PERFORMANCE GUARANTEE OF A SUB-OPTIMAL POLICY FOR A DISCRETE MARKOV DECISION PROCESS AND ITS APPLICATION TO A ROBOTIC SURVEILLANCE PROBLEM

A Dissertation

by

MYOUNGKUK PARK

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Swaroop Darbha
Co-Chair of Committee,	Sivakumar Rathinam
Committee Members,	Reza Langari
	Sergiy Butenko
Head of Department,	Andreas Polycarpou

May 2014

Major Subject: Mechanical Engineering

ABSTRACT

This dissertation deals with the development and analysis of sub-optimal decision algorithms for a collection of robots that assist a remotely located operator in perimeter surveillance. The operator is tasked with the classification of incursions across the perimeter. Whenever there is an incursion into the perimeter, a nearby Unattended Ground Sensor (UGS) signals an alert. A robot services the alert by visiting the alert location, collecting evidence in the form of video imagery, and transmitting it to the operator.

The accuracy of operator's classification depends on the volume and freshness of information gathered and provided by the robots at locations where incursions occur. There are two competing needs for a robot: it needs to spend adequate time at an alert location to collect evidence for aiding the operator in accurate classification but it also needs to service other alerts as soon as possible, so that the evidence collected is relevant. The control problem is to determine the optimal amount of time a robot must spend servicing an alert. The incursions are stochastic and their statistics are assumed to be known.

The control problem may be posed as a Markov Decision Problem (MDP). Dynamic Programming(DP) provides the optimal policy to the MDP. However, because of the "curse of dimensionality" of DP, finding the optimal policy is not practical in many applications. For a perimeter surveillance problem with two robots and five UGS locations, the number of states is of the order of billions. Approximate Dynamic Programming (ADP) via Linear Programming (LP) provides a way to approximate the value function and derive sub-optimal strategies. Using state partitioning and ADP, this dissertation provides different LP formulations for upper and lower

bounds to the value function of the MDP, and shows the relationship between LPs and MDP. The novel features of this dissertation are (1) the derivation of a tractable lower bound via LP and state partitioning, (2) the construction of a sub-optimal policy whose performance exceeds the lower bound, and (3) the derivation of an upper bound using a non-linear programming formuation. The upper and lower bounds provides approximation ratio to the value function. Finally, illustrative perimeter surveillance examples corroborate the results derived in this dissertation.

ACKNOWLEDGEMENTS

It has been a long voyage since I arrived at this destination, my Ph.D degree in the Department of Mechanical Engineering at Texas A&M University. I believe that I could not make it without many forms of supports from many others.

When I met my advisor, Dr. Swaroop Darbha, I did not have any knowledge about stochastic dynamic systems and controls. But he took me as his student, taught me, and waited for me until I finally got on the same page with him. Even if I failed to get successful results for my research, he and his smile taught me how to learn from my failure and how to treat other people. So my first thanks go to him for his restless teaching. I would also like to thank my committee members: Dr. Sivakumar Rathinam, my co-advisor, who helped me to understand my research and how to present my works, and gave me a chance to work on an exciting "Quadrocopter" project that I have enjoyed very much; Dr. Reza Langari and Dr. Sergiy Butenko, for their guidance for my dissertation to go to the right direction.

Special thanks to Kent Marshall who spent an hour for every Tuesday with me to talk about many different issues cncerning our lives - I would name our meeting as 'Tuesdays with Kent.' The conversations we have had kept me asking the fundamental questions to myself so that I could stand on a solid ground, even for my research.

I want to thank my families in South Korea for support, especially my in-law family who gave me enormous help when I went through the toughest time of my life.

Lastly, I want to thank to Jung Yun - my wife, my colleague, my friend, and my biggest supporter. I cannot imagine myself being here without her and cannot find

the proper words to show my appreciation to her. The only thing I can say is 'this destination is just one of many many stops we will go through, and I will always be with you as a husband, a colleague, a friend, and the biggest supporter.' Oh, I cannot forget my little girl, Jayna. She is indeed the most precious gift from heaven to me as her name stands for, and becomes the biggest encouragement for me to complete my degree.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

This dissertation is motivated by a robotic perimeter surveillance problem. A collection of robots assists a remotely located human operator in the task of classification of an incursion across the perimeter of a protected zone as either a nuisance or a threat. Incursions are stochastic and have both a spatial and temporal component; we assume that the statistics of the incursion processes is known.

In order to aid the robot-operator team in the timely classification of incursions, the perimeter is installed with a set of Unattended Ground Sensors (UGSs) at locations where incursions can occur; these locations will be referred to as stations or UGS stations. At the stations, UGS flag incursions, raise alerts and communicate them immediately to the robots. Subsequently, a robot services the alert by visiting the UGS stations where it was raised, and transmitting images, video, or other sensory information to the operator using on-board camera and other sensing devices. The operator performs the role of a classifier based on the information supplied by the robots. The classification accuracy of the operator depends on the volume and freshness of information supplied by the robots.

For an accurate classification, the robot should provide as much video or other evidence about the incursion to the operator as possible. We call these information achieved by the robots as *information gain*. Subject to certain limits, the longer a robot spends at an alert location, the robot supplies a higher volume of information about the alert it services. However, the freshness of information it can gather about other unserviced alerts suffers. For timely and accurate classification of incursions, the *service delay time* or simply *delay time*, defined as the time delay between an alert

signal and the time a robot attends to the alert, should be minimized. Thus, there are two competing needs: a robot needs to spend more time at an alert location and it also needs to service the alerts as quickly as possible. A natural question arises: How long should a robot spend time servicing an alert?

In this dissertation, we discretize the problem spatially and temporally and recast the optimization problem as follows: Should the robot spend the next time interval at the current alert location in terms of maximizing the expected, discounted payoff? The payoff considered herein is an increasing function of the time spent at the alert site (*dwell time*) and a decreasing function of the delay in servicing alerts.

This problem can be naturally posed as a Markov Decision Problem (MDP). However, the number of states runs into billions even for a modest size problem. For example, if one considers two robots and eight alert locations, with a maximum allowable delay time of 15 units, the number of states exceeds trillion! Hence, solving Bellman's equation to compute the optimal payoff (value function) is computationally intractable. For this reason, we consider a Linear Programming (LP) based Approximate Dynamic Programming (ADP) solution strategy (see [27]). The LP based ADP approach provides an upper bound on the optimal value function, and an *a priori* estimate of the quality of the resulting sub-optimal policy, e. g., see [28, 29]. The quality is estimated by metrizing the deviation between the value function and its approximation.

The LP based ADP approach seeks an approximation that lies in the column space of a set of chosen basis vectors; often, finding the basis vectors in itself is a major challenge. The motivating application in this dissertation seems to suggest a simple choice for basis vectors that is based on state partitioning. The state is partitioned into prespecified number of partitions. One can associate a basis vector with each partition. The basis vectors are binary and indicate whether a state

belongs to the partition corresponding to the basis vector. There are challenges associated with ADP using state partitioning, especially with respect to refining the state partitioning when the approximation to value function is not satisfactory. In order to enable one to metrize whether an approximate value function is satisfactory or not, one must develop good lower and upper bounds. There is a significant gap in the literature in providing both upper and lower bounds to the value function of a MDP. This dissertation is a first attempt at addressing this gap. Our final goal is finding a sub-optimal policy guaranteeing certain level of performance when the policy applies to the process.

This dissertation is organized in the following order; in Section 2, we will provide preliminaries about MDP, DP, and ADP. In Section 3, two mathematical programs will be presented: the first is a restricted linear programming (RLP), and the other is a non-linear programming (NLP). We show that the solutions of RLP and DLP provide upper and lower bounds to the value function respectively. We also show that, for each LP, there exists a unique solution for non-negative cost funtion. Based on this result, we provide an iterative algorithm to efficiently find solution to the NLP. In Section 4, illustrative perimeter surveillance problems are formulated, and our LP approaches are applied to these problems. For each problem, a sub-optimal policy found from the LPs is applied and its effectiveness is corroborated via Monte Carlo simulations.

1.2 Literature Review

Perimeter surveillance problems arise in a variety of practical applications and have recently received significant attention in the literature; for example, see [15, 24, 22, 23]. From the point of view of this application area, the results described in this dissertation and our prior work in [4, 16, 17, 18, 19] differ from the literature

in addressing the need to balance the information gained by the robots with the quality of service requirement of attending to alerts raised at the UGS locations in a timely manner. From an analytical point of view, this dissertation constructs a sub-optimal policy based on a lower bound to the value function and provides a performance guarantee of the sub-optimal policy. From the organization point of view, this dissertation provides a distinction between properties that hold for general MDPs and those that exploit the structure of the robotic surveillance problem.

The use of LP techniques for solving DP problems was introduced by [21, 5]; the use of aggregation via partitioning and the construction of sub-optimal policies using approximate value functions was discussed in [25]. The LP based approach to approximate dynamic programming is discussed in [27, 28, 29]. The results in this paper differ from the existing literature on two counts: (1) the restricted or constrained LPs that one obtains for this class of applications are computationally tractable and hence, there is no need for column generation or random sampling techniques as in [28, 6], and (2) this work presents a way to construct upper as well as lower bounds using LPs, a marked departure from prior work in this area, other than by the co-authored work in [20].

2. PRELIMINARIES

2.1 Markov Decision Process

The MDP is a stochastic process whose state transition probability only depends on the current state and action. Consider a stochastic process with state space $\mathcal{S}$. Let $\mathbf{s}(t) \in \mathcal{S}$ and $\mathbf{a}(t) \in \mathcal{U}_{\mathbf{s}(t)}$ denote a state of the process and an action taken at time t respectively, where $\mathcal{U}_{\mathbf{s}(t)}$ is a set of available actions(or controls, decisions) at the current state $\mathbf{s}(t)$. It is assumed that the cardinality of each set is finite. In a MDP, state transition probability only depends on $\mathbf{s}(t)$ and $\mathbf{a}(t)$ as:

$$\begin{aligned} &\text{Prob } \{\mathbf{s}(t+1) = \mathbf{y} \mid \mathbf{s}(0), \ldots, \mathbf{s}(t-1), \mathbf{s}(t) = \mathbf{x}, \mathbf{a}(t) = \mathbf{u}\} && (2.1)\\ &= \text{Prob } \{\mathbf{s}(t+1) = \mathbf{y} \mid \mathbf{s}(t) = \mathbf{x}, \mathbf{a}(t) = \mathbf{u}\} && (2.2)\\ &= [p_{\mathbf{x},\mathbf{y}}(\mathbf{u})](t). && (2.3) \end{aligned}$$

This means that if a stochastic process is a MDP, then the history of the process until the process arrives at the current state $\mathbf{s}(t) = \mathbf{x}$ would not affect the probability distribution of the next state. We assume that the process is a time-invariant system such that $p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) = [p_{\mathbf{x},\mathbf{y}}(\mathbf{u})](0) = [p_{\mathbf{x},\mathbf{y}}(\mathbf{u})](1) = \cdots = [p_{\mathbf{x},\mathbf{y}}(\mathbf{u})](t)$ for $t \geq 0$. Since it is a probability distribution, it satisfies the following property: for any given $\mathbf{x} \in \mathcal{S}$ and $\mathbf{u} \in \mathcal{U}_{\mathbf{x}}$,

$$\sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) = 1, \text{ and } p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) \geq 0, \forall \mathbf{y} \in \mathcal{S}. \tag{2.4}$$

We assume a cost structure imposed on the MDP. If the process is in state $\mathbf{x}$ and action $\mathbf{a}$ is taken, we assume that a known cost $r(\mathbf{s}(t), \mathbf{a}(t)) \in \Re$ is incurred. In the

literature, the cost structure is called as one-step reward or pay-off function. If the reward function represents some profit by taking an action, then one would take an action maximizing the reward. On the other hand, if it is a pay-off function such as the distance between two vertices in a traveling salesman problem, then one may want to choose an action minimizing the pay-off function at each time step. In this dissertation, we want to find a sequence of actions such that maximizes the total expected discounted reward for infinite time horizon as following:

$$V^*(\mathbf{x}) = \max_{\pi \in \Pi_{h(t)}, t \geq 0} \mathbf{E} \left\{ \sum_{t=0}^{\infty} \lambda^t r(\mathbf{s}(t), \mathbf{a}_\pi(t)) \;\middle|\; \mathbf{s}(0) = \mathbf{x} \right\}, \forall \mathbf{x} \in \mathcal{S}, \tag{2.5}$$

where λ is a discount factor that $0 \leq \lambda \leq 1$, and $\Pi_{h(t)}$ is a set of all history dependent policies. Let $h(t)$ denote the history of the process until time t, which means, $h(t) = (\mathbf{s}(0), \mathbf{s}(1), \cdots, \mathbf{s}(t))$. Then a set of all avaiable action sequences corresponding to $h(t)$ is

$$\Pi_{h(t)} = \mathcal{U}_{\mathbf{s}(0)} \times \mathcal{U}_{\mathbf{s}(1)} \times \cdots \times \mathcal{U}_{\mathbf{s}(t)}.$$

An element, $\pi \in \Pi_{h(t)}$ is a sequence of actions such that $\pi = (\mathbf{a}_\pi(0), \mathbf{a}_\pi(1), \cdots, \mathbf{a}_\pi(t))$. The objective is to maximize the total expected discounted one-step rewards over all possible combination of available actions for infinite time horizon.

2.2 Dynamic Programming

One may be interested in determining the value function, $V^*(\mathbf{x})$, defined by (2.5) for every initial state $\mathbf{x}$. The value function, V^* is a vector of the same dimension as the number of states and the component corresponding to the state $\mathbf{x}$ is given by (2.5). Finding the value function requires very exhaustive computation and may not be computationally tractable from Eq. (2.5). However, Bellman in [1] introduced the DP approach to solve the MDP. From the DP, one can write a Bellman equation

for (2.5) as:

$$V^*(\mathbf{x}) = \max_{\mathbf{u}\in\mathcal{U}_\mathbf{x}} \left\{ r(\mathbf{x},\mathbf{u}) + \lambda \sum_{\mathbf{y}\in\mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V^*(\mathbf{y}) \right\}, \ \forall \mathbf{x} \in \mathcal{S}. \tag{2.6}$$

This Bellman equation satisfies the following properties (see [2] and [26] for details);

- there is exactly one solution,
- there is a *stationary* policy that is optimal in the class of all history dependent policies, and
- for each $\mathbf{x} \in \mathcal{S}$, let

$$\pi^*(\mathbf{x}) := \operatorname*{argmax}_{\mathbf{u}\in\mathcal{U}_\mathbf{x}} \left\{ r(\mathbf{x},\mathbf{u}) + \lambda \sum_{\mathbf{y}\in\mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V^*(\mathbf{y}) \right\}, \tag{2.7}$$

  then the policy π^* is the optimal stationary policy.

It must be noted that the value function defines the optimal policy as given in the above equation. A stationary policy $\pi^*(\mathbf{x})$ is a function of state $\mathbf{x} \in \mathcal{S}$ mapping into an available action space, $\mathcal{U}_\mathbf{x}$. Hereafter we assume that all policies are stationary.

In the following sub-sections, three methodologies to find the value function will be introduced. Before we discuss about the methods, let us define operators for convenience.

Definition (DP/Bellman Operator): The DP operator, $\mathcal{T}$, is defined as following; for $\forall\, \mathbf{x} \in \mathcal{S}$,

$$\mathcal{T}V(\mathbf{x}) := \max_{\mathbf{u}\in\mathcal{U}_\mathbf{x}} \left\{ r(\mathbf{x},\mathbf{u}) + \lambda \sum_{\mathbf{y}\in\mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V(\mathbf{y}) \right\}. \tag{2.8}$$

Define another DP operator for a given policy π, $\mathcal{T}_\pi$, such that

$$\mathcal{T}_\pi V(\mathbf{x}) := r(\mathbf{x}, \pi(\mathbf{x})) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\pi(\mathbf{x}))V(\mathbf{y}). \tag{2.9}$$

Lemma 1 (Contraction Mapping)**.** *The DP operator, $\mathcal{T}$, is a contraction mapping such that for any given $V, W \in \Re^{|\mathcal{S}|}$, there exists a $\lambda \in (0,1)$ satisfying*

$$||\mathcal{T}V - \mathcal{T}W||_\infty \leq \lambda ||V - W||_\infty,$$

where $||V - W||_\infty := \max_{\mathbf{x} \in \mathcal{S}} |V(\mathbf{x}) - W(\mathbf{x})|$.

Proof. For given V, W,

$$\begin{aligned}\mathcal{T}V(\mathbf{x}) =& \max_{\mathbf{u} \in \mathcal{U}_\mathbf{x}} \left\{ r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u})V(\mathbf{y}) \right\} \\ =& r(\mathbf{x}, \mathbf{u}'(\mathbf{x})) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}'(\mathbf{x}))V(\mathbf{y}),\end{aligned}$$

where $\mathbf{u}'(\mathbf{x}) := \text{argmax}_{\mathbf{u} \in \mathcal{U}_\mathbf{x}} \left\{ r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u})V(\mathbf{y}) \right\}$. Similarly,

$$\begin{aligned}\mathcal{T}W(\mathbf{x}) =& \max_{\mathbf{u} \in \mathcal{U}_\mathbf{x}} \left\{ r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u})W(\mathbf{y}) \right\} \\ \geq& r(\mathbf{x}, \mathbf{u}'(\mathbf{x})) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}'(\mathbf{x}))W(\mathbf{y}).\end{aligned}$$

Then,

$$\begin{aligned}\mathcal{T}V(\mathbf{x}) - \mathcal{T}W(\mathbf{x}) \leq& \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}'(\mathbf{x}))(V(\mathbf{y}) - W(\mathbf{y})) \\ \leq& \lambda \max_{\mathbf{x} \in \mathcal{S}} |V(\mathbf{x}) - W(\mathbf{x})| = \lambda ||V - W||_\infty.\end{aligned}$$

Similarly,

$$\begin{aligned}\mathcal{T}W(\mathbf{x}) - \mathcal{T}V(\mathbf{x}) \leq & \lambda \sum_{\mathbf{y}\in\mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}''(\mathbf{x}))(W(\mathbf{y}) - V(\mathbf{y})) \\ \leq & \lambda \max_{\mathbf{x}\in\mathcal{S}} |W(\mathbf{x}) - V(\mathbf{x})| = \lambda ||V - W||_\infty,\end{aligned}$$

where $\mathbf{u}''(\mathbf{x}) := \text{argmax}_{\mathbf{u}\in\mathcal{U}_\mathbf{x}} \left\{ r(\mathbf{x},\mathbf{u}) + \lambda \sum_{\mathbf{y}\in\mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) W(\mathbf{y}) \right\}$. Hence,

$$||\mathcal{T}V - \mathcal{T}W||_\infty \leq \lambda ||V - W||_\infty$$

and it is a contraction mapping. □

Lemma 2 (Monotonicity)**.** *For any given policy π, if $V \geq W$ componentwise, then $\mathcal{T}_\pi V \geq \mathcal{T}_\pi W$.*

Proof. Assume that $V(\mathbf{x}) \geq W(\mathbf{x}), \forall\, \mathbf{x} \in \mathcal{S}$, then

$$\begin{aligned}\mathcal{T}_\pi V(\mathbf{x}) - \mathcal{T}_\pi W(\mathbf{x}) = & \lambda \sum_{\mathbf{y}\in\mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\pi(\mathbf{x}))(V(\mathbf{y}) - W(\mathbf{y})) \geq 0 \\ \rightarrow & \mathcal{T}_\pi V(\mathbf{x}) \geq \mathcal{T}_\pi W(\mathbf{x}), \forall\, \mathbf{x} \in \mathcal{S}\end{aligned}$$

□

2.2.1 Value Iteration

In [1], Bellman porposed an iterative method to compute the value function as shown in Algorithm 1.

From the definition of DP operator, we can see that Step 3 to 8 of Algorithm 1 are nothing but the process of the DP operator, which means,

$$V^{t+1}(\mathbf{x}) = \mathcal{T}V^t(\mathbf{x}).$$

Algorithm 1 Value Iteration

Initialize $t \leftarrow 0$ and $V^t(\mathbf{x})$ arbitrarily.
do
  for $\forall \mathbf{x} \in \mathcal{S}$ **do**
    for $\forall \mathbf{u} \in \mathcal{U}_{\mathbf{x}}$ **do**
      $Q(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V^t(\mathbf{y})$
    end for
    $V^{t+1}(\mathbf{x}) = \max_{\mathbf{u} \in \mathcal{U}_{\mathbf{x}}} Q(\mathbf{x}, \mathbf{u})$
  end for
  $t \leftarrow t + 1$
while until policy is good enough, e.g. $||V^t - V^{t-1}||_\infty > \epsilon$

Using the property of the DP operator, the convergence property of the value iteration can be shown as follows.

Proof of convergence.

$$
\begin{aligned}
||V^2 - V^1|| =& ||\mathcal{T}(\mathcal{T}V^0) - \mathcal{T}V^0|| \leq \lambda ||\mathcal{T}V^0 - V^0|| \\
||V^3 - V^2|| =& ||\mathcal{T}(\mathcal{T}^2V^0) - \mathcal{T}(\mathcal{T}V^0)|| \leq \lambda ||\mathcal{T}^2V^0 - \mathcal{T}V^0|| \leq \lambda^2 ||\mathcal{T}V^0 - V^0|| \\
& \vdots
\end{aligned}
$$

By induction,

$$||V^{t+1} - V^t|| \leq \lambda^t ||\mathcal{T}V^0 - V^0||.$$

So, as the number of iteration increases, $||V^{t+1} - V^t||$ decreases, and it satisfies the termination criterion and stops the iteration in finite steps. For the same reason,

$$\lim_{t \to \infty} ||V^{t+1} - V^t|| = 0,$$

that is, as $t \to \infty$, $V^{t+1} = V^t = z$ where z is a fixed point;

$$\lim_{t\to\infty} \mathcal{T}^t V^0 = z.$$

□

2.2.2 Policy Iteration

Policy iteration is a major alternative to the VI. In the VI, if $\epsilon = 0$, then the iteration might run indefinitely because $V^t = V^*$ when $t \to \infty$. However, Howard [14] proposed an iterative method complementing the weakness. It is shown in Algorithm 2 and will be terminated within finite number of iteration.

