optimality for DM^i to restrict attention to strategies of the form*

$$U^i_t = g^i_t(\pi^i_t). \tag{3.7}$$

To characterize the structure of the optimal strategy, we define the following.

Definition 3.2.1 (Threshold based strategy) *A strategy of the form* (3.7) *is called threshold based if there exists thresholds $\alpha^i_t, \beta^i_t \in [0, 1]$, $\alpha^i_t \leq \beta^i_t$, such that for any $\pi^i \in [0, 1]$,*

$$g^i_t(\pi^i) = \begin{cases} h_1 & \textit{if } \pi^i < \alpha^i_t, \\ \mathsf{C} & \textit{if } \alpha^i_t \leq \pi^i \leq \beta^i_t, \\ h_0 & \textit{if } \pi^i > \beta^i_t. \end{cases}$$

It was shown in [15] that threshold-based strategies are team optimal. In particular:

Proposition 3.2.2 ([15, Theorem 3.1]) *For any choice of $g^{-i} \in \mathcal{G}^{-i}$ of DM^{-i}, there is no loss of optimality in restricting attention to threshold based strategies at DM^i.*

Definition 3.2.2 (Time invariant strategy) *A strategy $g^i = (g_1^i, g_2^i, \dots)$ is called time invariant if for any $\pi^i \in [0,1]$, $g_t^i(\pi^i)$ does not depend on t.*

For infinite-horizon problems, for a single decision maker, time-invariant strategies are optimal. That is not always the case for multiple decision makers. It was shown in [15] that threshold-based time-invariant strategies are person-by-person optimal.

As similarly defined in Section 2.6, for any $i \in \{1,2\}$, $k \in \{0,1\}$, $u^i \in \{h_0, h_1\}$ and $g^i \in \mathcal{G}^i$, define

$$\xi_k^i(u^i, g^i) = P(U^i = u^i \mid H = h_k; g^i). \tag{3.8}$$

Proposition 3.2.3 *For any $i \in \{1,2\}$ and any* time-invariant *and* threshold-based *strategy $g^{-i} \in \mathcal{G}^{-i}$, the best response strategy g^i is a time-invariant threshold-based strategy that is given by the solution of the following DP: for any $\pi^i \in [0,1]$*

$$V^i(\pi^i) = \min\{W_0^i(\pi^i, g^{-i}), W_1^i(\pi^i, g^{-i}), W_{\mathsf{C}}^i(\pi^i, g^{-i})\}, \tag{3.9}$$

where

$$W_k^1(\pi^1, g^2) = \sum_{u^2 \in \{h_0, h_1\}} [\xi_0^2(u^2, g^2)\pi^1 \ell(h_k, u^2, h_0) + \xi_1^2(u^2, g^2)(1-\pi^1)\ell(h_k, u^2, h_1)], \tag{3.10}$$

W_k^2 is defined similarly, and

$$W_{\mathsf{C}}^i(\pi^i, g^{-i}) = c^i + [\mathscr{B}^i V^i](\pi^i), \tag{3.11}$$

where $\mathscr{B}^i$ is the Bellman operator given by

$$[\mathscr{B}^i V^i](\pi^i) = \sum_{y^i} V^i(\phi(\pi^i, y^i)) q(y^i \mid \pi^i),$$

$q(y^i \mid \pi^i)$ and $\phi(pi^i, y^i)$ are given by (3.4) *and* (3.5)*.*

3.3 Orthogonal search vs direct search

For ease of notation, denote a threshold-based strategy g^i by the tuple $\langle \alpha^i, \beta^i \rangle$. Note that Proposition 3.2.3 gives two coupled dynamic programs, which we write succinctly as

$$\langle \alpha^1, \beta^1 \rangle = \mathcal{D}^1(\langle \alpha^2, \beta^2 \rangle) \quad \text{and} \quad \langle \alpha^2, \beta^2 \rangle = \mathcal{D}^2(\langle \alpha^1, \beta^1 \rangle). \tag{3.12}$$

A solution of these coupled dynamic program determines a PBPO solution for Problem 2.

As mentioned in Section 2.4, under assumptions that f^i_k are Gaussian distributions [15] and

$$c^i \ll \min\{\ell(h_0, h_1, h_0), \ell(h_1, h_0, h_1)\},$$

the stopping time $N^i \gg 1$, and one can use the asymptotic expressions of Type I and Type II errors [8, 32] that were used in [15], to approximate this discrete-time Markov process by a continuous-time Markov process, also see Section 2.4 for details.

In this section, we propose two methods to compute PBPO threshold strategies. In the first method, which we call *orthogonal search*, we iteratively solve the coupled dynamic programs (3.12). To solve the dynamic program of Proposition 3.2.3, we need to compute ξ^i_k; which is the probability that a discrete-time Markov process crosses a threshold. The asymptotic expression of [8, 32] that were used in [15], approximate this discrete-time Markov process by a continuous-time Markov process. In Section 2.6, we propose an alternative method that approximate the discrete-time Markov process by a discrete-time *finite-state* Markov chain to approximate ξ_k^i.

In the second method, which we call *direct search*, we approximate the expected cost $J(\langle \alpha^1, \beta^1 \rangle, \langle \alpha^2, \beta^2 \rangle)$ of a threshold based strategy.

3.3.1 Orthogonal search

The coupled dynamic program of (3.12) may be solved using orthogonal search procedure described below:

1) Start by an arbitrary threshold-based strategy $[\langle \alpha^1_{(1)}, \beta^1_{(1)} \rangle, \langle \alpha^2_{(1)}, \beta^2_{(1)} \rangle]$.
2) Construct a sequence of strategies as follows:

   For even n:

$$\langle \alpha^1_{(n)}, \beta^1_{(n)} \rangle = \mathcal{D}^1(\langle \alpha^2_{(n-1)}, \beta^2_{(n-1)} \rangle),$$

   and

$$\langle \alpha^2_{(n)}, \beta^2_{(n)} \rangle = \langle \alpha^2_{(n-1)}, \beta^2_{(n-1)} \rangle.$$

   For odd n:

$$\langle \alpha^1_{(n)}, \beta^1_{(n)} \rangle = \langle \alpha^1_{(n-1)}, \beta^1_{(n-1)} \rangle,$$

   and

$$\langle \alpha^2_{(n)}, \beta^2_{(n)} \rangle = \mathcal{D}^2(\langle \alpha^1_{(n-1)}, \beta^1_{(n-1)} \rangle).$$

Theorem 3.3.1 *The orthogonal search procedure described above converges to a time-invariant threshold-based strategy* (g^1, g^2) *that is person-by-person optimal.*

Proof 3.3.1 *Let $(g^i_{(n)}, g^2_{(n)})$ denote the strategy at step n. By construction,*

$$J(g^1_{(n)}, g^2_{(n)}) \leq J(g^1_{(n-1)}, g^2_{(n-1)}).$$

Thus, the sequence $\{J(g^1_{(n)}, g^2_{(n)})\}$ is a decreasing sequence lower bounded by 0. Hence, a limit exists and the limiting strategy is PBPO.

There are two main steps in each iteration of orthogonal search. First, at step n, we need to compute ξ^i_k for a threshold-based strategy $\langle \alpha^i_{(n)}, \beta^i_{(n)} \rangle$. Second, the dynamic program at step n is a POMDP. So, we either need to discretize the state-space or use the point-based methods [43, 44].

A. Discretization

For any $m \in \mathbb{N}$, define $S_m = \{0, \frac{1}{m}, \frac{2}{m}, \dots, 1\}$. For any $i \in \{0, 1\}$, we consider three approximations that makes different assumptions on probability distribution of Y^i. We denote the corresponding transition probabilities by P^i_0, P^i_1, and P^i_*. For $P^i_k, k \in \{0, 1\}$, we assume that $Y^i \sim f^i_k$, for P^i_*, we assume that $Y^i \sim q^i(\cdot \mid \pi^i_t)$, which is given by (3.4). The discretization procedure follows similar manner as in Section 2.5.1. We only use first-order hold discretization in here.

Note that the transition probabilities $P^i_k, k \in \{0, 1\}$, approximate the evolution of the $\{\pi^i_t\}^\infty_{t=1}$ process when hypothesis $H = h_k$ is true. We use these to approximate probabilities ξ^i_k. On the other hand, the transition probability P^i_* approximates the uncontrolled evolution of $\{\pi^i_t\}^\infty_{t=1}$. This will be used to approximately solve the dynamic program of Proposition 3.2.3.

B. Approximately computing ξ^i_k

Fix a decision maker i, $i \in \{1, 2\}$, and the hypothesis $H = h_k, k \in \{0, 1\}$. Given an arbitrary strategy $g^i = \langle \alpha^i, \beta^i, \rangle$ such that $\alpha^i, \beta^i \in \S_m$. We can specify set A_0^i, $A_1^i \subset S_m$ as :

$$A_0^i = \{\beta^i, \beta^i + \frac{1}{m}, \dots, 1\},$$

and

$$A_1^i = \{0, \frac{1}{m}, \dots, \alpha^i\}.$$

Consider the Markov chain with transition probability P_k^i and absorption sets A_0^i, A_1^i. Let $\hat{P}_k^i$ be the transition matrix of re-ordered Markov chain that the transient states comes first:

$$\hat{P}_k^i = \begin{pmatrix} \hat{Q}_k^i & \hat{R}_k^i \\ 0 & I \end{pmatrix}.$$

Denote $B_k^i = (I - Q_k^i)^{-1} R_k^i$. From standard result in Markov chain, we know that for any transient state s and $b \in \{0, 1\}$, $[B_k^i]_{sb}$ is the probability that the Markov process starting in state s is absorbed in the set A_b. Thus,

$$\xi_k^i(h_b, \langle \alpha^i, \beta^i \rangle) \approx [B_k^i]_{s^* b}, \; b \in \{0, 1\}. \tag{3.13}$$

where s^* denotes the index of s in transient set $\{\alpha + \frac{1}{m}, \dots, \beta - \frac{1}{m}\}$.

C. Solving dynamic program

Using the procedure of the previous section, we can approximate $\xi_k^i(\pi^i, g^{-i})$, and therefore approximately compute $W_k^i(\pi^i, g^{-i})$. To approximately solve the dynamic program of Proposition 3.2.3, we also need to approximate the Bellman operator $\mathscr{B}^i$.

Define the approximate Bellman operator using the first-order transition matrix P_*^i as follows:

$$[\hat{\mathscr{B}}^i V^i](s) \approx c^i + \sum_{s' \in S_m} [P_*^i]_{ss'} V(s'). \tag{3.14}$$

Then $\hat{\mathscr{B}}^i$ corresponds to the discretization of $\mathscr{B}^i$ on S_m and performing linear interpolation on points outside S_m. It is used to approximately compute $W^i_{\mathsf{C}}(\pi^i, g^{-i})$.

Combining all these, we get an approximate procedure to solve the dynamic program of Proposition 3.2.3. This will find a PBPO strategy using orthogonal search.

Generally speaking, the procedure of orthogonal search consists of optimizing the decision rule of one sensor at a time while keeping the decision rule of the remaining sensors fixed. The overall performance of the decentralized team is guaranteed to improve with every iteration of the orthogonal search. However, system design policies resulting from this procedure represents necessary but not, in general, sufficient conditions to determine the globally optimum solution. The Gauss-Seidel cyclic coordinate descent algorithm has been proposed in literatures [45, 46] to obtain the PBPO solution satisfying the necessary conditions of optimality in an iterative manner.