Algorithm 2 Policy Iteration

Initialize $t \leftarrow 0$ and pick a policy π^t arbitrarily.
repeat
  Policy Evaluation: $V_{\pi^t} = [I - \lambda P_{\pi^t}]^{-1} R_{\pi^t}$.
  for $\forall \mathbf{x} \in \mathcal{S}$ **do**
    for $\forall \mathbf{u} \in \mathcal{U}_{\mathbf{x}}$ **do**
      $Q(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{\mathbf{y}\in\mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V_{\pi^t}(\mathbf{y})$
    end for
    $V^t(\mathbf{x}) = \max_{\mathbf{u}\in\mathcal{U}_{\mathbf{x}}} Q(\mathbf{x}, \mathbf{u})$
    $\pi^{t+1}(\mathbf{x}) = \operatorname{argmax}_{\mathbf{u}\in\mathcal{U}_{\mathbf{x}}} Q(\mathbf{x}, \mathbf{u})$
  end for
  $t \leftarrow t + 1$
until $V^{t-1} = V_{\pi^{t-1}}$

In this algorithm, P_π and R_π are a transition probability matrix and a one-step reward vector only associating with a given policy π. That is, the $(\mathbf{x}, \mathbf{y})$th element of P_π is $p_{\mathbf{x},\mathbf{y}}(\pi(\mathbf{x}))$, and $R_\pi(\mathbf{x}) = r(\mathbf{x}, \pi(\mathbf{x}))$.

Proof of convergence. For any $\mathbf{x} \in \mathcal{S}$ and t, the following inequality holds;

$$\begin{aligned} V_{\pi^t}(\mathbf{x}) =& r(\mathbf{x}, \pi^t(\mathbf{x})) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\pi^t(\mathbf{x})) V_{\pi^t}(\mathbf{y}) \\ \leq & \max_{\mathbf{u} \in \mathcal{U}_{\mathbf{x}}} \{ r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V_{\pi^t}(\mathbf{y}) \} \\ =& r(\mathbf{x}, \pi^{t+1}(\mathbf{x})) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\pi^{t+1}(\mathbf{x})) V_{\pi^t}(\mathbf{y}) = V^t(\mathbf{x}). \end{aligned}$$

If π^t is optimal, then $V_{\pi^t}(\mathbf{x}) = V^t(\mathbf{x}), \forall\, \mathbf{x} \in \mathcal{S}$ and the iteration will be terminated by the termination criterion. If it is not optimal, then there will exist some $\mathbf{x} \in \mathcal{S}$ such that

$$V_{\pi^t}(\mathbf{x}) < r(\mathbf{x}, \pi^{t+1}(\mathbf{x})) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\pi^{t+1}(\mathbf{x})) V_{\pi^t}(\mathbf{y}).$$

However, from the monotonicity, the following inequality holds;

$$V_{\pi^t}(\mathbf{x}) < \mathcal{T}_{\pi^{t+1}} V_{\pi^t}(\mathbf{x}) \leq \mathcal{T}^2_{\pi^{t+1}} V_{\pi^t}(\mathbf{x}) \leq \mathcal{T}^3_{\pi^{t+1}} V_{\pi^t}(\mathbf{x}) \leq \cdots \leq V_{\pi^{t+1}}(\mathbf{x}).$$

So, in each iteration, there will be improvement which is strictly greater than previous step for at least one state. Eventually, it terminates with the optimal policy within $|\mathcal{S}|$ iterations. □

2.2.3 Linear Programming Approach to Dynamic Programming

In this subsection, a linear program, referred to as exact LP (ELP), will be presented. The optimal solution of the ELP is also the value function. In order to describe the constraints of the ELP one requires the following observation: Bellman's equation, Eq. (2.5), suggests that the optimal value function, V^*, satisfies the

following set of linear inequalities, referred to as the Bellman Inequalities:

$$V(\mathbf{x}) \geq r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V(\mathbf{y}),\ \forall \mathbf{u} \in \mathcal{U}_{\mathbf{x}}, \forall \mathbf{x} \in \mathcal{S}, \tag{2.10}$$

or more compactly,

$$V \geq R(\mathbf{u}) + \lambda P(\mathbf{u}) V,\ \forall \mathbf{u}. \tag{2.11}$$

The following lemma shows that any feasible solution of the Bellman Inequalities is an upper bound to the value function.

Lemma 3. *Any feasible solution of the Bellman Inequalities (2.10) is an upper bound to the value function.*

Proof. The Bellman Inequalities (2.10) are feasible. To see that, set all the components of V to be equal to $\frac{\max_{\mathbf{x},\mathbf{u}} r(\mathbf{x},\mathbf{u})}{1-\lambda}$.

Let V be a feasible solution to the Bellman inequalities. Let π be any stationary policy and let P_π, R_π be the associated state transition probability matrix and one-step reward vector respectively. Then, for every π, the feasible solution, V satisfies

$$V \geq R_\pi + \lambda P_\pi V,$$

implying that

$$[I - \lambda P_\pi] V \geq R_\pi.$$

By the non-negativity and contraction of λP_π, the inverse of $[I - \lambda P_\pi]$ is non-negative

and has the following power series expansion:

$$[I - \lambda P_\pi]^{-1} = I + \lambda P_\pi + \lambda^2 P_\pi^2 + \lambda^3 P_\pi^3 + \dots .$$

Hence,

$$V \geq [I - \lambda P_\pi]^{-1} R_\pi = R_\pi + \lambda P_\pi R_\pi + \lambda P_\pi^2 R_\pi + \lambda P_\pi^3 R_\pi + \cdots , \ \ \forall \pi.$$

If the stationary policy π corresponds to an optimal stationary policy, then from the definition (2.5), the right-hand side of the inequality equals the value function, V. Hence, *every feasible solution V to the Bellman inequalities upper bounds the value function V^*.* □

As a result, for every non-negative cost function, the value function, V^*, is an optimal solution of the following LP, referred to as the Exact LP (ELP):

$$J_{\text{ELP}} = \min \sum_{\mathbf{x} \in \mathcal{S}} c(\mathbf{x}) V(\mathbf{x}), \tag{2.12}$$

$$V(\mathbf{x}) \geq r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V(\mathbf{y}), \ \ \forall \mathbf{u} \in \mathcal{U}_{\mathbf{x}}, \forall \mathbf{x} \in \mathcal{S}. \tag{2.13}$$

Formally,

Lemma 4. *V^* is the value function if and only if V^* is optimal for the ELP for every non-negative c.*

Proof. Let V_{LP} be an optimal solution of ELP corresponding to *every* cost vector c. The existence of V_{LP} can be asserted from the following observation: If V_1, V_2 are feasible solutions to the Bellman Inequalities (2.10), then the componentwise minimum $\min\{V_1, V_2\}$ of the solutions is also a solution to the Bellman Inequalities

(2.10). Since all feasible solutions to (2.10) are lower bounded by V^*, it follows that $V_{LP} := \min\{V : V\ satisfies\ Bellman\ Inequalities\}$ is well-defined and also satisfies Bellman Inequalities. Moreover, by construction, every feasible solution to (2.10) is lower bounded by V_{LP}. Hence, for every non-negative c, V_{LP} is optimal.

From Lemma 1, $V_{LP} \geq V^*$ and by construction $V^* \geq V_{LP}$ as V^* satisfies Bellman Inequalities (2.10). Hence, $V^* = V_{LP}$. □

It is remarkable that the family of LPs corresponding to different non-negative cost functions admit the same optimal solution. This invariance property is useful in practice as one need not have to concern with the choice of the cost function as long as it is non-negative.

3. LINEAR PROGRAMMING APPROACH TO APPROXIMATE DYNAMIC PROGRAMMING

The construction of optimal policy often requires the computation of value function. In the previous section, different methodologies for computing the value function were referred to such as the Value Iteration, Policy Iteration and LP method. However, one often encounters "the curse of dimensionality" in the application of Dynamic Programming to determine optimal policies for controlled Markov chains; essentially, it implies that the computation of value function and the optimal policy become computationally intractable as the number of states of the associate Markov Decision Process becomes large. In practice, optimality is traded for computational tractability to obtain approximate value functions and sub-optimal policies. A natural question arises: How close is the sub-optimal policy to the optimal policy or how good is the sub-optimal policy in relation to the optimal policy? Can one estimate the bounds of sub-optimality of the policy? It is reasonable to expect that a good approximation to the value function yield good approximation to the optimal policy. For this reason, one can metrize sub-optimality by the deviation between the value function and its approximation.

Since value function is difficult to compute, one can estimate bounds on sub-optimality by computing both upper and lower bounds to the value function. It is entirely possible that one of the bounds may be very close to the value function, while the other is quite far leading to a conservative estimate of the quality of the sub-optimal policy. However, having both upper and lower bounds to the value function may be useful in refining the bounds at the expense of additional computation.

The focus of this section is in constructing LPs that provide upper and lower

bounds for the value function. As seen in the previous section, every feasible solution to the Bellman Inequalities upper bounds the value function. A standard procedure to construct an upper bound then is to restrict the feasible set of ELP so that an optimal solution to the RLP is easy to find. This is the approach adopted in this dissertation. In the first part of the section, additional properties concerning the RLP and its optimal solution are discussed. In the latter part of the section, the focus is on the computation of the lower bound using the NLP.

In the construction of upper and lower bounds to the value function, state partitioning is adopted. The idea is to partition the set of states into a few partitions and approximate the value function to be a constant across each partition. These constraints are linear and are augmented to the Bellman Inequalities of ELP. By doing so, the number of variables of the augmented or RLP becomes smaller and the feasible set of the ELP is also smaller because of the additional restriction. The number of constraints may or may not reduce depending on the structure of the problem at hand.

In the following subsection, we discuss partitioning and the associated RLP used in the construction of an upper bound.

3.1 Partitioning

Assume that the cardinality of the state space of a MDP is n such that $n = |\mathcal{S}|$, then the states can be labelled with integers from one to n with one-to-one correspondence. For notational convenience, a state x represents a vector of state variables and also an index of the state. Let the set of all states $\mathcal{S}$ be partitioned into m disjoint sets, $\mathcal{S}_i, i = 1, 2, \cdots, m$. We define a *General Partitioning Scheme* of $\mathcal{S}$ as follows:

Definition (General Partitioning Scheme): Let $n \geq 1$. We will refer to the set $\mathcal{GP} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m\}$ as a general partitioning scheme of cardinality m if

(i) $\mathcal{S}_1, \ldots, \mathcal{S}_m$ are disjoint subsets of $\mathcal{S}$ and their union is $\mathcal{S}$,

(ii) any two states in $\mathcal{S}_i$ have the same set of allowable controllable actions.

We will call the sets $\mathcal{S}_1, \ldots, \mathcal{S}_n$ as general partitions, or simply partitions.

For a given $\mathcal{GP}$, all states in a partition have same available action set; $\mathcal{U}_{\mathbf{x}} = \mathcal{U}_{\mathbf{y}}, \forall \mathbf{x}, \mathbf{y} \in \mathcal{S}_i, i = 1, 2, \cdots, m$. However, all states with same avaiable action set do not have to be in the same partition. Let $\mathcal{U}_{\mathrm{i}}$ denote the available action set for the i^{th} partition in $\mathcal{GP}$.

3.2 Upper Bounding Linear Programming

3.2.1 Restricted Linear Programming

Recall the ELP from the previous section:

$$J_{\mathrm{ELP}} = \min \sum_{\mathbf{x} \in \mathcal{S}} c(\mathbf{x}) V(\mathbf{x}),$$
$$V(\mathbf{x}) \geq r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V(\mathbf{y}), \quad \forall \mathbf{u} \in \mathcal{U}_{\mathbf{x}}, \forall \mathbf{x} \in \mathcal{S}.$$

We restrict the ELP with a given $\mathcal{GP}$ such that the value function of states in a partition have the same value. That is, $V(\mathbf{x}) = V(\mathbf{y}), \forall \mathbf{x}, \mathbf{y} \in \mathcal{S}_i, i = 1, \cdots, m$. Augmenting these constraints to the ELP, one gets the following restricted LP (RLP);

$$J_{\mathrm{RLP}} = \min \sum_{\mathbf{x} \in \mathcal{S}} c(\mathbf{x}) V(\mathbf{x}), \tag{3.1}$$
$$V(\mathbf{x}) \geq r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V(\mathbf{y}), \quad \forall \mathbf{u} \in \mathcal{U}_{\mathbf{x}}, \forall \mathbf{x} \in \mathcal{S}, \tag{3.2}$$
$$V(\mathbf{x}) = V(\mathbf{y}), \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{S}_i, i = 1, \cdots, m. \tag{3.3}$$

Theorem 1. *There exists a solution V^*_{RLP} to the RLP defined by the cost (3.1) and constraints (3.2) and (3.3) that is optimal for every non-negative c.*

Proof. Let V_1, V_2 be a feasible solution to the constraints (3.2) and (3.3). Then, the componentwise minimum, $\min\{V_1, V_2\}$ also satisfies the constraints (3.2) and (3.3). Since every feasible solution to (3.2) upper bounds the value function V^*, it follows that $V^*_{RLP} := \min\{V : V \; satisfies \; Bellman \; Inequality \; constraint$ (3.2) *and partitioning constraint* (3.3) $\}$ is well defined and also satisfies constraints (3.2) and (3.3). By construction, every feasible solution, V, to RLP therefore upper bounds V^*_{RLP}, which, in itself, is a feasible solution to RLP; in other words, $V \geq V^*_{RLP}$. Hence, for every $c \geq 0$, $c^T V \geq c^T V^*_{RLP}$ and hence, V^*_{RLP} is optimal for every $c \geq 0$. □

Although the dimension of RLP is still same as that of ELP, we can reformulate the RLP as an LP in smaller number of variables as:

$$J_{\text{RLP}} = \min \sum_{i=1}^{m} \bar{c}(i) v(i), \tag{3.4}$$

$$v(i) \geq r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) v(j), \quad \forall \mathbf{u} \in \mathcal{U}_{\text{i}}, \; \forall \mathbf{x} \in \mathcal{S}_i, i = 1, \cdots, m, \tag{3.5}$$

where $\bar{c}(i) = \sum_{\mathbf{x} \in \mathcal{S}_i} c(\mathbf{x})$. We will refer to this LP as Reformulated RLP or RRLP. Let the solution to RRLP be v^*, so that $V^*_{RLP}(\mathbf{x}) = v^*(i), \forall \, \mathbf{x} \in \mathcal{S}_{\text{i}}, i = 1, \cdots, m$. Moreover, since V^*_{RLP} satisfies Bellman Inequalities (3.2), it follows that $V^*_{RLP}(\mathbf{x}) \geq V^*(\mathbf{x}), \forall \, \mathbf{x} \in \mathcal{S}$. If $m << n$, then one can find a upper bound by dealing with an LP involving smaller number of variables.

We can recast the results of Theorem 1 as it relates to the RLP with smaller number of variables as:

- There exists a solution v^* to the RRLP that is optimal for every $c \geq 0$ and every feasible solution, v, to RRLP dominates v^*, i.e., $v \geq v^*$.

- Moreover, it is the unique solution for RRLP for any $\bar{c} > 0$.

Theorem 1 is important from the following viewpoints:

- It is well known that the cost function c can be considered as the initial state distribution of a MDP. If a partition scheme is given, then $\bar{c}(i)$ is the initial probability of states lying in the partition $\mathcal{S}_i$. If the solution to the RRLP depends on $\bar{c}$, then it will imply that the solution to the RRLP depends on the initial probability distribution. If so, one must solve the RRLP every time the initial state distribution changes, which can be cumbersome computationally. However, if it is independent of $\bar{c}$, then one can pick any initial probability distribution of states so that there is a non-zero initial probability for a state lying in each partition.

- Theorem 1 also implies that the upper bound for the optimal value function cannot be improved by changing the cost function from a linear to a non-linear function or by restricting the feasible set of RLP further since the optimal solution of RLP is dominated by every feasible solution of RLP.

  Hence, a refinement of the upper bound must necessarily involve an enlargement of the feasible set if one wants to stick to an LP formulation, i.e., it should include the feasible set of RLP and possibly other tighter upper bounds than the optimal solution of RLP. Lifting of variables is one way to improve the bound; in this connection, we show in the following section that neither a general lifted LP nor one obtained by including the iterated Bellman inequalities in the constraint set improves the upper bound.

- The ELP for the original MDP were described described by the objective function in (2.12) and Bellman Inequality constraints in (2.13). In this case, the

optimal solution was shown to be independent of the positive cost function, and it is, in fact, the value function, V^*. So, a natural question that arises is the following: Is the optimal solution to the RLP also the optimal value function correspoding to a MDP of size m and if so, how is this MDP related to the original problem? To answer this question, we make the following observation. The constraints in RLP, Eq. (3.4) and (3.5), do not, in general, correspond to those of an MDP because the transition from one partition to another for a given control $\mathbf{u}$ is not specified unambiguously. This is because different states in the same partition can transition to different partitions for the same $\mathbf{u}$ and stochastic input. However, Theorem 1 provides a clue that there exists an underlying MDP in the RLP whose value function is an upper bound to the value function of the original MDP. In Section 3.2.3, we discuss about this in detail.

3.2.2 Lifted Restricted Linear Programming

It may appear that we can get tighter upper bounds than those provided by the RLP by considering either lifted LPs whose feasible set is larger than that of RLP or LPs with a different objective function. We will show, in this section, that unfortunately this is not the case. In general, one can construct a lifted LP (LLP) of the form:

$$J_{\text{LLP}} = \min \sum_{i=1}^{m} \bar{c}(i)v(i) + d^T z, \tag{3.6}$$

$$V(\mathbf{x}) \geq r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V(\mathbf{y}), \quad \forall \mathbf{u} \in \mathcal{U}_{\mathbf{x}}, \forall \mathbf{x} \in \mathcal{S}, \tag{3.7}$$

$$V(\mathbf{x}) = v(i), \quad \forall \mathbf{x} \in \mathcal{S}_i, i = 1, \cdots, m, \tag{3.8}$$

$$z \geq 0. \tag{3.9}$$

where z is the additional vector of variables used in lifting so that the feasible set is not empty. Then, if follows that if $(\tilde{V}, \tilde{v}, \tilde{z})$ is optimal to LLP, then $\tilde{V}$ will be feasible solution to RLP. Consequently, $\tilde{V} \geq V^*_{RLP}$. In other words, one gets no better bound via lifting if the constraints (3.7) and (3.8) are included.

One could also use the iterated Bellman inequalities for constructing a lifted LP (LLP) of the form:

$$J_{IB} = \min \sum_{k=1}^{L} \bar{c}^T v_k, \tag{3.10}$$

$$v_{k+1}(i) \geq r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{j=1}^{m} \mathbf{p}_{\mathbf{x},j}(\mathbf{u}) v_k(j), \ \forall \mathbf{u} \in \mathcal{U}_{\mathbf{x}}, \forall \mathbf{x} \in \mathcal{S}_i, \forall i, k = 1, \cdots, L-1, \tag{3.11}$$

$$v_1(i) \geq r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{j=1}^{m} \mathbf{p}_{\mathbf{x},j}(\mathbf{u}) v_L(j), \ \forall \mathbf{u} \in \mathcal{U}_{\mathbf{x}}, \forall \mathbf{x} \in \mathcal{S}_i, \forall i, \tag{3.12}$$

where $\mathbf{p}_{\mathbf{x},j}(\mathbf{u}) = \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u})$, which means, $\mathbf{p}_{\mathbf{x},j}(\mathbf{u})$ is a transition probability from state $\mathbf{x}$ to partition $\mathcal{S}_j$ under influence of action $\mathbf{u}$. Again, it turns out that the above lifted LP is incapable of providing a better bound, as can be seen from the following result.

Theorem 2. *If $v_{IB} = (v_1, \cdots, v_L)$ is a feasible solution to J_{IB}, then $v_k \geq v^*$, $k = 1, \cdots, L$, where v^* is the optimal solution to RRLP.*

Proof. The proof follows along the lines of Theorem 1 and its essential steps are:

- Show that every feasible solution is lower bounded.
- Construct a feasible solution to LLP that is the componentwise minimum of all feasible solutions and show that all its components equal $v_1, v_2, \ldots, v_L$ equal v^*. Then, show that it is optimal to LLP.

We first observe that the feasible set of LLP is not empty because setting $v_1 = v^*, v_2 = v^*, \ldots v_L = v^*$ readily satisfies the constraints of LLP as v^* satisfies the constraints of RRLP.

Let $(v_1, v_2, \ldots, v_L)$ be a feasible solution to the LLP; the solution $v_k = v^*,\ k = 1, 2, \ldots, L$ is feasible to the LLP. Then, it is easy to see that the componentwise minimum of the component vectors, $v_1, v_2, \ldots, v_L$, given by $\min\{v_1, v_2, \ldots, v_L\}$ satisfies the constraints of RRLP. Since every feasible solution of RRLP dominates v^*, it follows that $\min\{v_1, v_2, \ldots, v_L\} \geq v^*$. Hence, $v_k \geq v^*$ for $k = 1, 2, \ldots, L$.

Again, due to the non-negativity of the probabilities, the componentwise minimum of any two feasible solutions of LLP is also feasible. Since every solution of LLP is lower bounded, the componentwise minimum of all feasible solutions of LLP, say $(v^*_{1,LLP}, v^*_{2,LLP}, \ldots, v^*_{L,LLP})$ is well-defined and is a feasible solution to LLP. It is easy to observe that if $(v_1, v_2, \ldots, v_L)$ is a feasible solution to LLP, then any cyclical permutation of $(v_1, v_2, \ldots, v_L)$ is also a feasible solution to LLP; for example, $(v_2, v_3, \ldots, v_L, v_1)$ and $(v_3, v_4, \ldots, v_1, v_2)$ are two other feasible solutions. For this reason, $v^*_{1,LLP} = v^*_{2,LLP} = \ldots = v^*_{L,LLP}$. In other words, the componentwise minimum of all solutions can be expressed as $(v^*_{LLP}, v^*_{LLP}, \ldots, v^*_{LLP})$ by dropping the subscript indicating the index.

Since $(v^*, v^*, \ldots, v^*)$ is a feasible solution to LLP, by construction, $v^* \geq v^*_{LLP}$. By observing that v^*_{LLP} is feasible for RRLP, it follows that $v^*_{LLP} \geq v^*$. Hence $v^*_{LLP} = v^*$. Therefore, if $(v_1, v_2, \ldots, v_L)$ is feasible to LLP, then $v_k \geq v^*,\ k = 1, 2, \ldots, L$.

For any $\bar{c} \geq 0$, clearly, $(v^*, v^*, \ldots, v^*)$ is optimal as it is feasible to LLP and $v_k \geq v^*$ for every k implies $\sum_{k=1}^{L} \bar{c}^T v_k \geq \sum_{k=1}^{L} \bar{c}^T v^*$.