Among other optimization methods, the alternating direction method of multipliers (ADMM) is well suited to distributed optimization. The method was developed in the 1970s, and is equivalent or closely related to many other algorithms, such as dual decomposition, the method of multipliers and others. The global convergence of ADMM are shown for convex functions [47]. For non-convex optimization, the convergence should be verified case by case for different applications [48, 49]. For the sequential hypothesis testing problem discussed in our work, it may not be convex in the thresholds.

3.3.2 Direct search

Finding person-by-person optimal decision rules is relatively easy, however, the thresholds satisfying the above optimal stopping rule guarantee only member by member optimality. There is no criterion for deciding whether a person-by-person optimal decision rule is also team optimal. Next, we follow the same idea as in Section 2.6.3 and generalize it to decentralized case. In this method we approximate the expected cost $J(g^1, g^2; p)$ (given by (3.3)) of a threshold based strategy. As in the *orthogonal search*, we approximate this by approximating the evolution of π^i_t.

Given an arbitrary strategy (g^1, g^2), for $i \in \{1, 2\}$, $k \in \{0, 1\}$, and for an *a prior* probability p, define:

$$\theta^i_k(p, g^i) = \mathbb{E}[N^i \mid H = h_k; g^i, p].$$

We also add the dependence on p in ξ^i_k, that is, define: for any $u^i \in \{h_0, h_1\}$,

$$\xi^i_k(u^i, g^i, p) = \mathbb{P}(U^i = u^i \mid H = h^k; g^i, p).$$

Write $J(g^1, g^2; p)$ after adding dependency on p, for each p as:

$$\begin{aligned} J(g^1, g^2; p) =& p \cdot [c^1 \cdot \theta^1_0(p, g^1) + c^2 \cdot \theta^2_0(p, g^2)] + (1 - p) \cdot [c^1 \cdot \theta^1_1(p, g^1) + c^2 \cdot \theta^2_1(p, g^2)] \\ &+ \sum_{u^1, u^2 \in \{h^0, h^1\}^2} p \cdot \xi^1_0(u^1, g^1, p) \cdot \xi^2_0(u^2, g^2, p) \cdot \ell(u^1, u^2, h_0) \\ &+ \sum_{u^1, u^2 \in \{h^0, h^1\}^2} (1 - p) \cdot \xi^1_1(u^1, g^1, p) \cdot \xi^2_1(u^2, g^2, p) \cdot \ell(u^1, u^2, h_1). \end{aligned} \tag{3.15}$$

For an arbitrary strategy, ξ_k^i can be approximated by the absorption probabilities of a Markov chain with transition matrix P_k^i as shown in (3.13).

To approximate θ_k^i, define sets A_0^i, A_1^i and $\hat{P}_k^i$ as in Part B of Section 3.3.1.B. Define $T_k^i = (I - Q_k^i)^{-1}\mathbf{1}$, where $\mathbf{1}$ is a column vector with all entries as 1. From the result in Section 2.6.1, for any state $s \in S_m \setminus (A_0^i \cup A_1^i)$, $[T_k^i]_s$ is the expected stopping time that the Markov chain starting in state s is absorbed in $(A_0^i \cup A_1^i)$. Thus,

$$\theta_k^i(p, \langle \alpha^i, \beta^i \rangle) \approx [T_k^i]_{s^*}. \tag{3.16}$$

where s^* denotes the index of s in transient set $S_m \setminus (A_0^i \cup A_1^i)$.

By substitute them in (3.15), we can approximately compute $J(p, \langle \alpha^1, \beta^1 \rangle, \langle \alpha^2, \beta^2 \rangle)$ for any $p, \alpha^1, \beta^1, \alpha^2, \beta^2 \in S_m$. To reduce the dependence of the numerical results on the choice of p, we pick multiple values of p in a finite set $\mathcal{P} \subset [0, 1]$ and use

$$\hat{J}(\alpha^1, \beta^1, \alpha^2, \beta^2) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} J(p, \langle \alpha^1, \beta^1 \rangle, \langle \alpha^2, \beta^2 \rangle).$$

as the objective function for the non-convex derivative-free optimization function introduced in Section 2.6.3, we are able to achieve the optimal strategy.

3.4 Case Study

Both the approaches presented in this paper only guarantee local optimality. In this section, we compare their performance on a benchmark system in which

$\mathcal{Y}^1 = \mathcal{Y}^2 = \{0, 1\}$ and the loss function is of the form

$$\ell(u^1, u^2, h) = \begin{cases} 0, & \text{if } u^1 = u^2 = h, \\ 1, & \text{if } u^1 \neq u^2, \\ L, & \text{if } u^1 = u^2 \neq h. \end{cases} \tag{3.17}$$

For both methods, we use $m = 1000$ and in direct search, we use $\mathcal{P} = \mathcal{S}_m$.

Note that the choice of parameters (c^1, c^2, L) and observation distributions $(f_0^1, f_1^1, f_0^2, f_1^2)$ completely specifies the model. We first consider an example where we pick specific values for the parameters and the distributions. Then we compare the performance of the two methods when all the parameters are chosen at random.

3.4.1 Coupled Loss Case

Let $c^1 = c^2 = 0.05$, $L = 2.5$ and

$$f_0^1 = \begin{bmatrix} 0.25 & 0.75 \end{bmatrix}, \qquad f_0^2 = \begin{bmatrix} 0.80 & 0.20 \end{bmatrix},$$
$$f_1^1 = \begin{bmatrix} 0.60 & 0.40 \end{bmatrix}, \qquad f_1^2 = \begin{bmatrix} 0.30 & 0.70 \end{bmatrix}.$$

The result of orthogonal search and direct search are shown in Table 3–1. Both approaches converge to a locally optimal solution. For this particular example, direct search converges to a slightly better solution that orthogonal search. Although orthogonal search converges in significantly fewer number of iterations, each iteration of orthogonal search involves solving an infinite horizon dynamic program (which was solved using value iteration with convergence threshold 10^{-3}). The running time of both the algorithms is reported, but it should be noted that we did not attempt to optimize the Matlab implementation of the algorithms.

Table 3–1: Comparison of orthogonal search and direct search for the specific parameters presented in Section 3.4.1.

	$g^1 = \langle \alpha^*_{(1)}, \beta^*_{(1)} \rangle$	$g^2 = \langle \alpha^*_{(2)}, \beta^*_{(2)} \rangle$	$\hat{J}(g^1, g^2)$	iters.	runtime
OS[1]	$\langle 0.326, 0.73 \rangle$	$\langle 0.07, 0.931 \rangle$	0.455	5	1.45s
DS[1]	$\langle 0.287, 0.726 \rangle$	$\langle 0.14, 0.863 \rangle$	0.436	45	6.05s

[1] OS stands for othogonal search and DS stands for direct search.

3.4.2 Decomposable Case

In the previous parts, we mentioned that both orthogonal search and direct search converge to a local optimal solution. In general, the globally optimal solution is not known. In the special case when $L = 2$, the total stopping cost $\ell(U^1, U^2, H)$ is not coupled because it can be written as $\ell(U^1, H) + \ell(U^2, H)$. Hence the decentralized sequential hypothesis testing problem decomposes into two independent centralized sequential hypothesis testing problem. In this case, the solution $g^i = \langle \alpha^i, \beta^i \rangle$ can be obtained by separately solving the two centralized sequential hypothesis testing problems. We can use value iteration to find the optimal threshold-based policy for two centralized sequential hypothesis testing problems and compare the result with orthogonal search and direct search. We refer to this solution as centralized solution.

Let $L = 2$ and set $c^1, c^2, f_0^1, f_1^1, f_0^2, f_1^2$ to be the same as the values in Section 3.4.1. The result of orthogonal search, direct search, and the centralized solution is shown in Table 3–2. For this particular example, direct search gives the same solution as the centralized solution while orthogonal search converges to a solution with poorer performance.

Table 3–2: Comparison of decomposable orthogonal search and global search with centralized solution.

	$g^1 = \langle \alpha^1, \beta^1 \rangle$	$g^2 = \langle \alpha^2, \beta^2 \rangle$	$\hat{J}(g^1, g^2)$
OS	$\langle 0.318, 0.686 \rangle$	$\langle 0.089, 0.913 \rangle$	0.428
DS	$\langle 0.3053, 0.7055 \rangle$	$\langle 0.1845, 0.8218 \rangle$	0.406
CS [1]	$\langle 0.305, 0.705 \rangle$	$\langle 0.184, 0.822 \rangle$	0.406

[1] CS stands for centralized solution.

3.4.3 General Performance for Coupled Loss Cases

To compare the performance of the two methods, we test both of them over 500 randomly generated instances of the parameters (c^1, c^2, L) and $(f_0^1, f_1^1, f_0^2, f_1^2)$. Specifically, we use $c^1, c^2 \sim$ unif[0, 0.05], $L \sim$ unif[1, 4]. We pick f_k^i by picking a random number $\delta_k^i \sim$ unif[0,1] and setting $f_k^i = [\delta_k^i, 1 - \delta_k^i]$.

Let J_{OS} and J_{DS} denote the performance of the solution obtained by orthogonal search and direct search. Define $\Delta J_{OS} = (J_{OS} - J_{DS})/J_{OS}$ and $\Delta J_{DS} = (J_{DS} - J_{OS})/J_{DS}$ as the relative difference between the performance of orthogonal search and direct search. The histograms of ΔJ_{OS} and ΔJ_{DS} are shown in Figure 3–1. Note that for 488 of the 500 cases, $J_{OS} > J_{DS} + 10^{-4}$; therefore, for most scenarios, direct search performs better than orthogonal search.

3.4.4 General Performance for Decomposable Cases

It is also interesting to look at the general performance of two approaches when the globally optimal solution is known. In this experiment, we set $L = 2, c^1 = c^2 = 0.05$ and pick $f_0^1, f_1^1, f_0^2, f_1^2$ randomly as specified in Section 3.4.3. Let J_{OS}, J_{DS} denote the performance of the solution obtained by orthogonal search and direct search. Let J^* denote the centralized solution. Define the relative errors $E_{OS} = (J_{OS} - J^*)/J^*$

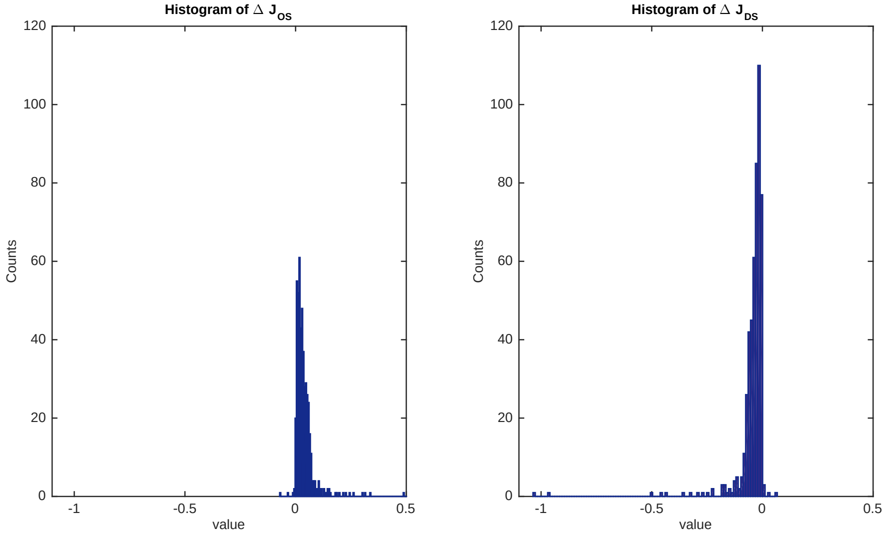

Figure 3–1: Histograms of ΔJ_{OS} and ΔJ_{DS}.

and $E_{DS} = (J_{DS} - J^*)/J^*$. The histograms of E_{OS} and E_{DS} are shown in Figure 3–2. From the plot, we can see that the errors of both approaches are within tolerable range and that the performance of direct search is better than the performance of orthogonal search in most cases.