□

So, we conclude that lifting through the use of iterated Bellman inequalities does

not help in finding a tighter upper bound than the RLP optimal solution. Also using any other non-linear objective function will not improve the upper bound as long as the iterated Bellman inequlities are included in the constraints set. In the next section, we focus our attention on the construction of a lower bound for the value function.

3.2.3 Relationship between Exact Linear Programming and Reformulated Restricted Linear Programming

As mentioned before, the constraints in RRLP do not correspond to those of an MDP, because the transition from one partition to another for a given control $\mathbf{u}$ is not specified unambiguously. Suppose we use a random selector to select a state from a partition, then the specification of $\mathbf{u}$ with the random selector tells us which partition the system would transition to next, from the current partition. A question is: how does one specify this random selector? To answer this, we consider the dual problem of the RRLP:

$$
\begin{aligned}
J_{DRLP} = & \max \sum_{i=1}^{m} \sum_{\mathbf{x} \in \mathcal{S}_\mathrm{i}} \sum_{\mathbf{u} \in \mathcal{U}_\mathrm{i}} \mu^i_{\mathbf{u}}(\mathbf{x}) r(\mathbf{x}, \mathbf{u}) \qquad (3.13) \\
& \sum_{\mathbf{x} \in \mathcal{S}_\mathrm{i}} \sum_{\mathbf{u} \in \mathcal{U}_\mathrm{i}} \mu^i_{\mathbf{u}}(\mathbf{x}) \left[v(i) - \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) v(j) \right] \leq \bar{c}(i),\ i = 1, \cdots, m, \\
& \mu^i_{\mathbf{u}} \geq 0.
\end{aligned}
$$

Recall that, for a given partition index i, the RRLP specifies a constraint on $v(i)$ for each $\mathbf{x} \in \mathcal{S}_\mathrm{i}$ and $\mathbf{u} \in \mathcal{U}_\mathrm{i}$; the corresponding dual variable is $\mu^i_{\mathbf{u}}(\mathbf{x})$. Let the optimal dual variable, that solves DRLP, be $\bar{\mu}^i_{\mathbf{u}}(\mathbf{x})$. We show, via the following lemma, that the so-called "surrogate dual" [8, 9, 7] obtained by aggregating the constraint of RRLP via the optimal dual variables is equivalent to the RRLP and moreover, is the

exact LP corresponding to a reduced order MDP, defined over the partitions.

Lemma 5. *Consider a surrogate LP (SLP) for the RRLP through the vector of dual variables given by:*

$$J_{SLP}(\mu) = \min \sum_{i=1}^{m} \bar{c}(i)v(i), \tag{3.14}$$

$$\sum_{\mathbf{x}\in\mathcal{S}_i} \mu^i_{\mathbf{u}}(\mathbf{x})v(i) \geq \sum_{\mathbf{x}\in\mathcal{S}_i} \mu^i_{\mathbf{u}}(\mathbf{x}) \left[r(\mathbf{x},\mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y}\in\mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u})v(j) \right], \forall \mathbf{u} \in \mathcal{U}_i, i = 1, \cdots, m.$$

Then, the surrogate dual problem (SDP) is related to the RRLP is following manner:

$$J_{SDP} = \max_{\mu\geq 0} J_{SLP}(\mu)$$

and

$$J_{SDP} = J_{RLP}.$$

Proof. Let us define a function ϕ as following;

$$\phi(v,\mu) := \sum_{i=1}^{m} \bar{c}(i)v(i) - \sum_{i=1}^{m} \sum_{\mathbf{x}\in\mathcal{S}_\mathrm{i}} \sum_{\mathbf{u}\in\mathcal{U}_\mathrm{i}} \mu^i_{\mathbf{u}}(\mathbf{x}) \left[v(i) - r(\mathbf{x},\mathbf{u}) - \lambda \sum_{j=1}^{m} \sum_{\mathbf{y}\in\mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u})v(j) \right], \tag{3.15}$$

then Lagrangian function, $L(\mu)$, used in the dual problem (LD) to the RRLP is

$$L(\mu) = \min_{v} \phi(v,\mu).$$

The Lagrangian dual is:

$$J_{LD} = \max_{\mu\geq 0} L(\mu),$$

respectively. Let $\mathcal{F}$ be the feasible set for the RRLP and let $\mathcal{F}(\mu)$ be the feasible set

of $J_{SLP}(\mu)$. Since $\mathcal{F} \subset \mathcal{F}(\mu), \forall \mu \geq 0$,

$$J_{RLP} \geq J_{SLP}(\mu) = \min_{v \in \mathcal{F}(\mu)} \bar{c}^T v \geq \min_{v \in \mathcal{F}(\mu)} \phi(v, \mu) \geq L(\mu), \ \forall \mu \geq 0. \tag{3.16}$$

Since the above inequality holds for all $\mu \geq 0$, by maximizing over $\mu \geq 0$, we have the following inequality:

$$J_{RLP} \geq J_{SDP} \geq J_{LD}.$$

Note that the primal problem, namely RRLP is feasible; a feasible solution is given by a vector v with all its components being $\frac{\max_{\mathbf{x},\mathbf{u}} r(\mathbf{x},\mathbf{u})}{1-\lambda}$. Since the primal problem is feasible, it satisfies the strong duality condition for LPs, and hence, $J_{LD} = J_{RLP}$. Hence, $J_{SDP} = J_{RLP}$. □

Note that there exists an optimal dual variable vector $\bar{\mu}$ such that $J_{LD} = L(\bar{\mu})$. From the inequality (3.16), it follows that $J_{SLP}(\bar{\mu}) = J_{LD} = J_{RLP}$.

Now consider $J_{SLP}(\mu)$ with $\mu = \bar{\mu}$; for every partition index $i = 1, \cdots, m$, there exists at least one $\bar{\mu}^i_{\mathbf{u}}(\mathbf{x})$ such that $\bar{\mu}^i_{\mathbf{u}}(\mathbf{x}) > 0$. If some i, $\bar{\mu}^i_{\mathbf{u}}(\mathbf{x}) = 0$ for every $\mathbf{x} \in \mathcal{S}_{\mathrm{i}}$ and every $\mathbf{u} \in \mathcal{U}_{\mathrm{i}}$, then $J_{SLP}(\bar{\mu})$ will not have any constraints lower bounding $v(i)$. It will then admit solutions for $v(i)$ that are arbitrarily negative and correspondingly, one can find a direction in which the cost of $J_{SLP}(\bar{\mu})$ decreases without bound. Howver, this is a constradiction, since J_{RLP} is lower bounded. So, we can rewrite $J_{SLP}(\mu)$ in the following manner:

$$\begin{aligned} J_{SLP}(\bar{\mu}) = & \min \sum_{i=1}^{m} \bar{c}(i) v(i), \\ v(i) \geq & \bar{r}(i, \mathbf{u}) + \lambda \sum_{\mathbf{x} \in \mathcal{S}_{\mathrm{i}}} h^i_{\mathbf{u}}(\mathbf{x}) \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) v(j), \forall \mathbf{u} \in \bar{\mathcal{U}}_i, i = 1, \cdots, m, \end{aligned} \tag{3.17}$$

where, $\mathbf{u} \in \bar{\mathcal{U}}_i$ if $\bar{\mu}^i_{\mathbf{u}}(\mathbf{x}) > 0$ for some $\mathbf{x} \in \mathcal{S}_{\mathrm{i}}$ and

$$h^i_{\mathbf{u}}(\mathbf{x}) = \frac{\bar{\mu}^i_{\mathbf{u}}(\mathbf{x})}{\sum_{\mathbf{x}\in\mathcal{S}_{\mathrm{i}}} \bar{\mu}^i_{\mathbf{u}}(\mathbf{x})},$$

and

$$\bar{r}(i, \mathbf{u}) = \sum_{\mathbf{x}\in\mathcal{S}_{\mathrm{i}}} h^i_{\mathbf{u}}(\mathbf{x}) r(\mathbf{x}, \mathbf{u}), \tag{3.18}$$

One may interpret the term, $h^i_{\mathbf{u}}(\mathbf{x})$, as the probability of picking the state $\mathbf{x}$ from the partition $\mathcal{S}_{\mathrm{i}}$. By inspection, we see that $J_{SLP}(\bar{\mu})$ is indeed the exact LP corresponding to an MDP (defined over the partitions) with immediate reward at partition i given by $\bar{r}(i, \mathbf{u})$ and transition probability between partitions i and j given by,

$$\bar{p}_{i,j}(\mathbf{u}) = \begin{cases} \sum_{\mathbf{x}\in\mathcal{S}_{\mathrm{i}}} h^i_{\mathbf{u}}(\mathbf{x}) \sum_{j=1}^{m} \sum_{\mathbf{y}\in\mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}), & \text{if } \mathbf{u} \in \bar{\mathcal{U}}_i, \\ 0, & \text{otherwise.} \end{cases} \tag{3.19}$$

So, the optimal solution to $J_{SLP}(\bar{\mu})$ is the optimal value function associated with the same underlying reduced order MDP.

Remark 1. *If one consider the sub-optimal dual variables, $\mu^i_{\mathbf{u}}(\mathbf{x}) = \frac{1}{|\mathcal{S}_i|}, \forall \mathbf{x} \in \mathcal{S}_i, \forall \mathbf{u}$, then solving the corresponding surrogate dual, $SLP(\mu)$, to obtain an approximate value function, would result in the so-called "hard aggregation" method (see Sec. 4 of [2]).*

Furthermore, in the following theorem, we will show that there exists $\bar{\mu}$ such that $\bar{\mu}^i_{\mathbf{u}}(\mathbf{x}) = 1$ only at a certain $\mathbf{x} \in \mathcal{S}_{\mathrm{i}}$ and $\mathbf{u} \in \mathcal{U}_{\mathrm{i}}$ for each i.

Theorem 3. *For the original MDP, if a $\mathcal{GP}$ is given, then for each partition, $\mathcal{S}_{\mathrm{i}}$, there exists $\bar{\mu}$, $\mathbf{x}^* \in \mathcal{S}_{\mathrm{i}}$ and $\mathbf{u}^* \in \mathcal{U}_{\mathrm{i}}$ such that such that $\bar{\mu}^i_{\mathbf{u}^*}(\mathbf{x}^*) = 1$ and $\bar{\mu}^i_{\mathbf{u}}(\mathbf{x}) = 0$*

for any other $\mathbf{x} \in \mathcal{S}_\mathrm{i}$ *and* $\mathbf{u} \in \mathcal{U}_\mathrm{i}$ *where* $\mathbf{x} \neq \mathbf{x}^*$ *and* $\mathbf{u} \neq \mathbf{u}^*$.

Proof. Let us define sets H_i for $i = 1, \cdots, m$ as following: for each i,

$$H_i = \mathcal{S}_\mathrm{i} \times \mathcal{U}_\mathrm{i} = \{a = (\mathbf{x}, \mathbf{u}) | \, \mathbf{x} \in \mathcal{S}_\mathrm{i}, \mathbf{u} \in \mathcal{U}_\mathrm{i}\}. \tag{3.20}$$

A set H_i is composed of all possible combinations of a state picked from partition $\mathcal{S}_\mathrm{i}$ and an action chosen from $\mathcal{U}_\mathrm{i}$. Let us consider an MDP whose state space is $\mathcal{S}' = \{1, 2, \cdots, m\}$, and available action set is H_i for each state i, reward function is $r'(i, a)$,and transition probability from state i to j under influence of action a is $p'_{i,j}(a)$. Note that an action on this MDP also requires one to select a state $\mathbf{x}$ of ELP for each state in $\mathcal{S}'$, and the corresponding $\mathbf{u}$ of ELP. Then the exact LP corresponding to this MDP is given as follows:

$$J'_{ELP} = \min \bar{c}^T w \tag{3.21}$$

$$w(i) \geq r'(i, a) + \lambda \sum_{j=1}^{m} p'_{i,j}(a) w(j), \; \forall a \in H_i, \forall i \in \mathcal{S}'. \tag{3.22}$$

Let the solution to J'_{ELP} be w^*, then because it is an MDP,

$$w^*(i) = r'(i, a^*) + \lambda \sum_{j=1}^{m} p'_{i,j}(a^*) w^*(j), \forall i,$$

where a^* is the optimal policy for the MDP as following,

$$a^*(i) = \underset{a \in H_i}{\operatorname{argmax}} [r'(i, a) + \lambda \sum_{j=1}^{m} p'_{i,j}(a) w^*(j)].$$

For any $a \in H_i$ and $i \in \mathcal{S}'$, there are corresponding reward function and transition

probability from the original MDP as follows:

$$r'(i,a) = r(\mathbf{x},\mathbf{u}), \tag{3.23}$$

and transition probability from state i to j

$$p'_{i,j}(a) = \sum_{\mathbf{y}\in\mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}). \tag{3.24}$$

With these terms, we can rewrite J'_{ELP} as following:

$$J'_{ELP} = \min \bar{c}^T w \tag{3.25}$$

$$w(i) \geq r(\mathbf{x},\mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y}\in\mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) w(j),\ \forall\, \mathbf{x} \in \mathcal{S}_{\mathrm{i}}, \forall\, \mathbf{u} \in \mathcal{U}_{\mathrm{i}}, i = 1, 2, \cdots, m. \tag{3.26}$$

Interestingly, this ELP is exactly same as RRLP, Eq. (3.4) and (3.5) implying $w^* = v^*$. Moreover, $J_{RLP} = J'_{ELP}$. We can consider another MDP whose state space is composed of chosen states $x^*(i), i = 1, \cdots, m$, and its ELP is the following;

$$J''_{ELP} = \min \bar{c}^T w \tag{3.27}$$

$$w(i) \geq r(\mathbf{x}^*(i),\mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y}\in\mathcal{S}_j} p_{\mathbf{x}^*(i),\mathbf{y}}(\mathbf{u}) w(j),\ \forall\, \mathbf{u} \in \mathcal{U}_{\mathrm{i}}, i = 1, 2, \cdots, m. \tag{3.28}$$

The solution to J''_{ELP} is the same as RLP, because $a^*(i) = (\mathbf{x}^*(i), \mathbf{u}^*(i))$ is the optimal

policy for MDP($\mathcal{S}'$),

$$w^*(i) = r'(i,a^*) + \lambda \sum_{j=1}^{m} p'_{i,j}(a^*) w^*(j), \forall i, \tag{3.29}$$

$$= r(\mathbf{x}^*, \mathbf{u}^*) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x}^*,\mathbf{y}}(\mathbf{u}^*) w^*(j), \forall i, \tag{3.30}$$

$$\geq r(\mathbf{x}^*, \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x}^*,\mathbf{y}}(\mathbf{u}) w^*(j), \forall \mathbf{u} \in \mathcal{U}_{\mathrm{i}}, \forall i. \tag{3.31}$$

Hence, $J''_{ELP} = J'_{ELP} = J_{RLP}$. Moreover, from Eq. (3.30), J''_{ELP} can be equivalently rewritten as follows:

$$J''_{ELP} = \min \bar{c}^T w \tag{3.32}$$

$$w(i) \geq r(\mathbf{x}^*(i), \mathbf{u}^*) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x}^*(i),\mathbf{y}}(\mathbf{u}^*) w(j), \ i = 1, 2, \cdots, m. \tag{3.33}$$

This LP is a surrogate LP with μ satisfying $\bar{\mu}^i_{\mathbf{u}^*}(\mathbf{x}^*) = 1$ and $\bar{\mu}^i_{\mathbf{u}}(\mathbf{x}) = 0$ for any other $\mathbf{x} \in \mathcal{S}_{\mathrm{i}}$ and $\mathbf{u} \in \mathcal{U}_{\mathrm{i}}$ where $\mathbf{x} \neq \mathbf{x}^*$ and $\mathbf{u} \neq \mathbf{u}^*$. Since $J_{RLP} = J''_{ELP}$, it follows that $J_{RLP} = J_{SLP}(\bar{\mu}) = J_{SDP} = J_{LD}$ and $\bar{\mu}$ is dual optimal. □

Recall that the RLP deals with a smaller number of variables, m, but the number of constraints is of the same order as the ELP of the original MDP. So, solving the RLP is no less difficult than solving the original ELP! However, an LP with a large number of constraints can be solved, if there is a computationally efficient scheme to identify a linear inequality that separates a non-optimal solution from an optimal one [12]. Otherwise, one has to resort to heuristics or settle for an approximate solution to the RLP. Heuristic methods include aggregation of constraints, sub-sampling of constraints [6], constraint generation methods [11, 13]. Other than these approaches, we reformulate the RLP to an ELP of an MDP whose state space is $\{1, 2, \cdots, m\}$,

available action set for each i is $H_i = \mathcal{S}_\mathrm{i} \times \mathcal{U}_\mathrm{i}$, during the proof of Theorem 3. Since it is an MDP, we can use general methods for the Dynamic program, including the value iteration and policy iteration. However, solving J'_{ELP} also requires expensive computation, because there are $|\mathcal{S}_\mathrm{i}| \times |\mathcal{U}_\mathrm{i}|$ number of available actions for each $i \in \mathcal{S}'$. We will provide some useful properties of the perimeter surveillance problem to decrease the amount of computation in the next section.

3.3 Lower Bounding Linear Programming

A simple but conservative lower bound for the value function is given by: $\underline{V}(\mathbf{x}) = \frac{\min_{\mathbf{y},\mathbf{u}} r(\mathbf{y},\mathbf{u})}{1-\lambda}, \forall\, \mathbf{x} \in \mathcal{S}$ for the following reason: For any stationary policy π,

$$V^* \geq [I - \lambda P_\pi]^{-1} R_\pi \geq [I - \lambda P_\pi]^{-1} r_{\min} = \frac{r_{\min}}{1-\lambda} \mathbb{1} = \underline{V},\ \forall \pi \in \Pi, \tag{3.34}$$

where $r_{\min} = \min_{\mathbf{y},\mathbf{u}} r(\mathbf{y},\mathbf{u})$. Eq. (3.34) indicates that if one were to pick a policy $\pi \in \Pi$, then a value function associated with the policy will be a lower bound to the value function of the MDP. Let us call the value function associated with policy π as the performance value function, V_π, of the policy π. Then

$$V_\pi = [I - \lambda P_\pi]^{-1} R_\pi, \tag{3.35}$$

and so V_π satisfies the following:

$$V_\pi = R_\pi + \lambda P_\pi V_\pi. \tag{3.36}$$

It is also difficult to compute V_π if the dimension, $|\mathcal{S}|$, is large. It is therefore reasonable to compute a lower bound for V_π, which is, in turn, a lower bound for the value function V^*.

There is a well-known policy so-called "greedy" policy in the literature. For any appriximate value function $\tilde{V}$, one can construct a sub-optimal greedy stationary policy according to:

$$\tilde{\pi}(\mathbf{x}) = \underset{\mathbf{u}}{\operatorname{argmax}}\{r(\mathbf{x},\mathbf{u}) + \lambda \sum_{\mathbf{y}\in\mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u})\tilde{V}(\mathbf{y})\},\ \forall\, \mathbf{x} \in \mathcal{S}\,. \tag{3.37}$$

Let us define the improvement in value function,

$$\alpha(\mathbf{x}) := r(\mathbf{x},\tilde{\pi}(\mathbf{x})) + \lambda \sum_{\mathbf{y}\in\mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\tilde{\pi}(\mathbf{x}))\tilde{V}(\mathbf{y}) - \tilde{V}(\mathbf{x}).$$

Note that there is no improvement, i.e., $\alpha = 0$, when $\tilde{V} = V^*$. The expected discounted one-step reward, $V_{\tilde{\pi}}$, corresponding to the sub-optimal policy $\tilde{\pi}$, satisfies the following bound ([25]):

$$\tilde{V}(\mathbf{x}) + \frac{1}{1-\lambda}\min_{\mathbf{y}\in\mathcal{S}}\alpha(\mathbf{y}) \le V_{\tilde{\pi}}(\mathbf{x}) \le V^*(\mathbf{x}),\ \forall\, \mathbf{x} \in \mathcal{S}\,.$$

In our experience, the lower bound to the optimal value function provided by the quantity $\tilde{V}(\mathbf{x}) + \frac{1}{1-\lambda}\min_{\mathbf{y}\in\mathcal{S}}\alpha(\mathbf{y})$ is very conservative. If $\tilde{V}$ is close to V^*, then $V_{\tilde{\pi}}$ could be a tight lower bound. However, computation of $V_{\tilde{\pi}}$ involves solving a linear system of equations of size $|\,\mathcal{S}\,|$, which would be expensive for a large state-space.

Alternative methods for computing a lower bound to the value function are desirable. In this respect, the optimal solution of following non-linear program (NLP)

can be used to compute a lower bound:

$$J_{NLP} = \min \sum_{i=1}^{m} \bar{c}(i)w(i), \tag{3.38}$$

$$w(i) \geq \min_{\mathbf{x}\in\mathcal{S}_\mathrm{i}} \{r(\mathbf{x},\mathbf{u}) + \lambda \sum_{j=1}^{m} \mathbf{p}_{\mathbf{x},j}(\mathbf{u})w(j)\},\ \forall\, \mathbf{u} \in \mathcal{U}_\mathrm{i}, i = 1, \cdots, m. \tag{3.39}$$

While it is not readily apparent why the optimal solution of NLP should be a lower bound to the value function, it can be addressed in two steps using the following lemmas. The first lemma shows that an optimal solution exists and the second lemma shows that it is a lower bound to the value function.

Lemma 6. *There exists a solution w^* to the NLP given by (3.38) and (3.39) that is optimal for every $\bar{c} \geq 0$.*

Proof. One can observe that the feasible set of RRLP is a subset of the feasible set of NLP; since RRLP is feasible, NLP is also feasible.

Every solution of NLP is lower bounded by $\frac{r_{min}}{1-\lambda}\mathbb{1}$. This follows from the following reasons:

- The constraint (3.39) is a disjunction and one can enumerate all the underlying LPs that constitute this disjunction.
- The feasible set of the NLPs is the union of the feasible sets of the underlying LPs.
- Since the underlying LPs are *finite* in number, and since every feasible solution to the underlying LP is lower bounded by $\frac{r_{min}}{1-\lambda}\mathbb{1}$, it follows that every solution of NLP is also lower bounded by $\frac{r_{min}}{1-\lambda}\mathbb{1}$.