The nature of the solution remains the same when $c^i \ll L$. For example, for $L = 2, c^1 = c^2 = 5 \times 10^{-4}$, and f_h^i chosen randomly as in Section 3.4.3, the histograms of E_{OS} and E_{DS} are shown in Figure 3–3.

3.4.5 Computational Complexity

The proposed search algorithms consist of two parts: computing the transition matrix (Algorithm 1) and using the transition matrix to compute the thresholds. The complexity of the first part is linear in $|\mathcal{Y}|$, but the complexity of the second part does not depend on $|\mathcal{Y}|$. Therefore, one would not expect a significant increase

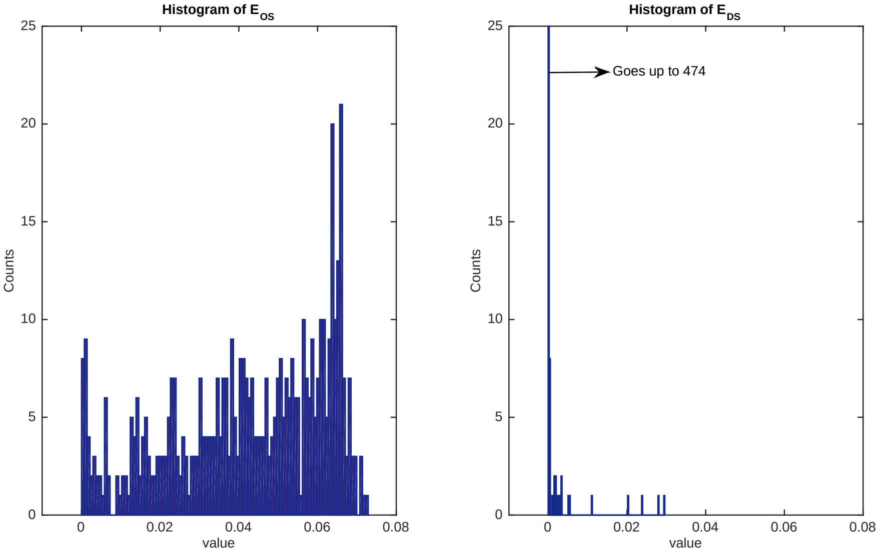

Figure 3–2: Histograms of E_{OS} and E_{DS}.

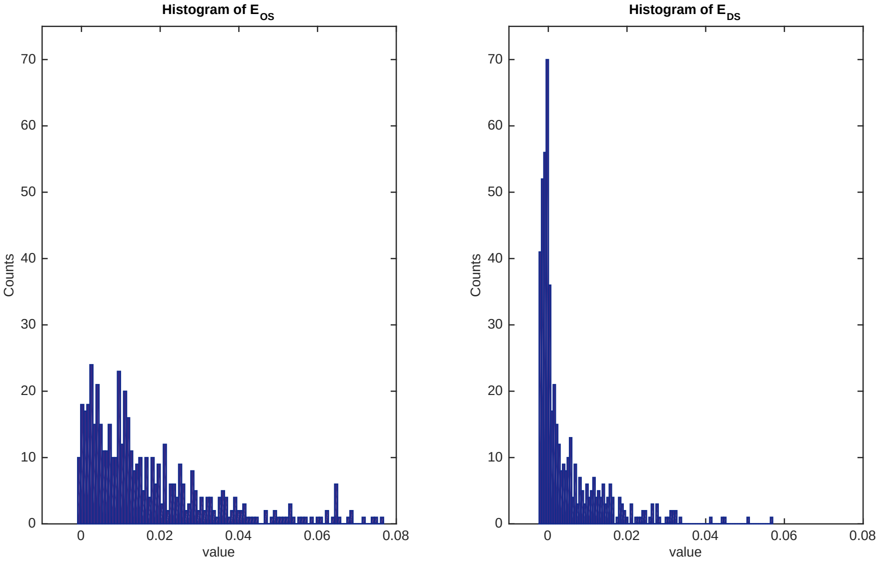

Figure 3–3: Histograms of E_{OS} and E_{DS} when $c^i \ll L$.

in the run-time with an increase in the size of the observations $|\mathcal{Y}|$. This is confirmed numerically as well.

We consider four cases: $|\mathcal{Y}| = 2$, $|\mathcal{Y}| = 4$, $|\mathcal{Y}| = 8$, $|\mathcal{Y}| = 16$. For each case, we run 100 simulations where c^1, c^2, L are chosen randomly as described in Section 3.4.3. To choose f^i_h, for $|\mathcal{Y}| = m$, we pick m random numbers $(\delta^i_{k0}, \ldots, \delta^i_{km}) \sim$ unif[0,1] and set $f^i_h = [\delta^i_{k0}, \ldots, \delta^i_{km}]/S^i_k$, where $S^i_k = \sum_{j=1}^{m} \delta^i_{kj}$. The average run-time for orthogonal search and direct search is shown in Table 3–3. As expected, with increasing $|\mathcal{Y}|$, the run-time does not increase accordingly, it remains at a steady state. This suggests that the proposed approaches should work well even when $\mathcal{Y}$ is continuous valued.

Table 3–3: Running time with respect to number of observations.

	$\|\mathcal{Y}\| = 2$	$\|\mathcal{Y}\| = 4$	$\|\mathcal{Y}\| = 8$	$\|\mathcal{Y}\| = 16$
OS average	5.497s	3.952s	6.692s	5.266s
DS average	3.560s	2.558s	2.380s	2.574s

3.5 Discussion

In this chapter, we delved into two methods to approximately compute the optimal threshold-based strategy in decentralized sequential hypothesis testing. Both methods are based on discretization of the continuous-valued information state process by a finite-valued Markov chain. The orthogonal search method computes PBPO strategies while the direct search methods attempts to compute globally optimal strategies. Direct search involves solving a non-convex optimization problem, so in practice, it will also converge to a local optimal. In our numerical investigation of the two algorithms, direct search performs better than orthogonal search; sometimes, significantly better.