As in Lemma 1, one can observe that if w_1, w_2 are feasible solutions of NLP, then their componentwise minimum, $\min\{w_1, w_2\}$, also satisfies the constraint (3.39) of

NLP. Hence, the componentwise minimum of all the feasible solutions of NLP exists because every solution is lower bounded; let this solution be w^*. If w is a feasible solution, then $w \geq w^*$ by construction. Since w^* is feasible and $\bar{c} \geq 0$, it follows that $\bar{c}^T w \geq \bar{c}^T w^*$ implying the optimality of w^*. Since the construction of w^* did not depend on $\bar{c}$, w^* is optimal for every $\bar{c} \geq 0$. □

The following lemma establishes that w^* is a lower bound to V^*:

Lemma 7. *Let $W(\mathbf{x}) := w^*(i), \forall\, \mathbf{x} \in \mathcal{S}_i$, then W is a lower bound to the solution to the ELP, V^*. That is, $V^*(\mathbf{x}) \geq W(\mathbf{x}), \forall\, \mathbf{x} \in \mathcal{S}$.*

Proof. Let us define a vector v^* such that $v^*(i) = \min_{\mathbf{x} \in \mathcal{S}_i} V^*(\mathbf{x})$, Then if v^* is feasible to the NLP, then the claim should hold. Since V^* is the solution to the ELP, it satisfies the follows;

$$\begin{aligned} V^*(\mathbf{x}) \geq & r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V^*(\mathbf{y}), \;\; \forall\, \mathbf{u} \in \mathcal{U}_{\mathbf{x}}, \forall\, \mathbf{x} \in \mathcal{S}, \\ \geq & r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) v^*(j), \;\; \forall\, \mathbf{u} \in \mathcal{U}_{\mathbf{x}}, \forall\, \mathbf{x} \in \mathcal{S}\,. \end{aligned}$$

Then for each partition $\mathcal{S}_i$, the following should hold;

$$v^*(i) = \min_{\mathbf{x} \in \mathcal{S}_i} V^*(\mathbf{x}) \geq \min_{\mathbf{x} \in \mathcal{S}_i} [r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) v^*(j)], \;\; \forall\, \mathbf{u} \in \mathcal{U}_{\mathrm{i}}, i = 1, \cdots, m.$$

This means that v^* is feasible to the NLP, and so $v^*(i) \geq w^*(i), \forall i$, from Lemma 6, as w^* is the componentwise minimum of all feasible solutions to NLP. Since $V^*(\mathbf{x}) \geq v^*(i) \geq w^*(i) = W(\mathbf{x}), \forall\, \mathbf{x} \in \mathcal{S}_i$, W is a lower bound to V^*. □

3.3.1 Iterative Linear Programming for Lower Bounds

Although the solution to NLP is a lower bound to the value function, solving NLP is not simple due to its non-linear constraints. So we provide an iterative method for a lower bound. The basic concept of the method is similar to the policy iteration. PI updates its policy in each iteration until it converges, however, in our approach, we find and update a set of states in each iteration. The iteration selects m states from m partitions, one state from one partition. Let us define a set $\mathcal{Q}$ such that $\mathcal{Q} := \prod_{i=1}^{m} \mathcal{S}_i$, then an element of set $\mathcal{Q}$ is a m-tuple vector and the iteration chooses one element of the set until its convergence.

Algorithm 3 State Iteration

1: Initialize $k \leftarrow 0$ and pick $h^k \in \mathcal{Q}$ arbitrarily.
2: Solve the following LP;

$$\begin{aligned} J_k &= \min \bar{c}^T w^k \\ w^k(i) &\geq r(h^k(i), \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{h^k(i),\mathbf{y}}(\mathbf{u}) w^k(j), \ \forall \mathbf{u} \in \mathcal{U}_\mathrm{i}, \forall i. \end{aligned} \tag{3.40}$$

3: Find z^k such that

$$z^k(i) = \min_{\mathbf{x} \in \mathcal{S}_i} \max_{\mathbf{u} \in \mathcal{U}_i} \left[r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) w^k(j) \right], \tag{3.41}$$

and choose a new set of states h^{k+1} from the following equation;

$$h^{k+1}(i) = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{S}_\mathrm{i}} \max_{\mathbf{u} \in \mathcal{U}_\mathrm{i}} \left[r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) w^k(j) \right].$$

4: If $w^k = z^k$, stop; else, set $k \leftarrow k + 1$ and go to Step 2.

Lemma 8. *Algorithm 3 terminates in finite number of iterations, and its solution is a lower bound to the value function.*

Once one chooses h^k, the LP, Eq. (3.40), becomes an ELP for a MDP whose state space is $\{1, 2, \cdots, m\}$, one-step reward of state i under influece of action $\mathbf{u}$ is $r(h^k(i), \mathbf{u})$, and its state transition probability from i to j is $\mathbf{p}_{i,j}(\mathbf{u}) = \sum_{\mathbf{y} \in \mathcal{S}_j} p_{h^k(i),\mathbf{y}}(\mathbf{u})$. This allows us to exploit the properites of Bellman's equation.

Proof. For any iteration k, $w^k(i) \geq z^k(i), \forall i$, because w^k is a solution to Eq. (3.40), it satisfies the following;

$$w^k(i) = \max_{\mathbf{u} \in \mathcal{U}_i} \left[r(h^k(i), \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{h^k(i),\mathbf{y}}(\mathbf{u}) w^k(j) \right], \tag{3.42}$$

and from the definition of z^k, Eq. (3.41). So if $w^k \neq z^k$, then there must exist at least one state i such that $w^k(i) > z^k(i)$.

At the next step, $k+1$, the solution to the LP is as following;

$$\begin{aligned} w^{k+1}(i) =& \max_{\mathbf{u} \in \mathcal{U}_i} \left[r(h^{k+1}(i), \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{h^{k+1}(i),\mathbf{y}}(\mathbf{u}) w^{k+1}(j) \right] && (3.43) \\ =& \mathcal{T}_{h^{k+1}} w^{k+1}(i), && (3.44) \end{aligned}$$

where $\mathcal{T}_{h^{k+1}}$ is a DP operator associating with h^{k+1}. From the definition of h^{k+1}, z^k can be rewritten as following;

$$\begin{aligned} z^k(i) =& \max_{\mathbf{u} \in \mathcal{U}_i} \left[r(h^{k+1}(i), \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{h^{k+1}(i),\mathbf{y}}(\mathbf{u}) w^k(j) \right], && (3.45) \\ =& \mathcal{T}_{h^{k+1}} w^k(i). && (3.46) \end{aligned}$$

Since $w^k(i) \geq z^k(i), \forall i$ and $\mathcal{T}_{h^{k+1}}$ is a DP operator, $\mathcal{T}_{h^{k+1}} w^k(i) = z^k(i) \geq \mathcal{T}_{h^{k+1}} z^k(i)$, which means,

$$z^k(i) \geq \mathcal{T}_{h^{k+1}} z^k(i) \geq \mathcal{T}^2_{h^{k+1}} z^k(i) \geq \cdots \geq w^{k+1}(i),$$

and

$$w^k(i) \geq z^k(i) \geq w^{k+1}(i).$$

Hence, if $w^k \neq z^k$ and so there exist some states such that $w^k(i) > z^k(i)$, then the value function for those states will strictly decrease at the next iteration,

$$w^k(i) > w^{k+1}(i).$$

Secondly, assume that the algorithm stops after K iterations. We will show that $w^K = w^*$ where w^* is the solution to the NLP, Eq. (3.38) and (3.39). By construction, $w^k, k = 0, 1, \cdots, K$ is feasible to the NLP. So, $w^k(i) \geq w^*(i), i = 1, \cdots, m, k = 0, 1, \cdots, K$. If w^* is the solution to the NLP, then it satisfies the following;

$$w^*(i) = \max_{\mathbf{u} \in \mathcal{U}_\mathrm{i}} \left[r(h^*(i), \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{h^*(i), \mathbf{y}}(\mathbf{u}) w^*(j) \right],$$

where

$$h^*(i) = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{S}_\mathrm{i}} \max_{\mathbf{u} \in \mathcal{U}_\mathrm{i}} \left[r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y} \in \mathcal{S}_j} p_{\mathbf{x}, \mathbf{y}}(\mathbf{u}) w^*(j) \right].$$

From the monotonicity,

$$
\begin{aligned}
w^K(i) \geq & \max_{\mathbf{u}\in\mathcal{U}_\mathrm{i}} \left[r(h^*(i),\mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y}\in\mathcal{S}_j} p_{h^*(i),\mathbf{y}}(\mathbf{u}) w^K(j) \right], \\
\geq & \min_{\mathbf{x}\in\mathcal{S}_\mathrm{i}} \max_{\mathbf{u}\in\mathcal{U}_\mathrm{i}} \left[r(\mathbf{x},\mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y}\in\mathcal{S}_j} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) w^K(j) \right], \\
= & z^K(i)
\end{aligned}
$$

Since $w^K = z^K$, the following equality holds

$$
w^K(i) = \max_{\mathbf{u}\in\mathcal{U}_\mathrm{i}} \left[r(h^*(i),\mathbf{u}) + \lambda \sum_{j=1}^{m} \sum_{\mathbf{y}\in\mathcal{S}_j} p_{h^*(i),\mathbf{y}}(\mathbf{u}) w^K(j) \right],
$$

and it means w^K is a fixed point. Hence, $w^* = w^K$. □

This algorithm will compute w^* in a finite number of iterations. The solution of LP in **Step 2** is computationally tractable as the number of variables and constraints are only of the order of the number of partitions. **Step 3** hides the complexity - if $\mathcal{S}_\mathrm{i}$ is of low cardinality, this is computationally tractable. Otherwise, one must exploit structure in the problem to compute w^*.

4. APPLICATION TO PERIMETER SURVEILLANCE PROBLEM

The perimeter surveillance problem arose from the Cooperative Operations in Urban Terrain (COUNTER) project at AFRL [10]. In this problem, there is a perimeter which must be monitored by a collection of UAVs. Along the perimeter, there are n_s alert stations equipped with Unattended Ground Sensors (UGSs) which detect intrusions or incursions into the perimeter. For the sake of simplicity, we assume that incursions into the perimeter can only occur at the stations. An incursion could be a nuisance (false alarm) or a real threat. The UGS raise an alarm or an alert whenever there is an incursion. The camera equipped UAV responds to an alert by flying to the alert site and loitering there, while a remotely located operator steers the gimballed camera looking for the source of the alarm. Here the operator serves the role of a classifier or a sensor, i.e., the operator must determine, from the video information, whether the intrusion is a nuisance or a threat. For details on the perimeter alert surveillance problem and the variants thereof, we refer the reader to the authors' prior work [3, 4, 18, 16]. Figure 4.1 shows a typical scenario, where there are 4 alert stations with the UAV at a station (location 0) with an alert. The decision problem we solve is the following: Given that the probability of arriving an alert at a station, what is the optimal time a UAV should spend at a station before resuming its patrol? We associate an information gain with a UAV loitering and servicing an alert and we model this gain as a monotonically increasing function of the loiter/dwell time T.

4.1 Problem Formulation

We discretize the perimeter surveillance control problem spatially and temporally; nodes on the perimeter partition it uniformly. The distance between adjacent nodes

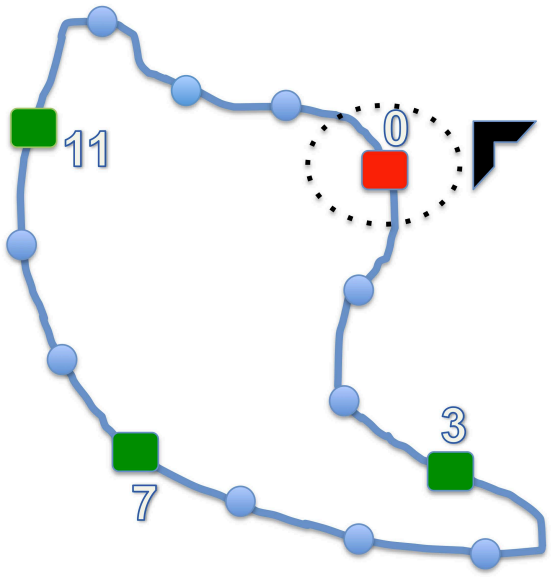

Figure 4.1: Perimeter surveillance scenario with UAV loitering at alert station.

on the perimeter is of unit length and the time taken by a robot to traverse between two adjacent nodes is a unit of time. Let the real vectors $\mathbf{x}_r(t), \mathbf{u}(t)$ denote the states of n_r robots in the collection and their control actions respectively at time t. Let a real vector $\mathbf{x}_s(t)$ denote the states associated with the n_s UGS locations. Further, let $\mathbf{d}(t) \in \{0,1\}^{n_s}$ denote the vector of disturbances (incursions) occuring at the n_s UGS locations respectively. We intentionally leave out a precise definition of the states $\mathbf{x}_r$ and $\mathbf{x}_s$, to allow for greater generality and to accommodate application needs, as they arise later in the article. As an example, one may include the location of a UAV, its direction of travel around the perimeter in the definition of $\mathbf{x}_r$, and the amount of time they spend (or dwell) servicing an alert at the UGS location, while $\mathbf{x}_s$ may contain the delays associated with UGS stations. The control actions of the robots at time t are captured by the vector $\mathbf{u}(t)$; a sample control action indicates whether a robot should dwell at its current location or continue in the same direction or reverse. The

disturbance $\mathbf{d}(t)$ can take any of the possible L values, namely $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_L$ with corresponding probabilities $p_1, \ldots, p_L$; these probabilities are assumed to be known a priori. The number of possible values the disturbance $\mathbf{d}(t)$ can take depends on the model of incursion processes; for example, if at most one incursion is allowed at any time across the n_s stations, then $L = n_s + 1$; if, on the other hand, incursions can occur simultaneously at all the stations, then $L = 2^{n_s}$.

Let the evolution of states $\mathbf{x}_r$ and $\mathbf{x}_s$ be governed by the state transiton equations:

$$\mathbf{x}_r(t+1) = f_r(\mathbf{x}_r(t), \mathbf{u}(t)), \tag{4.1}$$

$$\mathbf{x}_s(t+1) = f_s(\mathbf{x}_r(t), \mathbf{x}_s(t), \mathbf{u}(t), \mathbf{d}(t)), \tag{4.2}$$

where f_r and f_s are suitably defined vector fields. For the sake of notational convenience, let the state of the system $\mathbf{x}(t) := (\mathbf{x}_r(t), \mathbf{x}_s(t))$. The evolution equations (4.1) and (4.2) can be combined as:

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)), \tag{4.3}$$

for the augmented vector field f. Additionally, there may be constraints on the state and control input, of the form:

$$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \leq 0, \quad \forall t \geq 0, \tag{4.4}$$

which model the allowable control actions of the robots. For example, the state of an UGS can only be altered by the action of a robot that has spent a pre-specified amount of time in its neighborhood.

Let $\mathcal{S}_r, \mathcal{S}_s$ represent the set of all possible *discrete* states of robots and stations respectively. Let $\mathcal{S} = \mathcal{S}_r \times \mathcal{S}_s$ be the Cartesian product of the sets $\mathcal{S}_r$ and $\mathcal{S}_s$

and denote the set of all possible states of the system. Let $r(\mathbf{x}, \mathbf{u})$ denote the one-step payoff/ reward associated with the state $\mathbf{x}$ and the control input $\mathbf{u}$. Let $\mathcal{U}_{\mathbf{x}}$ denote the set of control actions for state $\mathbf{x}$. Let $\mathcal{U}$ be the set of all possible control actions. We focus our attention on stationary policies, $\pi \in \Pi$, where $\pi \in \Pi$ maps $\mathcal{S}$ into $\mathcal{U}$, i.e., $\mathbf{u} = \mathbf{u}_\pi(\mathbf{x}) \in \mathcal{U}_{\mathbf{x}}$. Consider the stochastic optimization problem: for a specified discount factor, $\lambda \in [0, 1)$, find a stationary policy, π, such that the following objective is maximized:

$$V^*(\mathbf{x}^0) := \max_{\pi \in \Pi} \mathbf{E}\left[\sum_{t=0}^{\infty} \lambda^t r(\mathbf{x}(t), \mathbf{u}_\pi(\mathbf{x}(t)))|\mathbf{x}(0) = \mathbf{x}^0\right], \tag{4.5}$$

where Π is the set of all possible stationary policies. We make the following standard assumptions about the finiteness of the states and control actions:

- **Assumption 1:** The robots are identical. Let a set of allowed control actions for robot i at state $\mathbf{x}$ be $\mathcal{A}^i_{\mathbf{x}}$, and assume $\mathcal{A}^i_{\mathbf{x}}$ is finite, then a set of available actions for state $\mathbf{x}$ is $\mathcal{U}_{\mathbf{x}} = \prod_{i=1}^{n_r} \mathcal{A}^i_{\mathbf{x}}$ for each $\mathbf{x} \in \mathcal{S}$.

- **Assumption 2:** Since the problem has been discretized, the perimeter is of finite length and since the disturbances and control decisions are finite, the sets $\mathcal{S}_r$ and $\mathcal{S}_s$ are also finite. Hence, the state space $\mathcal{S}$ of the system is finite.

As we discussed in Section 2, the value function V^* from (4.5) satisfies the following Bellman equation:

$$V^*(\mathbf{x}) = \max_{\mathbf{u} \in \mathcal{U}_{\mathbf{x}}} \left\{ r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{\mathbf{y} \in \mathcal{S}} p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) V^*(\mathbf{y}), \right\} \ \forall \mathbf{x} \in \mathcal{S}, \tag{4.6}$$

or we can write it using the evolution equation (4.3) in the following:

$$V^*(\mathbf{x}) = \max_{\mathbf{u}\in\mathcal{U}_\mathbf{x}} \left\{ r(\mathbf{x},\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V^*(f(\mathbf{x},\mathbf{u},\mathbf{d}_l)), \right\} \ \forall \mathbf{x} \in \mathcal{S}, \tag{4.7}$$

where $p_l = \text{Prob}\{\mathbf{d} = \mathbf{d}_l\}$. Both equations are equivalent, and both or either will be used for explanations in the following sections. For a modest size problem involving 2 robots and 8 stations, the value of $|\mathcal{S}|$ can be upwards of 180 billion! For this reason, the conventional techniques to solving Bellman's equation, such as value and policy iteration, are unsuitable. So, our methodologies from previous sections may be proper to this application.

We use $\geq$ to compare approximations of value functions; in particular, if $\mathbf{V}_1 \geq \mathbf{V}_2$, we mean that every component of $\mathbf{V}_1 - \mathbf{V}_2 \geq 0$. The partial ordering of states may not obey the same relationship. To distinguish this difference, we use $\succeq$ to compare two states whenever it is possible.

4.2 Results Exploiting the Structure of the Perimeter Surveillance Problem

Motivated by the perimeter surveillance application, we make the following additional assumptions about the structure of the system:

- **Assumption 3:** For a given $\mathbf{x}_r, \mathbf{d}, \mathbf{u}$, the function f_s is monotone in $\mathbf{x}_s$, i.e., if $\mathbf{x}_s \geq \mathbf{z}_s$ then $f_s(\mathbf{x}_r, \mathbf{x}_s, \mathbf{u}, \mathbf{d}) \geq f_s(\mathbf{x}_r, \mathbf{z}_s, \mathbf{u}, \mathbf{d})$. In the perimeter surveillance application, we treat the delay in servicing an alert at a location as the state $\mathbf{x}_s$. In this case, the delay increases monotonically until it is reset by the action of the robots.

- **Assumption 4:** We assume the following structure for the one-step payoff function:

$$r(\mathbf{x}, \mathbf{u}) = \psi_r(\mathbf{x}_r, \mathbf{u}) - \psi_s(\mathbf{x}_s), \tag{4.8}$$

  where $\psi_r : \mathcal{S}_r \times \mathcal{U} \to \Re$, and $\psi_s : \mathcal{S}_s \to \Re_+$. The function ψ_r is a monotone function of $\mathbf{x}_r$ for every $\mathbf{u}$, while the function ψ_s is a monotone function of $\mathbf{x}_s$. This structure is motivated by the following consideration: the information gained by robots depends on how long they dwell at a station, while there is a penalty associated with tardiness in servicing alerts at other locations. So, ψ_r can be considered as the information gain from actions of robots, and ψ_s is a penalty function for the tardy responses.

- **Assumption 5:** Each robot knows the complete state, $\mathbf{x}_s$, of all stations. While this may not be realistic, we make this assumption in order to avoid complexities that arise from incomplete information.

Assumption 3 motivates the following partial ordering relationship amongst the states:

Definition : (Partial Ordering of States) Let $\mathbf{x}, \mathbf{y} \in \mathcal{S}$. Then $\mathbf{x} \succeq \mathbf{y}$ if $\mathbf{x}_r = \mathbf{y}_r$, $\mathbf{x}_s - \mathbf{y}_s \geq 0$.

Assumptions 3 and 4 also suggest a partitioning scheme of a second kind, wherein the one-step reward across all the states in the partition is the same:

Definition: (Constant Reward Partitioning Scheme) A general partitioning scheme $\mathcal{CRP} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m\}$ of cardinality m is a Constant Reward Partitioning

Scheme if for every $\mathbf{x}, \mathbf{y} \in \mathcal{S}_i,\ i = 1, 2, \ldots, m$,

$$\mathbf{x}_r = \mathbf{y}_r,\ \ \psi_s(\mathbf{x}_s) = \psi_s(\mathbf{y}_s),\ \ \forall \mathbf{x}, \mathbf{y} \in \mathcal{S}_k,\ k = 1, \ldots, m.$$

We will refer to the subsets $\mathcal{S}_k,\ k = 1, 2, \ldots, m$, as *constant reward partitions* or simply partitions, when the context is clear. It is also clear that for any pair of states $\mathbf{x}, \mathbf{y}$ in a constant reward partition, the one-step reward is the same, for the same control action $\mathbf{u}$:

$$r(\mathbf{x}, \mathbf{u}) = \psi_r(\mathbf{x}_r, \mathbf{u}) - \psi_s(\mathbf{x}_s) = \psi_r(\mathbf{y}_r, \mathbf{u}) - \psi_s(\mathbf{y}_s) = r(\mathbf{y}, \mathbf{u}).$$

We denote the reward corresponding to any state in a constant reward partition $\mathcal{S}_i$ as $r_i(\mathbf{u})$. Since it is a general partitioning, $\mathcal{U}_{\mathbf{x}} = \mathcal{U}_{\mathbf{y}} =: \mathcal{U}_i, \forall\, \mathbf{x}, \mathbf{y} \in \mathcal{S}_{\mathrm{i}}$.

The perimeter surveillance also allows for the partial ordering of constant reward partitions in the following way:

Definition: (Partial Ordering of Partitions) Given a constant reward partitioning scheme $\{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$, we define $\mathcal{S}_i \succeq \mathcal{S}_j$ if

1. for every $\mathbf{x} \in \mathcal{S}_i$, there is a $\mathbf{z} \in \mathcal{S}_j$ such that $\mathbf{x} \succeq \mathbf{z}$; moreover, there is no $\mathbf{s} \in \mathcal{S}_j$ such that $\mathbf{s} \succeq \mathbf{x}$, and

2. for every $\mathbf{z} \in \mathcal{S}_j$, there is a $\mathbf{x} \in \mathcal{S}_i$ such that $\mathbf{x} \succeq \mathbf{z}$; moreover, there is no $\mathbf{s} \in \mathcal{S}_i$ such that $\mathbf{z} \succeq \mathbf{s}$.

We require another definition to ensure that ordering of states induces a consistent ordering of partitions.

Definition: (Consistent Partitioning) A constant reward partitioning scheme $\{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$ of order m is consistent if $\mathbf{x}, \mathbf{z} \in \mathcal{S}$ and $\mathbf{x} \succeq \mathbf{z}$ implies one of the following conditions holds:

(i) there exist distinct partitions $\mathcal{S}_i, \mathcal{S}_j$ such that $\mathbf{x} \in \mathcal{S}_i$, $\mathbf{z} \in \mathcal{S}_j$ and $\mathcal{S}_i \succeq \mathcal{S}_j$, or

(ii) there exists a partition $\mathcal{S}_i$ such that $\mathbf{x}, \mathbf{z} \in \mathcal{S}_i$.

We then refer to $\mathcal{S}_1, \ldots, \mathcal{S}_m$ as consistent partitions. The perimeter surveillance problem allows for the existence of a consistent partitioning scheme. By way of notation, we define $\bar{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}_l) = j$ if $\mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}_l) \in \mathcal{S}_j$.

Definition : (Maximal/Minimal State) For a given partition $\mathcal{S}_\mathrm{i}$,

- State $\mathbf{x} \in \mathcal{S}_\mathrm{i}$ is a maximal state, if there is no $\mathbf{y} \in \mathcal{S}_\mathrm{i}$ such that $\mathbf{y} \succeq \mathbf{x}$. Let a set $\bar{\mathcal{S}}_i$ denote a set of all maximal states of $\mathcal{S}_\mathrm{i}$.

- State $\mathbf{x} \in \mathcal{S}_\mathrm{i}$ is a minimal state, if there is no $\mathbf{y} \in \mathcal{S}_\mathrm{i}$ such that $\mathbf{x} \succeq \mathbf{y}$. Let a set $\underline{\mathcal{S}}_i$ denote a set of all minimal states of $\mathcal{S}_\mathrm{i}$.

Then, $\bar{\mathcal{S}}_i \succeq \underline{\mathcal{S}}_i$.

We are now ready to state the following theorem which simplifies the computation of upper and lower bounds:

Theorem 4. *Let $\mathcal{S}_1, \ldots, \mathcal{S}_m$ be consistent partitions. Let $\mathbf{c}$ be any positive vector. Then,*

1. *an upper bound $\bar{\mathbf{V}}_{ub}$ of $\mathbf{V}^*$ may be computed from the following upper bounding LP referred to as* **UBLP***:*

$$\begin{aligned} J_u &= \min \sum_{j=1}^{m} [\sum_{\mathbf{x} \in \mathcal{S}_i} c(\mathbf{x})] w(i), \\ w(i) &\geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l w(\bar{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}_l)), \ \forall \mathbf{z} \in \underline{\mathcal{S}}_i, \forall \mathbf{u} \in \mathcal{U}_\mathrm{i}, i = 1, \cdots, m, \\ w(j) &\geq w(i), \ \ \forall \mathcal{S}_i \succeq \mathcal{S}_j. \end{aligned}$$

Let $\bar{w}$ denote the solution to **UBLP** *and $\bar{V}_{ub}(\mathbf{x}) = \bar{w}(i)$ for all $\mathbf{x} \in \mathcal{S}_i$.*

2. *A lower bound* $\bar{\mathbf{V}}_{lb}$ *of* $\mathbf{V}^*$ *can be computed from the following lower bounding non-linear LP referred to as* **LBNLP***:*

$$\begin{aligned} J_l &= \min \sum_{j=1}^{m} [\sum_{\mathbf{x}\in\mathcal{S}_i} c(\mathbf{x})] w(i), \text{ subject to,} \\ w(i) &\geq \min_{\mathbf{x}\in\bar{\mathcal{S}}_i} \left[r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l w(\bar{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}_l)) \right], \ \forall \mathbf{u} \in \mathcal{U}_\mathrm{i}, i = 1, \cdots, m. \end{aligned}$$

Let $\underline{w}$ *denote the solution to* **LBNLP** *and* $\bar{V}_{lb}(\mathbf{x}) = \underline{w}(i)$ *for every* $\mathbf{x} \in \mathcal{S}_i$*.*

The proof of this theorem requires the following lemma:

Lemma 9. *Let* $\mathbf{x}, \mathbf{z}$ *correspond to two different initial states of the system described by Eq.* (4.1) *and* (4.2)*, and satisfying assumptions (1) through (5). Let the corresponding trajectories subject to the same input* $\mathbf{u}(t)$ *and disturbance* $\mathbf{d}(t)$ *be respectively* $\mathbf{x}(t)$ *and* $\mathbf{z}(t)$*. If* $\mathbf{x} \succeq \mathbf{z}$*, then*

(i) $\mathbf{x}(t) \succeq \mathbf{z}(t)$ *for all* $t \geq 0$*,*

(ii) $V^*(\mathbf{x}) \leq V^*(\mathbf{z})$ *and*

(iii) $V^*(\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)) \leq V^*(\mathbf{f}(\mathbf{z}(t), \mathbf{u}(t), \mathbf{d}(t)), \ \ t \geq 0.$

Proof. The proof of (i) is by induction. At $t = 0$, it is readily true from the hypothesis. It suffices to show that if $\mathbf{x}(t) \succeq \mathbf{z}(t)$, then $\mathbf{x}(t+1) \succeq \mathbf{z}(t+1)$; however, this readily follows from evolution equations (4.1), (4.2) and the assumption (3).

For the same sequence of inputs $\mathbf{u}(t), \mathbf{d}(t)$, from (i) and assumption (4), we can infer that $r(\mathbf{x}(t), \mathbf{u}(t)) \leq r(\mathbf{z}(t), \mathbf{u}(t))$ for $t \geq 0$. Hence, for the same sequence of inputs $\mathbf{u}(t), \mathbf{d}(t)$, the total discounted reward associated with the initial state $\mathbf{x}$ is no more than the initial state $\mathbf{z}$. Taking expectation over all the disturbances and maximizing over all the control actions, one readily obtains $V^*(\mathbf{x}) \leq V^*(\mathbf{z})$.

From part (i), since $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)) = \mathbf{x}(t+1) \succeq \mathbf{z}(t+1) = \mathbf{f}(\mathbf{z}(t), \mathbf{u}(t), \mathbf{d}(t))$, it follows that $V^*(\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)) \leq V^*(\mathbf{f}(\mathbf{z}(t), \mathbf{u}(t), \mathbf{d}(t)), \quad t \geq 0.$ □

We now provide the proof of Theorem 4.

Proof of Theorem 4. Since $r(\mathbf{x}, \mathbf{u}) = r_i(\mathbf{u})$ for every $\mathbf{x} \in \mathcal{S}_i$, and $p_{\mathbf{x},\mathbf{y}}(\mathbf{u}) = p_l$ if $\mathbf{y} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}_l)$, one may express the Bellman inequalities as:

$$V(\mathbf{x}) \geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V(f(\mathbf{x}, \mathbf{u}, \mathbf{d}_l)), \qquad \forall \mathbf{x} \in \mathcal{S}_i, \forall \mathbf{u} \in \mathcal{U}_{\mathrm{i}}, \forall i. \tag{4.9}$$

1. Consider the following minimization problem:

$$\begin{aligned} J_s &= \min \mathbf{c} \cdot \mathbf{V}, && (4.10)\\ V(\mathbf{x}) &\geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V(f(\mathbf{x}, \mathbf{u}, \mathbf{d}_l)), \quad \forall \mathbf{x} \in \mathcal{S}_i, \forall \mathbf{u} \in \mathcal{U}_{\mathrm{i}}, \forall i, && (4.11)\\ V(\mathbf{y}) &\geq V(\mathbf{x}), \quad \forall \mathbf{x} \succeq \mathbf{y}. && (4.12) \end{aligned}$$

   Given any $\mathbf{x} \in \mathcal{S}_i$, $\mathbf{x} \succeq \mathbf{y}$ for some $\mathbf{y} \in \underline{\mathcal{S}}_i$. If $\mathbf{V}$ satisfies the following strengthened version of Bellman inequality for all $\mathbf{x} \in \mathcal{S}_i$, $\mathbf{y} \in \underline{\mathcal{S}}_i$,

$$V(\mathbf{x}) \geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V(f(\mathbf{y}, \mathbf{u}, \mathbf{d}_l)),$$

   by part (iii) of Lemma 9, it would automatically satisfy Bellman's inequalities:

$$V(\mathbf{x}) \geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V(f(\mathbf{x}, \mathbf{u}, \mathbf{d}_l)), \quad \forall \mathbf{x} \in \mathcal{S}_i.$$

Consider the following intermediate LP (**ILP**):

$$
\begin{aligned}
&J = \min \mathbf{c} \cdot \mathbf{V}, \\
&V(\mathbf{x}) \geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V(f(\mathbf{y}, \mathbf{u}, \mathbf{d}_l)), \quad \forall \mathbf{x} \in \mathcal{S}_i,\ \forall \mathbf{y} \in \mathcal{S}_i, \forall\, \mathbf{u} \in \mathcal{U}_{\mathrm{i}}, \forall i, \\
&V(\mathbf{y}) \geq V(\mathbf{x}), \quad \forall \mathbf{x} \succeq \mathbf{y}, \quad \mathbf{x}, \mathbf{y} \in \mathcal{S}.
\end{aligned}
$$

Since Bellman inequalities of the ELP are satisfied by the optimal solution, $\bar{\mathbf{V}}$ of ILP, it automatically upper bounds $\mathbf{V}^*$. The LPs **ILP** and **UBLP** have the same optimal value and the optimal solution of one can be used to construct the optimal solution of the other in the following way:

- Since $\bar{V}_{ub}(\mathbf{x}) = \bar{w}(i)$ for all $\mathbf{x} \in \mathcal{S}_i$, we can see that $\bar{\mathbf{V}}_{ub}$ readily satisfies the first set of constraints of **ILP**. Since $\mathcal{S}_1, \ldots, \mathcal{S}_m$ are consistent partitions, the last set of constraints is met: if $\mathbf{x} \succeq \mathbf{y}$, then either $\mathbf{x}, \mathbf{y}$ belong to the same partition or belong to different partitions; in the former case, the last constraint is readily met. In the latter case, there exist partitions $\mathcal{S}_i \ni \mathbf{x}, \quad \mathcal{S}_j \ni \mathbf{y}$ such that $\mathcal{S}_i \succeq \mathcal{S}_j$. Since $\bar{w}(j) \geq \bar{w}(i)$ for all $\mathcal{S}_i \succeq \mathcal{S}_j$, it follows that $\bar{V}_{ub}(\mathbf{y}) = \bar{w}(j) \geq \bar{w}(i) = \bar{V}_{ub}(\mathbf{x})$. Since $\bar{\mathbf{V}}$ is optimal, $\mathbf{c} \cdot \bar{\mathbf{V}} \leq \mathbf{c} \cdot \bar{\mathbf{V}}_{ub} = \sum_{i=1}^{m} [\sum_{\mathbf{x} \in \mathcal{S}_i} c(\mathbf{x})] \bar{w}(i)$. By Lemma 1, we additionally have $\bar{\mathbf{V}}_{ub} \geq \bar{\mathbf{V}}$.
- By the same token, if we set $w(i) = \bar{V}(\mathbf{x}),\ \forall x \in \mathcal{S}_i$, we see that it is feasible for **UBLP**. Hence, $\sum_{i=1}^{m} [\sum_{\mathbf{x} \in \mathcal{S}_i} c(\mathbf{x})] \bar{w}(i) \leq \sum_{i=1}^{m} [\sum_{\mathbf{x} \in \mathcal{S}_i} c(\mathbf{x})] w(i) = \mathbf{c} \cdot \bar{\mathbf{V}}$.

Since $\mathbf{c} > 0$ and $\bar{\mathbf{V}}_{ub} \geq \bar{\mathbf{V}}$, $\mathbf{c} \cdot \bar{\mathbf{V}} = \mathbf{c} \cdot \bar{\mathbf{V}}_{ub} \Rightarrow \bar{\mathbf{V}} = \bar{\mathbf{V}}_{ub}$. Since $\bar{\mathbf{V}} \geq \mathbf{V}^*$, it follows that $\bar{\mathbf{V}}_{ub} \geq \mathbf{V}^*$.

2. For every $\mathbf{x} \in \mathcal{S}_i$, $\mathbf{y} \succeq \mathbf{x}$ for some $\mathbf{y} \in \bar{\mathcal{S}}_i$; it follows from part (ii) of Lemma

9 that $V^*(\mathbf{x}) \geq V^*(\mathbf{y})$. Let us define $w^*(i) := \min_{\mathbf{x}\in\mathcal{S}_i} V^*(\mathbf{x})$ and $x^*(i) := \operatorname{argmin}_{\mathbf{x}\in\bar{\mathcal{S}}_i} V^*(\mathbf{x})$, then, by the definition of $\bar{\mathcal{S}}_i$, $w^*(i) = \min_{\mathbf{y}\in\bar{\mathcal{S}}_i} V^*(\mathbf{y})$, and $w^*(i) = V^*(x^*(i))$. Since $\mathbf{V}^*$ satisfies Bellman inequalities, it follows that

$$\begin{aligned} w^*(i) = \min_{\mathbf{x}\in\mathcal{S}_i} V^*(\mathbf{x}) &= V^*(x^*(i)) \\ &\geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V^*(f(x^*(i), \mathbf{u}, \mathbf{d}_l)), \quad \forall\, \mathbf{u} \in \mathcal{U}_{\mathrm{i}}, \\ &\geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l \min_{\mathbf{z}\in\mathcal{S}_{\bar{f}(x^*(i),\mathbf{u},\mathbf{d}_l)}} V^*(\mathbf{z}), \quad \forall\, \mathbf{u} \in \mathcal{U}_{\mathrm{i}}, \\ &\geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l w^*(\bar{f}(x^*(i), \mathbf{u}, \mathbf{d}_l)), \quad \forall\, \mathbf{u} \in \mathcal{U}_{\mathrm{i}}\,. \end{aligned}$$

Moreover, $x^*(i)$ is an element in $\bar{\mathcal{S}}_i$, so w^* is feasible to **LBNLP**. Hence, $w^* \geq \underline{w}$.

□

The subsequent remark concerns possible simplifications when there is symmetry in the problem. Symmetry induces equivalence classes of states.

Remark 2. *If $\mathcal{E}_i$ is an equivalence class, it is natural that $\mathcal{E}_i$ is subset of a consistent partition. Symmetry also implies that $V^*(\mathbf{x}) = V^*(\mathbf{z})$ for all $\mathbf{x}, \mathbf{z} \in \mathcal{E}_i$. These constraints may easily be accommodated in* **UBLP** *by setting $\mathbf{x} \succeq \mathbf{z}$ for all $\mathbf{x}, \mathbf{z} \in \mathcal{E}_i$, and in* **LBNLP** *by requiring $\mathcal{E}_i$ to be contained wholly in a consistent partition. Moreover, if $\mathbf{x}, \mathbf{z} \in \mathcal{E}_i$, we have $\bar{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}_l) = \bar{f}(\mathbf{z}, \mathbf{u}, \mathbf{d}_l)$ for every $\mathbf{u}$ and $\mathbf{d}_l$. Hence, the inequality constraints corresponding to states in an equivalence class of the form*

$$w(i) \;\geq\; r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l w(\bar{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}_l)), \;\; \forall i, \mathbf{u},$$

can be replaced by a single inequality constraints where $\mathbf{x}$ *is a representative of the equivalance class. Similarly, the NLP constraint of the form*

$$w(i) \quad \geq \min_{\mathbf{s}\in\bar{\mathcal{S}}_i}[r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l w(\bar{f}(\mathbf{s},\mathbf{u},\mathbf{d}_l))] \; \forall i, \mathbf{u},$$

simplifies in the following way: the minimization on the right hand side of the inequality will now only need to be carried out among representative states of an equivalence class contained in the set $\bar{\mathcal{S}}_i$.

Remark 3. *If for every* i, $\bar{\mathcal{S}}_i$ *is a singleton set, i.e.,* $\bar{\mathcal{S}}_i = \{\mathbf{z}_i\}$, *then* **LBNLP** *reduces to the following simplified LP (***LLP***):*

$$\begin{aligned} J_l &= \min \sum_{j=1}^{M} [\sum_{\mathbf{x}\in\mathcal{S}_i} c(\mathbf{x})] w(i),, \\ w(i) &\geq r_i(\mathbf{u}) + \lambda \sum_{l=1}^{L} p_l w(\bar{f}(\mathbf{z}_i,\mathbf{u},\mathbf{d}_l)), \quad \forall i, \mathbf{u}, \end{aligned}$$

and $\bar{V}_2(\mathbf{x}) = \bar{w}(i)$ *for every* $\mathbf{x} \in \mathcal{S}_i$. *Solving a LP is considerably simpler than solving a NLP and hence, there is an associated simplification in computing the lower bound.*

4.3 Perimeter Surveillance Problems

We will provide details about the perimeter surveillance problem for several different instances for illustration purposes. The patrolled perimeter is a simple closed curve with $N(\geq n_s)$ nodes which are (spatially) uniformly separated, of which n_s correspond to the alert stations. Let the n_s distinct station locations be elements of the set $\Omega \subset \{0,\ldots,N-1\}$. A typical scenario shown in Figure 4.1 has 15 nodes, of which, nodes $\{0,3,7,11\}$ correspond to the UGS. We will consider the elements of Ω ordered in increasing order so that we can label the stations as the 1^{st} station locates at node 0, the 2^{nd} one is at node 3, and so on. Let Ω' denote the index set of the

station nodes; $\Omega' = \{1, 2, \cdots, n_s\}$. Here, station locations 3, 7 and 11 have no alerts, and station location 0 has an alert being serviced by the loitering UAV. A perimeter is "symmetric" if number of nodes between two adjacent stations is idential, otherwise, "asymmetic" perimeter. In the example, number of nodes between station 0 and 11, 11 and 7, 7 and 3, is all 3. However, the one between station 0 and 3 is 2, So the perimeter is asymmetric. If the number of nodes between station 0 and 3 is also 3, then it will be a symmetric perimeter. In this dissertation, we consider only symmetric perimeter problems. However, even if the perimeter is asymmetric, one can exploit the structure of symmetric perimeters to achieve the bounds. They will loose the tightness between upper and lower bounds, but our claims will still hold.

If a UGS detects an incursion, an alert is raised at the location and communicated instantaneously to the robots. As we briefly mentioned before, the probability of alert arrivals depends on alerting process. We will consider two types of alerting processes as follows:

1. Single alert queue: There is a single queue where an alert arrives at the queue with probability p_α. After an alert is queued up, we assume it shows up arbitrarily at any one of the n_s stations (assuming choice of station is a uniformly distributed random variable). For this reason, only one alert can arrive at one of the n_s stations at any instant of time. Hence, there are n_s+1 possibilities for the value of the vector of alerts $\mathbf{d}(t) \in \{\mathbf{d}_0, \mathbf{d}_1, \cdots, \mathbf{d}_{n_s}\} =: \mathcal{D}$ where $\mathcal{D}$ is a set of all possible disturbance inputs, with the first one being that there is no alert at any station and the other n_s correspond to an alert at each of the n_s stations. Then $Prob\{\mathbf{d}(t) = \mathbf{d}_0\} = 1 - p_\alpha$ and $Prob\{\mathbf{d}(t) = \mathbf{d}_k\} = p_\alpha/n_s, k = 1, \cdots, n_s$.

2. n_s alert queues: Each station has an independent alert queue, then there can be n_s new alerts in the perimeter at a time. So $|\mathcal{D}| = 2^{n_s}$ and then probability

that k stations raise an alert at each time step is $p_\alpha^k(1-p_\alpha)^{n_s-k}$.

Consider a perimeter to be monitored with the aid of n_r identical robots. Let $a_k(t)$ denote the action of the k^{th} robot at time t, so that a control for the surveillance system at time t is a combination of actions of all robot, $\mathbf{u}(t) = (a_1(t), a_2(t), \ldots, a_{n_r}(t))$. The set of allowable actions for each robot is $\{1, 0, -1\}$ with $a_k(t) = 0$ if it dwells at its current location and equals 1 or -1 respectively if it moves counterclockwise (CCW) or clockwise (CW). The maximum number of allowable values of $\mathbf{u}$ is 3^{n_r}. We will restrict on the action of the robots such that a robot can only dwell at a UGS location. This is a valid restriction, because there is no benefit for the robot to dwell at a non-UGS location. So, the allowable actions at a non-UGS location for the robot is $\{-1, 1\}$. The disturbance input $\mathbf{d}(t)$ is an element of $\mathcal{D}$, and the set $\mathcal{D}$ depends on the type of alert queue given in the previous section. For notational conveniences, let $\delta(\cdot)$ denote the Kronecker delta function and $\bar{\delta}(\cdot) = 1 - \delta(\cdot)$.