CHAPTER 4
Conclusion

4.1 Summary

In this paper, we focus on computing optimal thresholds in sequential hypothesis testing problem. We implemented existing algorithms and proposed new algorithms. The results of different methods are presented and discussed.

The theory of sequential hypothesis testing problem has been well developed and researched since it's first formulation in 1947. Until now, a lot is known about the property and structure of the optimal control strategy, but little is known about computation methods. The contribution of this work is to provide a cookbook for someone who wants to solve a sequential hypothesis testing problem.

In Chapter 2, we focus on single decision maker sequential hypothesis testing problem, also some of these discussions is further developed in the decentralized scenario. Firstly, we introduced the model and problem formulation. Secondly, an exact solution using α-vectors is presented, we showed that the complexity grows exponentially with dimension, that's why approximation methods are considered. Thirdly, we introduce one approximation computation by discretizing the continuous state MDP. Based on the discretized MDP, three methods are discussed. For each method, the basic principle and the detailed computation procedure is provided. In addition, we also propose a non-convex derivative-free optimization method as an alternative of finding optimal strategy. At last, we conclude the centralized sequential hypothesis testing problem with a case study. It is shown that the discretized model approximates the value function well and among all approximation methods, one method using the absorption probabilities of absorbing Markov chain has better performance in general.

In Chapter 3, we generalize the problem from centralized sequential hypothesis testing to decentralized case, we focus on the approximation method shown with good performance in Chapter 2. We showed how to apply this approximation method in the process of finding person-by-person optimal strategies as well as directly searching global optimal strategy, and compared the result from orthogonal search and direct search in a case study. It is shown that, direct search generally performs better than orthogonal search.

Overall, this work summarizes the topics on numerically solving sequential hypothesis testing problem and also opens up further work on decentralized sequential hypothesis testing.

4.2 Future Work

For future work, one aspect is to generalize the problem. In this work, we only consider binary hypothesis testing problem for two decision makers. If the number of hypothesis increases, it will become a multi-dimensional hypothesis testing problem. The discretization technique of multi-dimensional model is provided in the Appendix. It would be better to provide another case study on how the proposed approaches work in multi-dimensional scenarios. It is interesting to look at how the computational complexity of different methods scales as the number of decision makers increases. In our case studies, we used simple examples to illustrate the problem and show the result. Next step, we could apply the theory into a real decision problem (like the machine replacement problem discussed in [23].) and demonstrate the results. Another direction is to generalize the approximation methods developed in this work to more general decentralized sequential hypothesis models of [41, 42].

Another aspect is to improve the efficiency of the algorithms. One example would be the Monte Carlo approximation, to make sure the simulation result is a good approximation of true value, we should keep the number of sampling a big number. However, when the number of sampling increases, the running time also increase, even if we use the high performance computer, since the current Matlab code is not parallel, the advantage of high performance computation is not used. Since in each sampling, the computation is independent, we could parallelize the code and use Matlab parallel toolbox to speed the computation. Another example is the non-convex optimization function *fminsearch* that we use. There are many other (more efficient and accurate) algorithms, we could explore other possible methods and make improvements in the result. If one is interested to develop a solver targeting the sequential hypothesis testing problem discussed in this thesis, it would be an non-convex derivative free optimization algorithm that finds the optimal parameter for a function without knowing the mathematical expression of the function and the function may not be convex.

Also, we could look at other mechanisms to solve the sequential hypothesis testing problem, for example reinforcement learning is also a well known method for solving POMDP problems. Many researchers have studied this and published their results [50, 51, 52]. We are interested to look at this further into detail and see how to use reinforcement learning in our work and how the result compares with the results of our current methods.

Lastly, the approximation bounds of discretizing the continuous belief space is not discussed in this thesis. We could follow [53] and try to give a formal proof about the lower and upper bound of grid-based discretization.

Appendix A: Multi-dimensional Discretization

In this appendix, we will introduce a method to discretization in more than two dimensions. As we all know, in a two dimension case, if we want to discretize a line we equally divide the line into several pieces by setting many equally placed points on the line; if we want to discretize a plane, we discretize each side of the plane and line the points. However, for dimensions more than two, a method named The Freudenthal Triangulation is one way to discretize. The results presented in this section closely follow the presentation in [23].

The Freudenthal triangulation is named after the German mathematician who introduced it in 1942. It is shown in Eaves [54] that a triangulation of R^n can be constructed with the n-dimensional integer vectors as vertices. Given any $x \in R^n$, the particular simplex that contains x can be generated as follows. Let the base v be the largest integer vector less than or equal to x. Define the direction form the base to x as $d = x - v$, let p denote a permutation of the integers $\{1, 2, \cdots, n\}$ that orders the components of d in descending manner, so that $d_{p1} \geq d_{p2} \geq \cdots \geq d_{pn}$. Let e_j denote the jth unit vector in R^n, the vertices $\{v^i\}$ of a simplex containing x can be constructed as

$$
\begin{aligned}
&v^1 = v \\
&v^2 = v^1 + e_{p1} \\
&v^3 = v^2 + e_{p2} \\
&\vdots \\
&v^{k+1} = v^k + e_{pk} \quad k \leq n.
\end{aligned}
$$

If we have a function f that is evaluated at each possible vertex, then we can construct a continuous, piece-wise linear function on all of R^n by defining

$$f(x) = \sum_{i=1}^{n+1} \lambda_i f(v^i),$$

where v^i are the vertices of a simplex containing x, and the λ_i are the barycentric coordinates of x with respect to those vertices. The solving of λ_i can be deducted by back substitution.