At time instant t, for the k^{th} robot, let $\ell_k(t)$ be the position of the robot on the perimeter ($\ell_k \in \mathcal{N}$), $T_k(t)$ be the dwell time and $\tau_j(t)$ be the delay in servicing an alert at location $j \in \Omega$. The evolution equations may be expressed as:

$$\ell_k(t+1) = (\ell_k(t) + a_k(t)) \; mod \; N, \quad k = 1, \ldots, n_r, \tag{4.13}$$

$$T_k(t+1) = (T_k(t) + 1)\delta(a_k(t)), \quad k = 1, \ldots, n_r, \tag{4.14}$$

$$\tau_j\,(t+1) = h(\tau_j(t), \ell_1(t), a_1(t), \ldots, \ell_{n_r}(t), a_{n_r}(t), d_j(t)), \forall j \in \Omega, \tag{4.15}$$

where

$$h(\tau_j, \ell_1, a_1, \ldots, \ell_{n_r}, a_{n_r}, d_j) := \max\left\{(\tau_j + 1)\sigma\left(\tau_j(t)\right)\left[1 - \max_{i=1,\ldots,n_r}\left\{\delta(\ell_k - j)\delta(a_k)\right\}\right], d_j\right\}.$$

The state of the robots is given by $\mathbf{x}_r(t) = (\ell_1(t), T_1(t), \ell_2(t), T_2(t), \ldots, \ell_{n_r}(t), T_{n_r}(t))$

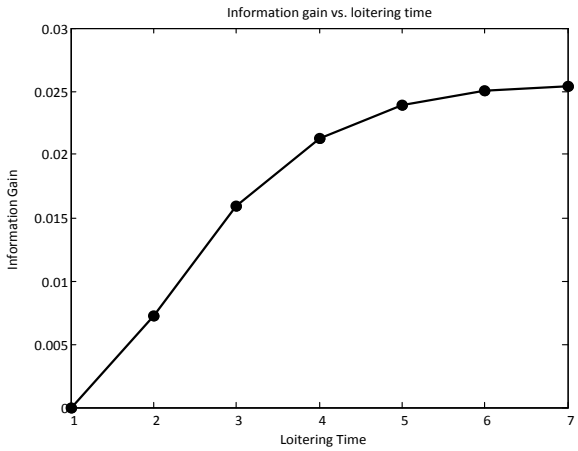

Figure 4.2: Information gain vs. dwell time

and $\mathbf{x}_s(t) = (\tau_j(t), j \in \Omega)$. Let $\mathbf{x} = (\mathbf{x}_r, \mathbf{x}_s)$, then again we can express the evolution equations compactly as:

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)).$$

To be consistent with the notation introduced earilier, we shall use $\mathcal{S}$ to denote the set of all system states and $\mathbf{x} \in \mathcal{S}$ to denote a particular state. Every state is unique on the state space $\mathcal{S}$, so we sometimes consider $\mathcal{S}$ is a set of indices of states such that if $\mathbf{x} \in \mathcal{S} = \{1, 2, \cdots, |\,\mathcal{S}\,|\}$, then $\mathbf{x}$ denotes a state and an index of the state.

Our objective is to find a suitable policy that simultaneously minimizes the service delay and maximizes the information gained upon loitering. The information gain, $\mathcal{I}$, which is based on an operator error model (further details about the information gain in [17]), is plotted as a function of dwell time in Fig. 4.2. We model the one-step payoff/ reward function as follows:

$$r(\mathbf{x},\mathbf{u}) = \sum_{k=1}^{n_r} \left[\mathcal{I}(T_{\mathbf{x}}^k + 1) - \mathcal{I}(T_{\mathbf{x}}^k)\right] \delta(a_k) - \rho \ \min\{\bar{\tau}_{\mathbf{x}}, \Gamma\}, \quad \forall\, \mathbf{u} \in \mathcal{U}_{\mathbf{x}}, \forall\, \mathbf{x} \in \mathcal{S}, \ (4.16)$$

where $T_{\mathbf{x}}^k$ is the dwell of robot k associated with state $\mathbf{x}$ and $\bar{\tau}_{\mathbf{x}} = \max_{j\in\Omega} \tau_{j,x}$ is the worst service delay (among all stations) associated with state $\mathbf{x}$. The parameter $\Gamma(>> 0)$ is a judiciously chosen maximum penalty. If we borrow the expression form the assumption 4, then $\psi_r(\mathbf{x}_r, \mathbf{u}) = \sum_{k=1}^{n_r} \left[\mathcal{I}(T_{\mathbf{x}}^k + 1) - \mathcal{I}(T_{\mathbf{x}}^k)\right] \delta(a_k)$ and $\psi_s(\mathbf{x}_s) = \rho \ \min\{\bar{\tau}_{\mathbf{x}}, \Gamma\}$. Assume that Eq. (4.16) satisfies the assumption 4. The positive parameter ρ is a constant weighing the incremental information gained upon loitering once more at the current location against the delay in servicing alerts at other stations. From the state definition, we can compute the total number of states in the MDP to be, if $n_r = 2$ with n_s alert queues,

$$|\mathcal{S}| = \underbrace{n_s(n_s-1)T_{\max}^2(\Gamma+1)^{n_s-2}}_{\text{Both robots dwell}} + \underbrace{2n_sNT_{\max}(\Gamma+1)^{n_s-1}}_{\text{One robot dwells}} + \underbrace{N^2(\Gamma+1)^{n_s}}_{\text{Neither dwells}}, \quad (4.17)$$

where $T_{\max}$ is the maximum loitering time; the robot does not stay more than $T_{\max}$ at a station. In a case of a single queue, the number of states is

$$\begin{aligned}
|\mathcal{S}| =& N^2 \sum_{j=0}^{n_s} \binom{n_s}{j} \left(j! \binom{\Gamma}{j} + \sum_{k=0}^{j-2} \binom{j}{k} \frac{(\Gamma-1)!}{(\Gamma-1-k)!} \right) \\
&+ 2n_sNT_{\max} \sum_{j=0}^{n_s-1} \binom{n_s-1}{j} \left(j! \binom{\Gamma}{j} + \sum_{k=0}^{j-2} \binom{j}{k} \frac{(\Gamma-1)!}{(\Gamma-1-k)!} \right) \\
&+ n_s(n_s-1)T_{\max}^2 \sum_{j=0}^{n_s-2} \binom{n_s-2}{j} \left(j! \binom{\Gamma}{j} + \sum_{k=0}^{j-2} \binom{j}{k} \frac{(\Gamma-1)!}{(\Gamma-1-k)!} \right). \quad (4.18)
\end{aligned}$$

Note that, in lieu of the reward function defintion (4.16), we do not keep track of delays beyond Γ and hence the state space $\mathcal{S}$ only includes states $\mathbf{x}$ with $\tau_j \leq$

$\Gamma, \forall j \in \Omega$ and so, is finite. We immediately see that the problem size is an n_s^{th} order polynomial in Γ and hence solving for the optimal value function and policy using exact dynamic programming methods are rendered intractable for practical values of Γ and n_s.

There is a natural partitioning of states; where no matter what the delays are at the other stations, the reward is the same, as long as the maximum delay and the dwell time of the robot at the station are the same. So, we aggregate all the states which have the same values for $\ell,\ T,\ \mathcal{A}_j,\ \forall j \in \Omega$ and $\bar{\tau} = \max_{j\in\Omega} \tau_j$, into one partition, where $\mathcal{A}_j$ indicates whether the j^{th} UGS has an alert and referred as to alert status of the j^{th} station; any two states $\mathbf{x}, \mathbf{y} \in \mathcal{S}_i$ implies $\mathbf{x}_r = \mathbf{y}_r$ and $\psi_s(\mathbf{x}_s) = \psi_s(\mathbf{y}_s)$. Moreover, this parititoning is a constant reward partition scheme, $\mathcal{CRP}$, and so we can apply the methodologies from the previous sections. By the aggregations, the number of partitions in the case of $n_r = 2$ with n_s alert queues can be shown to be,

$$\begin{aligned} m =& n_s(n_s-1)T_{\max}^2\left[\left(2^{n_s-2}-1\right)\Gamma+1\right] + 2n_sNT_{\max}\left[\left(2^{n_s-1}-1\right)\Gamma+1\right] \\ &+ N^2\left[\left(2^{n_s}-1\right)\Gamma+1\right] \end{aligned}$$

which is linear in Γ and hence considerably smaller than the total number of states (4.17). The number of states and partitions is shown in Table 4.1 for several different problem instances.

This partitioning scheme has the following properties: Let $n_i = \sum_{j=1}^{n_s} A_j$ for partition $\mathcal{S}_i$, then it denotes the number of alerts not serviced yet.

- Single alert queue case:
  - For each $\mathcal{S}_{\text{i}}$, the number of maximal states is $n_i!$, if $\bar{\tau}_{\mathbf{x}} < \Gamma$, and one, if

$\bar{\tau}_{\mathbf{x}} = \Gamma$.

- For each $\mathcal{S}_{\mathrm{i}}$, the number of minimal states is $n_i!$.

- n_s alert queue case:

  - For each $\mathcal{S}_{\mathrm{i}}$, there exists only one maximal state in the partition. $|\bar{S}_i| = 1$
  - For each $\mathcal{S}_{\mathrm{i}}$, the number of minimal states in the partition is n_i, which is the number of alerts in the current state.

Currently, we only consider symmetric perimters. As long as the perimeter is symmetric, one can reduce the number of partition one step further. In the governing equations, Eq. (4.13) to (4.15), $\ell_k(t), T_k(t)$ respectively denote the current location of the k^{th} robot and the time it has spent at its current location. Let $l_s(t)$ and $l_r(t)$ respectively denote the distance from the first robot to the nearest station and the second robot in the CCW direction. It is intuitive that states of the robots with the same l_s and l_r values are related by cyclic symmetry if the delays at stations are also correspondingly cylically permuted; in such a case, all states that can be transformed from each other by a cyclic permutations can be aggregated into a partition. The state of the two robots is given by $\mathbf{x}_r(t) = (l_s(t), l_r(t), T_1(t), T_2(t))$.

The governing equations for $n_r = 2$ case may be rewritten as:

$$l_s(t+1) = (l_s(t) + a_1(t)) \mod N/n_s, \tag{4.19}$$

$$l_r(t+1) = (l_r(t) + a_2(t)) \mod N, \tag{4.20}$$

$$T_k(t+1) = (T_k(t) + 1)\delta(a_k(t)), \quad k = 1, 2, \tag{4.21}$$

$$\tau_j\,(t+1) = h(\tau_j(t), l_1(t), a_1(t), l_2(t), a_2(t), d_j(t)), \forall j \in \Omega, \tag{4.22}$$

where,

$$h(\tau_j, \ell_1, a_1, \ell_2, a_2, d_j) := \max\left\{(\tau_j + 1)\sigma\left(\tau_j(t)\right)\left[1 - \max_{i=1,2}\{\delta(l_i - j)\delta(a_i)\}\right], d_j\right\}.$$

This new definition of states reduces the number of partitions approximately up to factor of n_s. The last column in Table 4.1 provides the number of new partitions. As we can see, the number of cyclic partitions is much smaller, and it still holds the properties of $\mathcal{CRP}$.

4.3.1 Numerical Results

In this part, we provide numerical results of several different problem instances supporting our claims and methodologies. First, we take a relatively simple problem so that we can compute the value function V^*. And then, we will move to a large problem.

4.3.1.1 Single robot, single alert queue problem

Consider a problem instance shown in the 4th row on Table 4.1; $n_s = 4, N = 8, T_{\max} = 5$, and $\Gamma = 15$. The other parameters were chosen to be: weighing factor, $\rho = .005$ and temporal discount factor, $\lambda = 0.9$. Based on experience, we chose the alert arrival rate $\alpha = \frac{1}{60}$. This reflects a rather low arrival rate where we expect 2 alerts to occur on average in the time taken by the UAV to complete an uninterrupted patrol around the perimeter. This problem includes only 100 thousand states, so the value iteration or the ELP is applicable to this problem. We utilized VI, and it took 48 iterations to converge to the optimal.

In Figure 4.3, we show results supporting the claim that for partially ordered states $x_1 \geq x_2$, the corresponding optimal value functions satisfy $V^*(x_1) \leq V^*(x_2)$. For this, we plot the value function V^* corresponding to states with alert status

# of Queues	n_r	n_s	N	$T_{\max}$	Γ	# of states	# of partitions	# of cyclic partitions
1	**1**	**4**	**4**	**2**	**6**	8,992	600	150
1	**1**	**4**	**8**	**2**	**6**	14,016	896	224
1	**1**	**4**	**4**	**5**	**15**	323,392	2,856	714
1	**1**	**4**	**8**	**5**	**15**	508,704	3,692	923
1	**1**	**6**	**12**	**5**	**15**	117,601,344	22,314	3,719
1	**1**	**8**	**16**	**5**	**15**	18,526,703,104	112,312	14,039
1	**2**	**4**	**8**	**5**	**15**	4,142,232	62,184	14,799
1	**2**	**5**	**10**	**5**	**15**	83,245,200	196,700	39,340
1	**2**	**6**	**12**	**5**	**15**	1,445,962,128	574,638	95,773
1	**2**	**7**	**14**	**5**	**15**	22,219,358,784	1,549,170	221,310
1	**2**	**8**	**16**	**5**	**15**	306,062,542,464	3,955,192	494,399
1	**2**	**9**	**18**	**5**	**15**	3,809,825,789,184	9,708,192	1,078,688
4	**1**	**4**	**4**	**2**	**6**	12,348	708	177
4	**1**	**4**	**8**	**2**	**6**	21,952	1,072	268
4	**1**	**4**	**4**	**5**	**15**	344,064	3,024	756
4	**1**	**4**	**8**	**5**	**15**	606,208	3,928	982
6	**1**	**6**	**12**	**5**	**15**	232,783,872	25,332	4,222
8	**1**	**8**	**16**	**5**	**15**	79,456,894,976	137,456	17,182
4	**2**	**4**	**8**	**5**	**15**	5,581,824	75,684	15,546
5	**2**	**5**	**10**	**5**	**15**	139,673,600	265,100	42,520
6	**2**	**6**	**12**	**5**	**15**	3,220,045,824	809,994	106,874
7	**2**	**7**	**14**	**5**	**15**	70,156,025,856	2,278,206	255,708
8	**2**	**8**	**16**	**5**	**15**	1,466,597,113,856	6,066,536	592,942
9	**2**	**9**	**18**	**5**	**15**	29,706,141,302,784	15,541,704	1,345,856

Table 4.1: Number of states and partitions for different instances of the surveillance problem.

$\mathcal{A}_{\mathbf{x}} = (1, 1, 0, 0)$ (alerts on station 1 and 2), dwell $d = 0$, the UAV located at one of station 1 ($\ell = 0$). The X- and Y-axis represent the service delay time of station 2 and 1, respectively. Each point in the plot represents the value function of a state. Arrows connecting the points denote the dominating relationship between two states. If x_1 dominates x_2 ($x_1 \geq x_2$), then $x_1 \rightarrow x_2$. Naturally, if $x_1 \geq x_2$ and $x_2 \geq x_3$, then $x_1 \geq x_3$. So, the plot shows only the closest dominating relationship between two states. We can observe that the value fucntion are non-decreasing in the direction of any arrow.

In Figure 4.4, we show results supporting the claim that for partially ordered partitions $\mathcal{S}_i \geq \mathcal{S}_j$, the corresponding optimal value functions satisfy $\min_{\mathbf{x}\in\mathcal{S}_i} V^*(\mathbf{x}) \leq \min_{\mathbf{y}\in\mathcal{S}_j} V^*(\mathbf{y})$. For this, we plot the value functions corresponding to states from above. The partially ordered partitions demarcated by the dotted grid lines in the X-axis are non-decreasing from left to right with maximum delay $\bar{\tau}$ varying from 2 to Γ. Within each partition, we plot the value function associated with every state in the partition and also the least value function in the partition shown as the green line. One can easily see that the claim above is satisfied.

Now, we shall consider the same example problem and show that the proposed approximate methodology is effective. For this, we compute the approximate value functions via the restricted LP formulation and the non-linear LP, and compare them with the optimal value function. In addition, we also compute the greedy sub-optimal policy corresponding to the approximate value function and compare it with the optimal policy in terms of the two performance metrics: alert service delay and information gained upon loitering. We aggregate the states in the example problem based on the reward function ($\mathcal{CRP}$). This results in $m = 923$ partitions, which is considerably smaller than the original number of states, $|\mathcal{S}|$. We solve both the $UBLP$ and $LBNLP$ formulations which give us the upper and lower bounds, v^* and

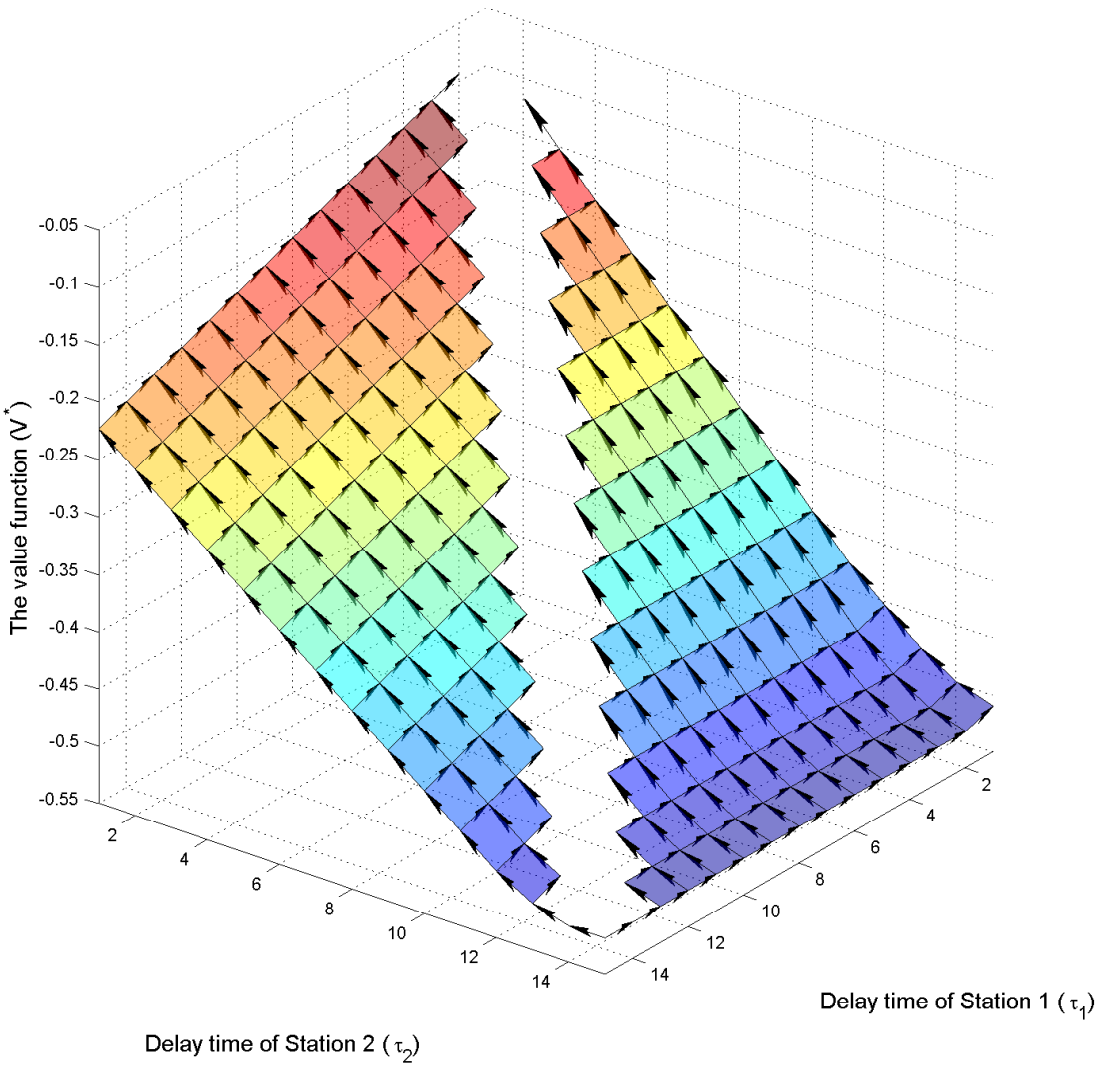

Figure 4.3: Monotonicity of the value function (states).

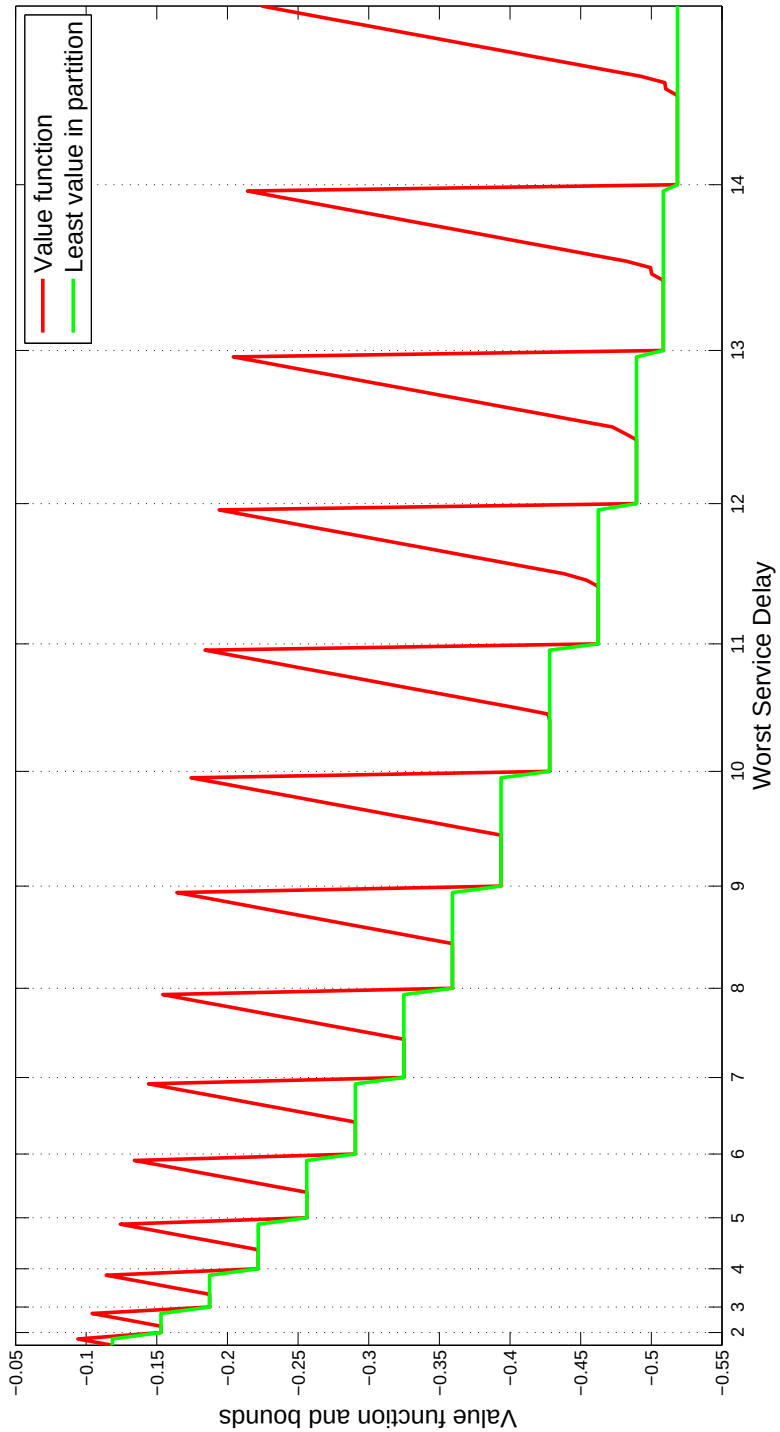

Figure 4.4: Monotonicity of the value function (partitions)

w^* respectively, to the optimal value function V^*; $v^*(i) = V_{ub}(\mathbf{x}), w^*(i) = V_{lb}(\mathbf{x}) \forall\, \mathbf{x} \in \mathcal{S}_\mathrm{i}$. Since we have the optimal value function for the example problem, we use it for comparison with the approximations. Note that for higher values of n_s and Γ, the problem would essentially become intractable and one would not have access to the optimal value function. Nevertheless, one can compute v^* and w^* and the difference between the two would give an estimate of the quality of the approximation.