$$
\begin{aligned}
&\lambda_{n+1} = d_{pn} \\
&\lambda_n = d_{p(n-1)} - \lambda_{n+1} = d_{p(n-1)} - d_{pn} \\
&\lambda_{n-1} = d_{p(n-2)} - \lambda_n - \lambda_{n+1} = d_{p(n-2)} - d_{p(n-1)} \\
&\vdots \\
&\lambda_2 = d_{p1} - d_{p2} \\
&\lambda_1 = 1 - \sum_{i=2}^{n+1} \lambda_i.
\end{aligned}
$$

If we want to triangulate $\pi(S)$. Let M be any positive integer, and consider the subset of vertices in Freudenthal triangulation of R^n

$$G' = \{q \in I_+^n \mid M = q_1 \geq q_2 \geq q_3 \geq \cdots \geq q_n \geq 0\}.$$

Define the $n \times n$ non-singular matrix

$$B = \frac{1}{M} = \begin{bmatrix} 1 & \text{-1} & 0 & 0 & & 0 \\ 0 & 1 & \text{-1} & 0 & & 0 \\ 0 & 0 & 1 & \text{-1} & & 0 \\ 0 & 0 & & & 1 & \text{-1} \\ 0 & 0 & 0 & & & 1 \end{bmatrix}.$$

Then for $q \in G'$, $Bq = (1/M)(M - q_2, q_2 - q_3, q_3 - q_4, \cdots, q_{n-1} - q_n, q_n)$, i.e.,

$$Bq \in G = \{\pi = \frac{1}{M}m \mid m \in I_+^n, \sum_{i=1}^{n} m_i = M\}.$$

The set G is the set of regular grid points in $\pi(S)$. Since any 1:1 linear mapping from R^n onto itself will map one triangulation into another[41], we can use Freudenthal triangulation in R^n in induce a triangulation of $\pi(S)$ via the mapping B. For example, consider the case with $n = 3$ and $M = 2$. Then

$$\pi(S) = \{\pi \in R^3 : \pi_i \geq 0 \ for \ all \ i, \sum_{i=1}^{3} = \pi_i = 1\}.$$

The grid points q in G' are (2, 0, 0), (2, 2, 0), (2, 2, 2), (2, 1, 0), (2, 1, 1), (2, 2, 1). B will map these into the grid points in $G \subset \pi(S)$, in accordance, (1, 0, 0), (0, 1, 0),(0, 0, 1),(0.5, 0.5, 0), (0.5, 0, 0.5), (0, 0.5, 0.5). Algebraically, B maps one (n - 1)-dimensional affine set, $M \times R^{n-1}$, into another, the affine hull of $\pi(S)$ and establishes an isomorphic relationship between points in G and points in G'.

Say we have a function V defined on the regular grid, G in $\pi(S)$. V has been evaluated for $\{\pi : \pi \in G\} = \{Bv : v \in G'\}$, each v in G' is a vertex in the Freudenthal triangulation of R^n. Therefore, given any $\pi \in \pi(S)$, we can calculate $x = B^{-1}\pi$, and find the vertices $\{v^i\}$ and the barycentric coordinated $\{\lambda^i\}$ from the Freudenthal triangulation on R^n. We have

$$B^{-1}\pi = x = \sum_{i=1}^{n+1} \lambda_i v_i,$$

$$\pi = \sum_{i=1}^{n+1} \lambda_i B v_i,$$

$$V(\pi) = \sum_{i=1}^{n+1} \lambda_i V(Bv_i).$$

As it can been seen above, the only work that need to be done in $\pi(S)$ is setting up and evaluating the regular grid points G. Other things are done in the Freudenthal triangulation of $\pi(S)$ under B.

References

[1] A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 1998.

[2] P. Abbeel, "Lecture on discretization of continuous state mdp." `http://www.cs.berkeley.edu/~pabbeel/cs287-fa12/slides/discretization.pdf`, 2011.

[3] E. J. Sondik, "The optimal control of partially observable markov processes over the infinite horizon: Discounted costs," *Operations Research*, vol. 26, no. 2, pp. 282–304, 1978.

[4] R. Bellman, "E. 1957. dynamic programming," *Princeton UniversityPress. BellmanDynamic programming1957*, 1957.

[5] R. A. Howard, "Dynamic programming and markov process.," 1960.

[6] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2014.

[7] D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 1. Athena Scientific Belmont, Massachusetts, 1996.

[8] A. Wald, *Sequential analysis.* Courier Corporation, 1973.

[9] S. Thiébaux and M.-O. Cordier, "Supply restoration in power distribution systems—a benchmark for planning under uncertainty," in *Sixth European Conference on Planning*, 2014.

[10] R. Washington, "Uncertainty and real-time therapy planning: incremental markov-model approaches," in *AAAI Spring Symposium on Artificial Intelligence in Medicine*, Citeseer, 1996.

[11] R. S. Kaplan, "Optimal investigation strategies with imperfect information," *Journal of Accounting Research*, pp. 32–43, 1969.

[12] R. D. Smallwood, "The analysis of economic teaching strategies for a simple learning model," *Journal of Mathematical Psychology*, vol. 8, no. 2, pp. 285–301, 1971.

[13] S. M. Pollock, "A simple model of search for a moving target," *Operations Research*, vol. 18, no. 5, pp. 883–903, 1970.

[14] R. H. Crites, *Large-scale dynamic optimization using teams of reinforcement learning agents.* PhD thesis, University of Massachusetts, 1996.

[15] D. Teneketzis and Y.-C. Ho, "The decentralized wald problem," *Information and Computation*, vol. 73, no. 1, pp. 23–44, 1987.

[16] A. Mahajan, "Lecture notes on pomdp." `http://www.cim.mcgill.ca/~decinfo/courses/ecse506/winter-2014/notes/POMDP-structure.pdf`, November 2013.

[17] K. J. Arrow, D. Blackwell, and M. A. Girshick, "Bayes and minimax solutions of sequential decision problems," *Econometrica, Journal of the Econometric Society*, pp. 213–244, 1949.

[18] B. K. Ghosh and B. K. Ghosh, *Sequential tests of statistical hypotheses.* Addison-Wesley Reading, Mass, 1970.

[19] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable markov processes over a finite horizon," *Operations Research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[20] E. J. Sondik, "The optimal control of partially observable markov processes.," tech. rep., DTIC Document, 1971.

[21] G. E. Monahan, "State of the art—a survey of partially observable markov decision processes: Theory, models, and algorithms," *Management Science*, vol. 28, no. 1, pp. 1–16, 1982.

[22] C. C. White III, "A survey of solution techniques for the partially observed markov decision process," *Annals of Operations Research*, vol. 32, no. 1, pp. 215–230, 1991.

[23] W. S. Lovejoy, "Computationally feasible bounds for partially observed markov decision processes," *Operations research*, vol. 39, no. 1, pp. 162–175, 1991.

[24] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Efficient dynamic-programming updates in partially observable markov decision processes," 1995.

[25] H.-T. Cheng, *Algorithms for Partially Observable Markov Decision Processes.* PhD thesis, University of British Columbia, Vancouver, BC, 1988.

[26] N. L. Zhang and W. Liu, "Planning in stochastic domains: Problem characteristics and approximation," tech. rep., Citeseer, 1996.

[27] N. L. Zhang and W. Zhang, "Speeding up the convergence of value iteration in partially observable markov decision processes," *J. Artif. Intell. Res.(JAIR)*, vol. 14, pp. 29–51, 2001.

[28] E. J. Sondik, "The optimal control of partially observable markov processes over the infinite horizon: Discounted costs," *Operations Research*, vol. 26, no. 2, pp. 282–304, 1978.

[29] E. A. Hansen, "Solving pomdps by searching in policy space," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 211–219, Morgan Kaufmann Publishers Inc., 1998.

[30] N. Meuleau, L. Peshkin, K.-E. Kim, and L. P. Kaelbling, "Learning finite-state controllers for partially observable environments," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 427–436, Morgan Kaufmann Publishers Inc., 1999.

[31] A. Wald, "Sequential tests of statistical hypotheses," *The Annals of Mathematical Statistics*, vol. 16, no. 2, pp. 117–186, 1945.

[32] M. H. DeGroot, *Optimal statistical decisions*, vol. 82. John Wiley & Sons, 2005.

[33] A. W. Drake, *Observation of a Markov process through a noisy channel.* PhD thesis, Massachusetts Institute of Technology, 1962.

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

[35] R. Zhou and E. A. Hansen, "An improved grid-based approximation algorithm for pomdps," in *IJCAI*, pp. 707–716, 2001.

[36] S. Karlin, *A first course in stochastic processes.* Academic press, 2014.

[37] "Optimizing nonlinear functions." `http://www.mathworks.com/help/matlab/math/optimizing-nonlinear-functions.html#bsgpq6p-11`, 2014.

[38] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the nelder–mead simplex method in low dimensions," *SIAM Journal on optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[39] S. A. Baeurle, "Multiscale modeling of polymer materials using field-theoretic methodologies: a survey about recent developments," *Journal of mathematical chemistry*, vol. 46, no. 2, pp. 363–426, 2009.

[40] A. LaVigna, A. M. Makowski, and J. S. Baras, *A continuous time distributed version of wald's sequential hypothesis testing problem.* Springer, 1986.

[41] V. V. Veeravalli, T. BaSar, and H. V. Poor, "Decentralized sequential detection with a fusion center performing the sequential test," *Information Theory, IEEE Transactions on*, vol. 39, no. 2, pp. 433–442, 1993.

[42] A. Nayyar and D. Teneketzis, "Sequential problems in decentralized detection with communication," *Information Theory, IEEE Transactions on*, vol. 57, no. 8, pp. 5410–5435, 2011.

[43] A. Y. Ng and M. Jordan, "Pegasus: A policy search method for large mdps and pomdps," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pp. 406–415, Morgan Kaufmann Publishers Inc., 2000.

[44] M. T. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for pomdps," *Journal of artificial intelligence research*, pp. 195–220, 2005.

[45] Z.-B. Tang, K. R. Pattipati, and D. L. Kleinman, "An algorithm for determining the decision thresholds in a distributed detection problem," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 21, no. 1, pp. 231–237, 1991.

[46] Z. B. Tang, *Optimization of Detection Networks.* PhD thesis, University of Connecticut, 1990.

[47] S. Boyd, "Alternating direction method of multipliers," in *Talk at NIPS Workshop on Optimization and Machine Learning*, 2011.

[48] S. You and Q. Peng, "A non-convex alternating direction method of multipliers heuristic for optimal power flow," in *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pp. 788–793, IEEE, 2014.

[49] M. Hong, Z.-Q. Luo, and M. Razaviyayn, “Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems,” *arXiv preprint arXiv:1410.1390*, 2014.

[50] T. Michael and I. Jordan, “Reinforcement learning algorithm for partially observable markov decision problems,” *Proceedings of the Advances in Neural Information Processing Systems*, pp. 345–352, 1995.

[51] S. Ross, B. Chaib-draa, and J. Pineau, “Bayesian reinforcement learning in continuous pomdps with application to robot navigation,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 2845–2851, IEEE, 2008.

[52] H. Kimura, K. Miyazaki, and S. Kobayashi, “Reinforcement learning in pomdps with function approximation,” in *ICML*, vol. 97, pp. 152–160, 1997.

[53] D. P. Bertsekas, “Convergence of discretization procedures in dynamic programming,” *Automatic Control, IEEE Transactions on*, vol. 20, no. 3, pp. 415–419, 1975.

[54] C. T. A. D. Subdivisions, “A course in triangulations for solving equations with deformations,” *Lecture Notes in Economics and Mathematical Systems*, vol. 234, 1984.