Figure 4.5 shows the value function and its bounds, V_{ub} and V_{lb}. The upper bound V_{ub} is the solution to the $UBLP$, and the lower bound V_{lb} is the result from our iteration, Algorithm 3. From this figure, we can observe the boundness of the value function.

We give a representative sample of the approximation results by choosing all the states in partitions corresponding to alert status $\mathcal{A} = (1, 1, 1, 1)$ (all stations have alerts) and maximum delay $\bar{\tau} = 4$. Figure 4.6 compares the optimal value function V^* with the upper and lower bound approximate value functions, V_{ub} and V_{lb} for this subset of the state-space. Interestingly, we notice immediately that the lower bound appears to be tighter than the upper bound. Recall that our objective is to obtain a good sub-optimal policy and so, we consider the policy that is *greedy* with respect to V_{lb}:

$$\pi(\mathbf{x}) = \underset{\mathbf{u}}{\operatorname{argmax}} \left\{ r(\mathbf{x}, \mathbf{u}) + \lambda \sum_{l=1}^{L} p_l V_{lb}(f(\mathbf{x}, \mathbf{u}, d_l)) \right\}, \quad \forall\, \mathbf{x} \in \mathcal{S}\,. \tag{4.23}$$

To assess the quality of the sub-optimal policy, we also compute the expected discounted payoff, V_π that corresponds to the sub-optimal policy π, by solving the system of equations:

$$[I - \lambda P_\pi]\, V_\pi = R_\pi. \tag{4.24}$$

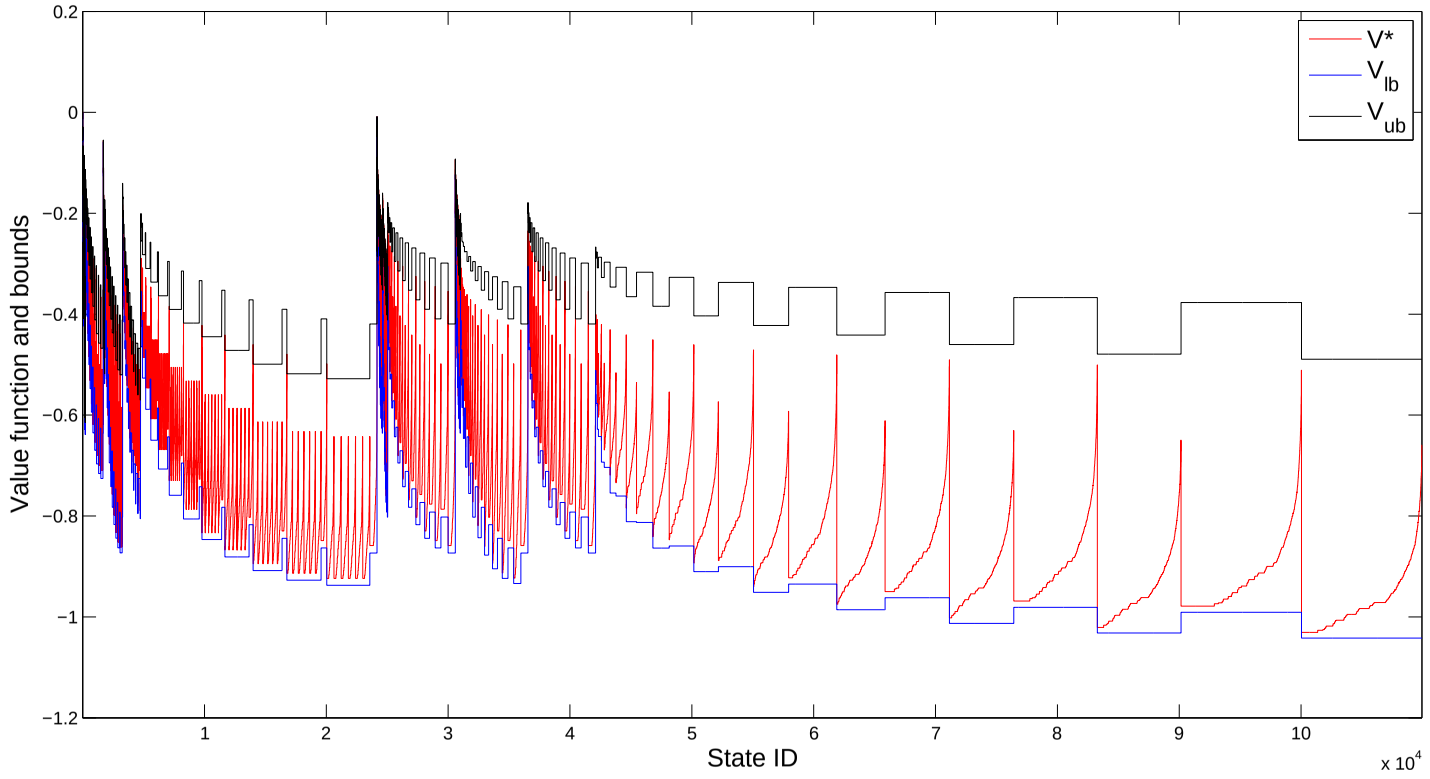

Figure 4.5: The value function and bounds: Single robot, single alert queue, 4 station, and symmetric perimeter problem

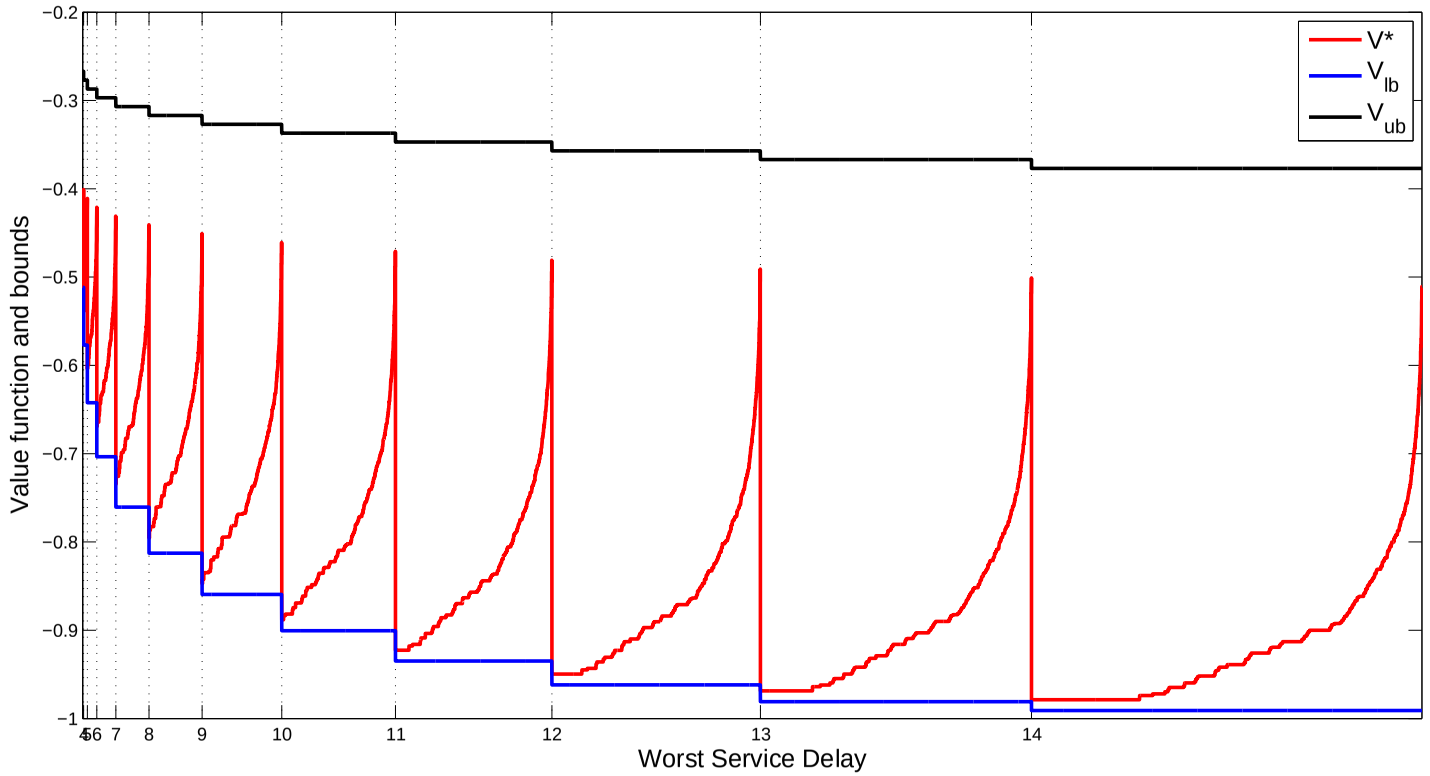

Figure 4.6: The value function and bounds (sampled): Single robot, single alert queue, 4 station, and symmetric perimeter problem

Since V_π corresponds to a sub-optimal policy and in lieu of the monotonicity property of the Bellman operator, the following inequalities hold:

$$V_{lb} \leq V_\pi \leq V^* \leq V_{ub}.$$

In Figure 4.7, we compare V_π with the optimal value function V^* and the lower bound V_{lb} for the sampled states and note that the approximation is quite good. One can consider the average percentage error between two value functions as an approximation rate. If V^* and V_π are available,

$$\%\text{Err}^* = \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in \mathcal{S}} \frac{V^*(\mathbf{x}) - V_\pi(\mathbf{x})}{|V_\pi(\mathbf{x})|}, \tag{4.25}$$

if not,

$$\%\text{Err} = \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in \mathcal{S}} \frac{V_{ub}(\mathbf{x}) - V_{lb}(\mathbf{x})}{|V_{lb}(\mathbf{x})|}. \tag{4.26}$$

The smaller percentage error means the better approximation. In this example problem, $\%\text{Err} = 55.54\%$ and $\%\text{Err}^* = 3.68\%$. So we can expect the sub-optimal policy is close to the optimal policy.

We performed Monte-Carlo simulations in order to test the effectiveness of the sub-optimal policy, π, to examine the following quantities of practical interest: (a) average dwell time, (b) average delay time in servicing an alert, (c) worst delay time, and (d) total information gained. We compare the performance of the sub-optimal policy π with that of the optimal strategy π^*, apparently π^* is available because the value funciton is given. To collect the performance statistics, Monte Carlo simulations run with alerts generated from a Poisson arrival stream with rate $\alpha = \frac{1}{60}$ over a 60000 time unit simulation window. Both the optimal and sub-optimal

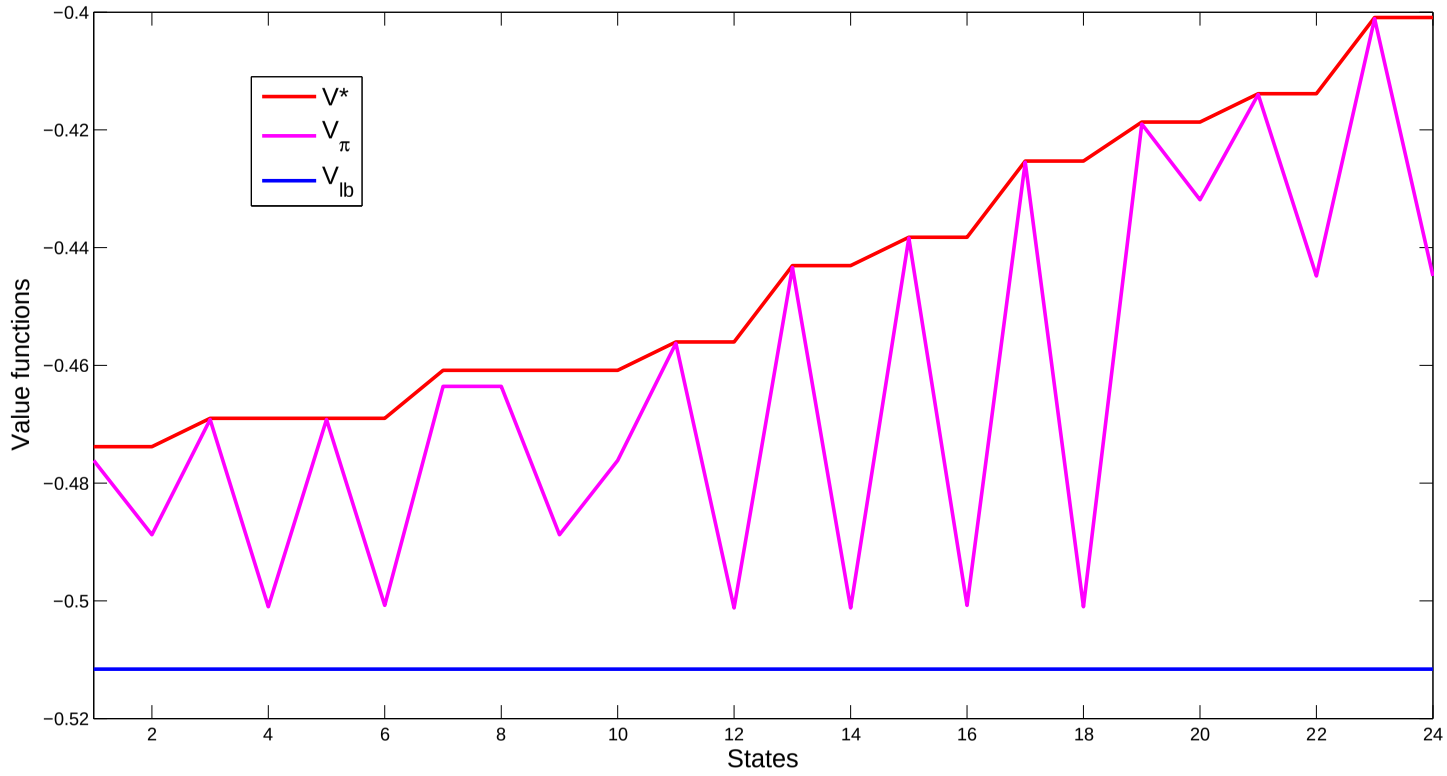

Figure 4.7: The value function and bounds (sampled): Single robot, single alert queue, 4 station, and symmetric perimeter problem

Policy	Mean dwell	Mean delay	Worst delay	Total Info. Gain
Optimal Policy (π^*)	4.4	3.27	8	3.032
Sub-optimal policy (π)	4.5	3.17	8	3.079

Table 4.2: Comparison of alert servicing performance between optimal and sub-optimal policies (single instance).

policies were tested against the same alert sequence. Figure 4.8 shows histogram plots for the service delay and the dwell time for all serviced alerts in the simulation run. The corresponding mean and worst case service delays and the mean dwell time are also shown in Table 4.2. We see that there is hardly any difference in terms of either metric between the optimal and the sub-optimal policies. This substantiates the claim that the aggregation approach gives us a sub-optimal policy that performs almost as well as the optimal policy itself.

This is to be expected, given that the value functions corresponding to the optimal and sub-optimal policies are close to each other (see Figure 4.7). Since the false alarm rate α is fairly low, we see from the right plot of Figure 4.8 that roughly 90% of the alerts were cleared within ten time steps. Also from the left plot of Figure 4.8, we see that maximum information was gained (5 loiters completed) on almost 90% of the serviced alerts. Table 4.2 shows the performance indices mentioned above from the simulations. The values are closed to each other as expected. According to the results, the performance of the sub-optimal policy is better than one of the optimal policy. However, it is due to a specific incursion instance. So we run the same simulation for 50 different incursion instances, and the results show in Table 4.3. In this table, the values are closer to each other, and the performance of the optimal policy is slightly better.

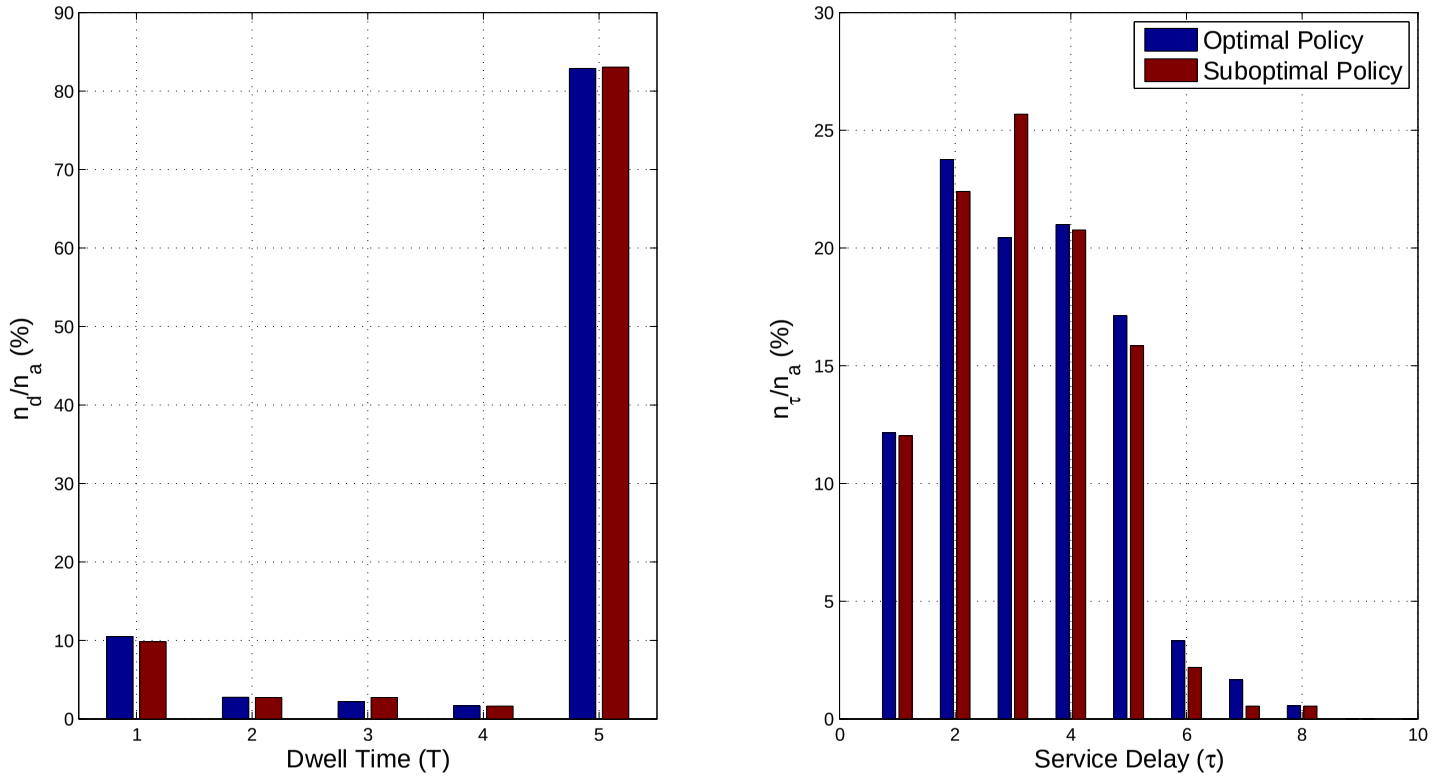

Figure 4.8: Comparison of service delay and number of loiters between optimal and sub-optimal policies (single instance).

Policy	Mean dwell	Mean delay	Worst delay	Total Info. Gain
Optimal Policy (π^*)	4.56	3.20	10	3.203
Sub-optimal policy (π)	4.56	3.18	11	3.206

Table 4.3: Comparison of alert servicing performance between optimal and sub-optimal policies (50 instance).

4.3.1.2 Two robots, multiple alert queue problem

If two UAVs are emplyeed for the task, then then the number of states will increase significantly as shown in Table 4.1. Moreover, as the number stations increases, the number of states is increasing dramatically. From the result of single robot problem, we observed that the value funciton is bounded by the solutions to **UBLP** and **LBNLP**. In this section, we take a surveillance problem with two UAVs and eight stations which originally involves 1.5 trillion states. We consider a problem with $n_r = 2$, $n_s = 8$, $N = 16$, $T_{\max} = 5$, $\Gamma = 15$, multiple alert queue, and $\Omega = \{0, 2, 4, 6, 8, 10, 12, 14\}$ which are symmetrically located. By adopting the same partitioning scheme as in the previous example, the number of partitions is 592,942. This number does not overwhelm us, but it implies that the average number of states in a partition is approximately 2.5 million. That means, for each partition, there are 2.5 million constraints in **UBLP** and **LBNLP**, unless we exploit the structure of the surveillance problem as given in the previous section.

The exact computation of value function (or the sub-optimal performance corresponding to any sub-optimal policy) for this instance of the problem is not tractable owing to the number of states. However, using the proposed methods, we computed the upper and lower bounds for the value function as shown in Figure 4.9. In this example, percentage error between upper and lower bounds is 69.7%. Theorem 3 assures us that if we were to choose a policy that is greedy with respect to the lower

bound, we are guaranteed a sub-optimal performance that exceeds the lower bound.

Based on our methodologies, we know that the value function $V^*(\mathbf{x})$ is laid in between $V_{ub}(\mathbf{x})$ and $V_{lb}(\mathbf{x})$ in Figure 4.9. However, it is not easy to show numerically. So we will introduce another value function, *empirical* value function, to show the boundness briefly later.

The Monte Carlo simulation run time was set to 60000 time units with three different alert rate, $\alpha = 1, 2$, or 6. For each α, we ran the simulation for 50 different incursion instances to see the performance of the policy properly. So, all values shown in this section are mean values of 50 results, except the worst delay time. We collected the data from the Monte-Carlo simulations and the results are shown in Figure 4.10 and 4.11, and Table 4.4. During the Monte Carlo simulation interval, let n_a, n_d and n_τ respectively represent the total number of alerts, number of alerts which were serviced with a dwell time T and number of alerts which have been serviced after a time delay of τ. If a policy is effective, then all the alerts will be serviced with reasonable service delays. The dwell time associated with servicing an alert can be anywhere from 1 to 5 time steps as shown in Figure 4.10. The plot shows the fraction of times an alert is serviced as a function of the dwell time. Among the alerts serviced by the robots when they employ the sub-optimal policy, a majority of the alerts were serviced with the maximum dwell time of 5 units, except the case of $\alpha = 6$. If the actual alert rate increases, the robots needs to keep moving to another station to service alerts after it clears an alert at the current station. So, most of alerts will be serviced only one unit time by the robots. The average dwell time associated with the sub-optimal policy can be seen from the Table 4.4.

TheFigure 4.11 provides the information concerning the time delay associated with servicing an alert. For example, when the robots employ the sub-optimal policy with $\alpha = 2$, the highest green bar shows the percentage of alerts, $\approx 30\%$, that were

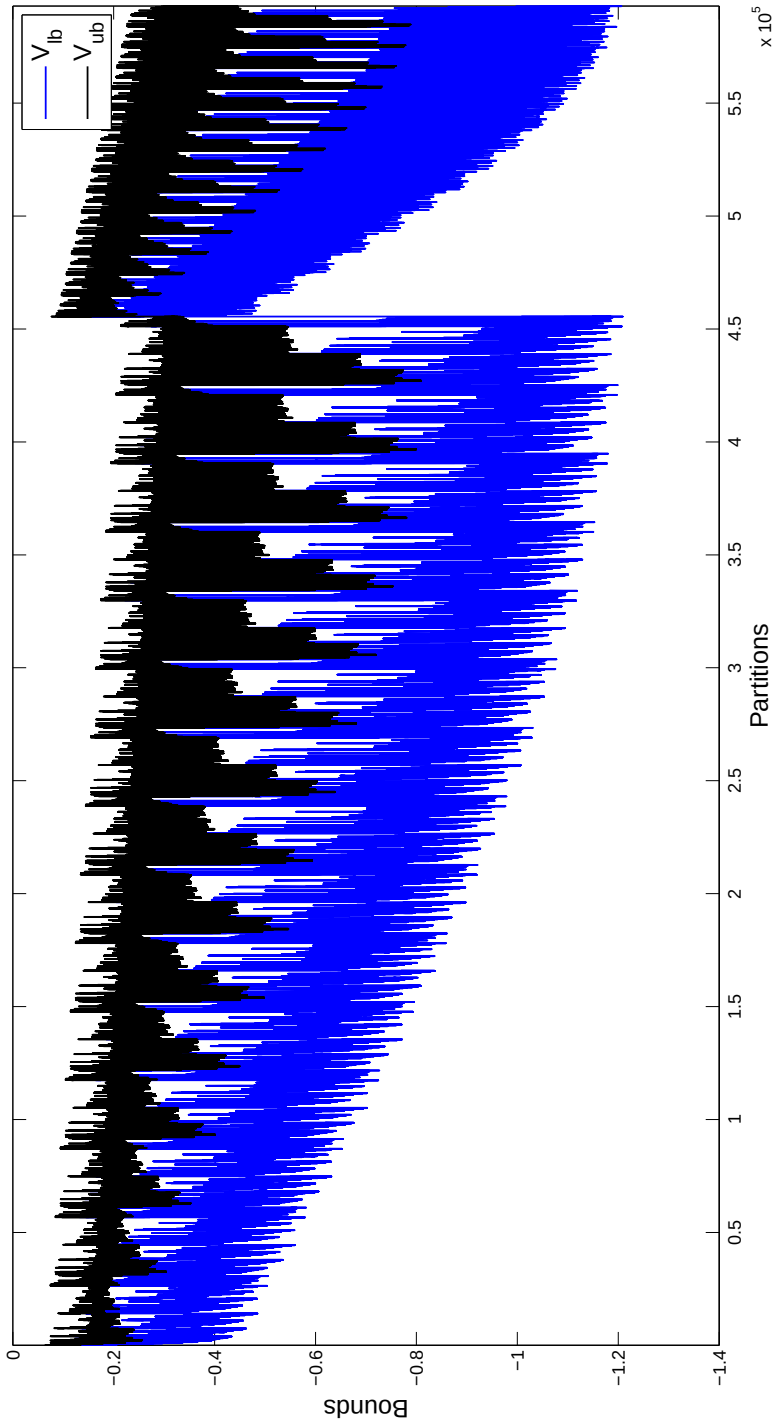

Figure 4.9: Upper and lower bounds for 2 UAVs and 8 stations problem.

serviced by them after 3 time units after they were raised. As one can see, the sub-optimal policy enables the robots to service more than 90% of the alerts within 5 units of time. Again, if the actual alert rate increases, the service delay will inevitably increases too. However, from our experience, $\alpha \leq 2$ is a reasonable number for the alert rate. Furthermore, the percentage of instances of alert where the maximum delay is more than 7 units of time is quite small. This can also be inferred from the Table 4.4.

4.3.1.3 Empirical Value Function

Since the value function of the surveillance problem with two robots is not available to compute, we propose an alternative way to test the effectiveness of the sub-optimal policy, π. Let $V_{emp}(\mathbf{x})$ be an empirical value function starting from state $\mathbf{x}$, which means, $V_{emp}(\mathbf{x})$ is actual discounted pay-off of the initial state $\mathbf{x}$ with the sub-optimal policy π. For demonstration purpose, we choose 15 partition sets such that all partition sets have same $l_s = 0, l_r = 2, T_1 = T_2 = 0, \bar{A} = (1, 0, 0, 1, 0, 1, 0, 1)$, but each partition set has different worst delay time, $\bar{\tau}$ is 1 to 15. Because there are still many states in a partition set, we pick non-dominating and non-dominated states in each partition set as the initial states. For each initial state, we run Monte Carlo simulation for 50 unit times; since $\lambda = 0.9$, the discounted pay-off for $t \geq 50$ is very small, $\lambda^t << 1$. For each initial state, 200 Monte Carlo simulations with differ-

Alert rate (α)	1	2	6
Mean Dwell Time	4.15	3.32	1.46
Mean Delay Time	3.41	3.86	5.57
Worst Delay Time	18	21	37
Total Info. Gain	0.0157	0.0123	0.0046

Table 4.4: Comparison of alert servicing performance for different alert rates.

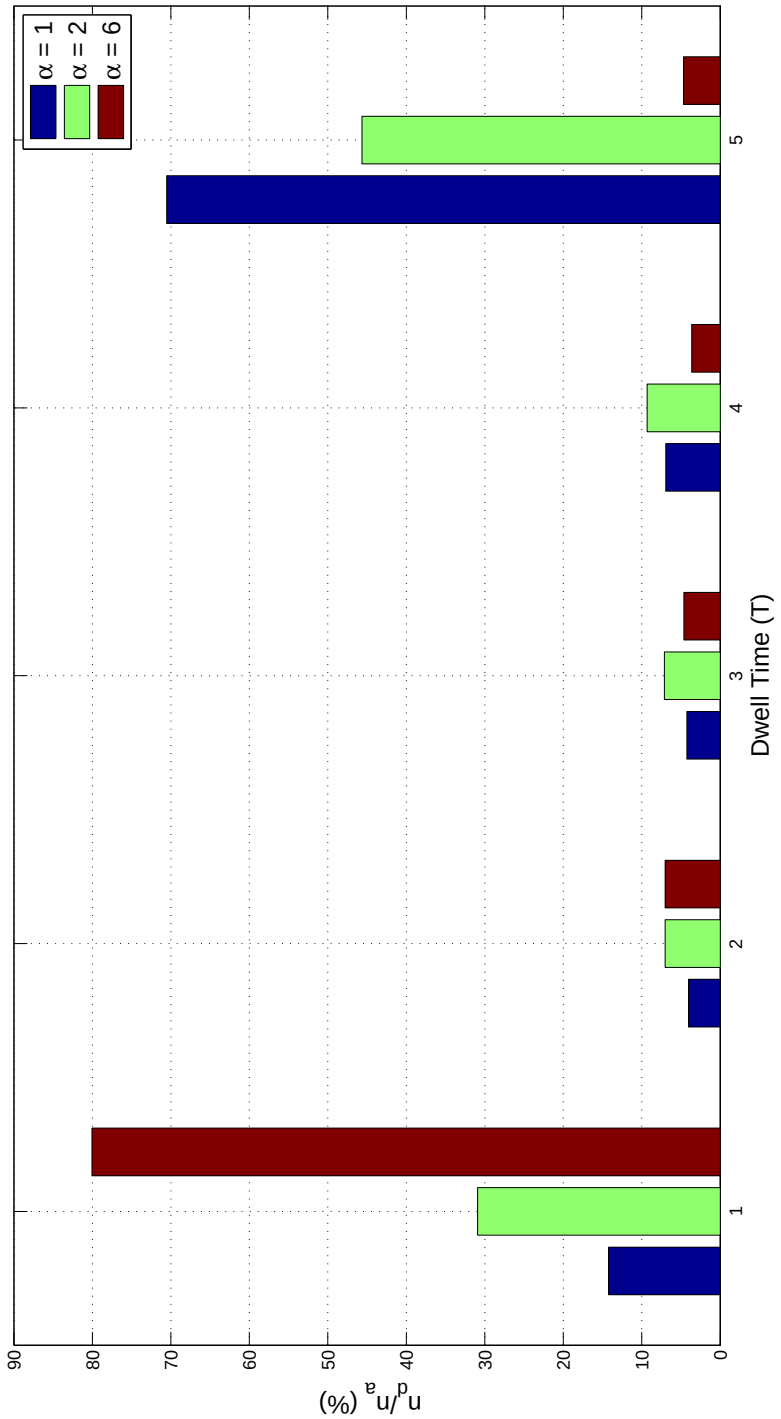

Figure 4.10: Monte-Carlo simulation results (dwell time)

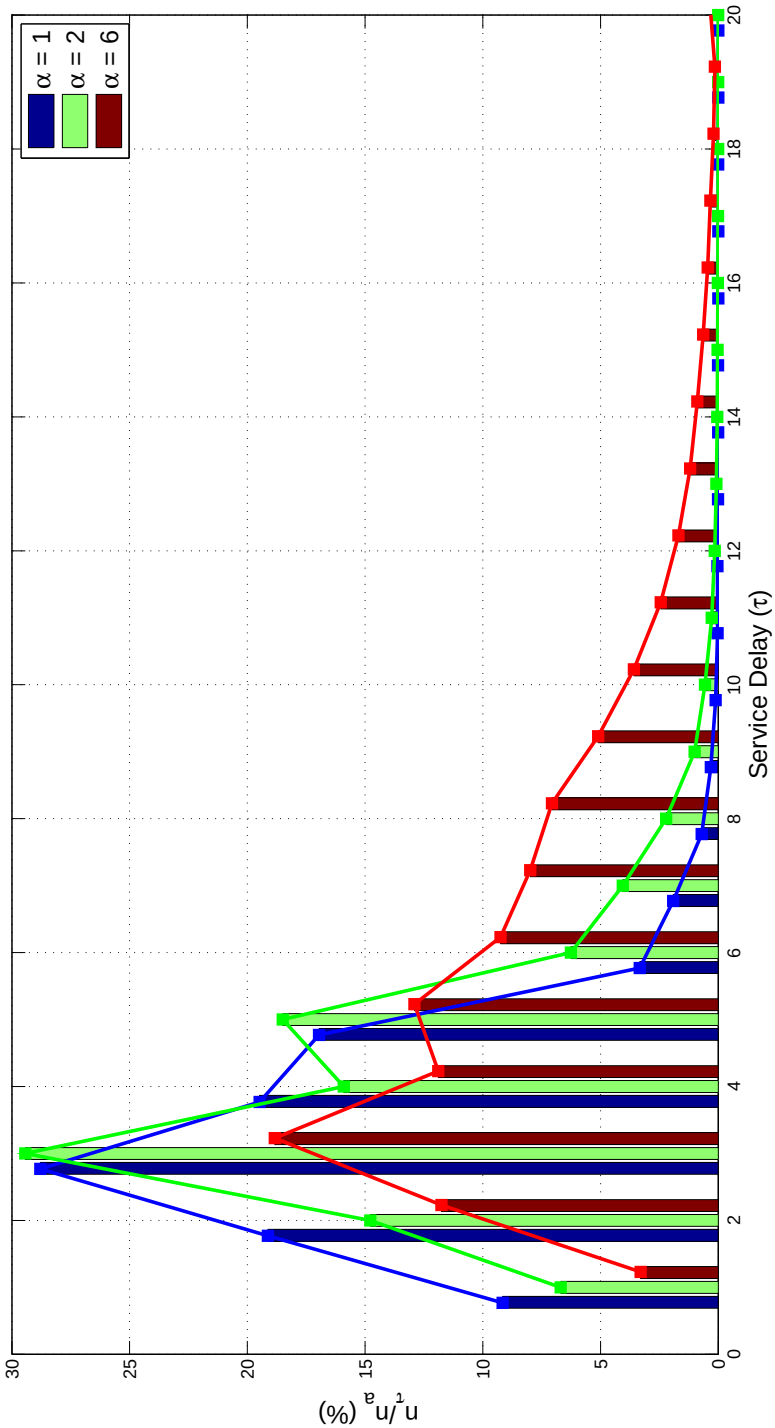

Figure 4.11: Monte-Carlo simulation results (delay time)

ent stochastic disturbance input instances are performed. We consider the average value of those result for each initial state as the empirical value function of state $\mathbf{x}$. Figure 4.12 shows the results. In each partition set, five states are chosen for the simulations; one for non-dominated state, and four states for non-dominating states. The X-axis label represents $\bar{\tau}$, so each seperation line is boundary of partition sets. As we can see, $V_{emp}(\mathbf{x})$ for sampled states $\mathbf{x} \in \mathcal{S}_i$ are bouned below by V_{lb}. If we increase the number of Monte Carlo simuations and simulation duration for each initial state, $V_{emp}(\mathbf{x}) \to V_\pi(\mathbf{x})$, where V_π is the sub-optimal performance function with the sub-optimal policy π. Hence the result supports our approach.

From two illustrative examples, it is shown that our bounds to the value function are reasonably tight and the performance value funcion of the sub-optimal policy is tighter. We solve **UBLP** and **LBNLP** for several different surveillance problems and those results are shown in Table 4.5. As you can see, the percentage error, %Err, is increasing as the number of states increases. However, the increasing ratio of the error is much less than one of the number of states. Moreover, %Err* is much less than %Err, which means, actual performance of our sub-optimal policy is closer to the performance of the optimal policy.

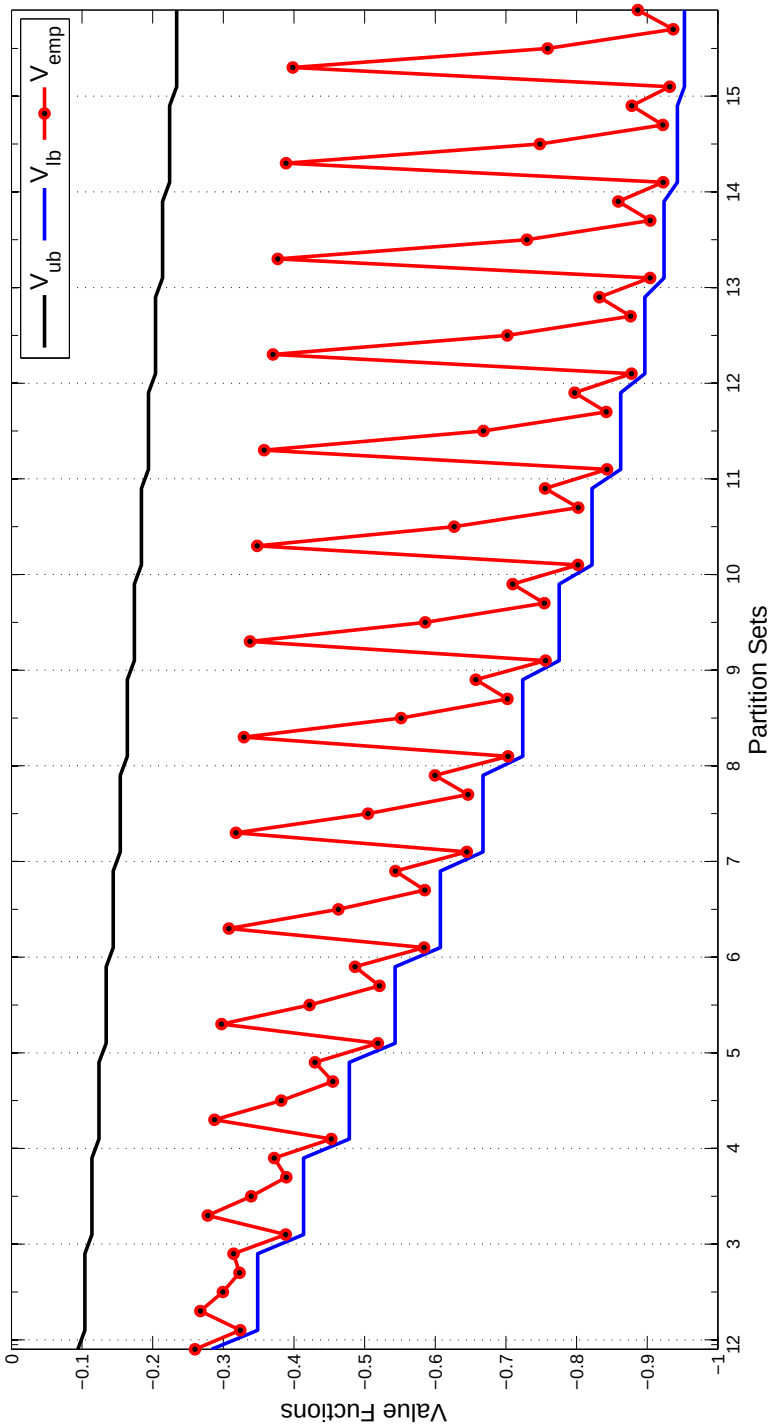

Figure 4.12: Empirical value function

n_r	n_s	N	$T_{\max}$	Γ	# of Queues	# of states	# of cyclic partitions	%Err (V_{ub} vs. V_{lb})	%Err* (V^* vs. V_π)
1	4	4	2	6	1	8,992	150	33.23	0.75
1	4	4	2	6	4	12,348	177	43.14	0.74
1	4	8	2	6	1	14,016	224	26.23	0.15
1	4	8	2	6	4	21,952	268	37.14	0.24
1	4	4	5	15	1	323,392	714	58.8	5.64
1	4	4	5	15	4	344,064	756	64.54	5.4
1	4	8	5	15	1	508,704	923	55.54	3.68
1	4	8	5	15	4	606,208	982	61.89	3.8
1	6	12	5	15	1	117,601,344	3,719	52.42	0.32
1	6	12	5	15	6	232,783,872	4,222	65.87	N/A
1	8	16	5	15	1	18,526,703,104	14,039	46.26	N/A
1	8	16	5	15	8	79,456,894,976	17,182	67.17	N/A
2	4	8	5	15	1	4,142,232	14,799	52.67	5.25
2	4	8	5	15	4	5,581,824	15,546	57.5	4.28
2	5	10	5	15	1	83,245,200	39,340	56.12	12.8
2	7	14	5	15	7	70,156,025,856	255,708	68.34	N/A
2	8	16	5	15	8	1,466,597,113,856	592,942	69.7	N/A

Table 4.5: Numerical results of surveillance problems.

5. CONCLUSION

In this dissertation, we have provided a state aggregation based restricted LP method to construct sub-optimal policies for stochastic DPs along with a bound for the deviation of such a policy from the optimum value function. As a key result, we have shown that the solution to the aggregation based LP is independent of the underlying cost function and we do so by demonstrating that the restricted LP is, in fact, the exact LP that corresponds to a lower dimensional MDP defined over the partitions. We also provide a novel non-linear program that can be used to compute a non-trivial lower bound to the optimal value function. In particular, for the perimeter patrol stochastic control problem, we have shown that both the upper and lower bound formulations simplify to exact LPs corresponding to some reduced order MDPs. To do so, we have exploited the partial ordering of the states that comes about because of the structure inherent in the reward function. It would be interesting to see if the simplification can be achieved for other problems that exhibit a similar structure. For the perimeter patrol problem, numerical results obtained via Monte Carlo simulations show that the sub-optimal policy obtained via the approximate value functions perform almost as well as the optimal policy. The literature suggests that, in general, the solution to a restricted LP depends on the underlying cost function; when the value function is parameterized by arbitrary basis functions. We have shown that, for the special case of hard aggregation, this is not true. Surely, there exist other basis functions with the same property and it would be useful to uncover the class of basis functions, for which the independence result holds.

We proposed a non-linear programming for the lower bound to the value function.

The solution to the DLP is independent of the cost function. There exists an m-dimensional MDP with much less constraints and its value function is the same with the solution to the NLP. A partition-level policy was proposed and we showed its performance is guaranteed by the solution to the NLP associated with the policy. We also provided a partition-level policy which provides better lower bound than any other partition-level policies.

Based on these approaches, we sucessfully could find a sub-optimal stationary policy for the perimeter patrol problem.

REFERENCES

[1] R.E. Bellman. *Dynamic programming*. Princeton University Press, 1957.

[2] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, 2007.

[3] P. Chandler, J. Hansen, R. Holsapple, S. Darbha, and M. Pachter. Optimal perimeter patrol alert servicing with Poisson arrival rate. In *AIAA Guidance, Navigation and Control Conf.*, Chicago, IL, August 2009.

[4] S. Darbha, K. Krishnamoorthy, M. Pachter, and P. Chandler. State aggregation based linear programming approach to approximate dynamic programming. In *Proc. IEEE Conf. Decision and Control*, pages 935–941, 2010.

[5] F. d'Epenoux. A probabilistic production and inventory problem. *Management Science*, 10(1):98–108, 1963.

[6] D. P. de Farias and B. Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, 2004.

[7] F. Glover. Surrogate constraints. *Operations Research*, 16(4):741–749, 1968.

[8] F. Glover. Surrogate constraint duality in mathematical pragramming. *Operations Research*, 23(3):434–451, 1975.

[9] H. J. Greenberg and W. P. Pierskalla. Surrogate mathematical programming. *Operations Research*, 18(5):924–939, 1970.

[10] D. Gross, S. Rasmussen, P. Chandler, and G. Feitshans. Cooperative Operations in UrbaN TERrain (COUNTER). In *Defense and Security Sympos.*, Orlando, FL, April 2006. SPIE.

[11] M. Grötschel and O. Holland. Solution of large-scale symmetric travelling salesman problems. *Math. Programming*, 51:141–202, 1991.

[12] M. Grötschel, L.Lovász, and A. Schijver. The ellipsoid method and its consequences in combinatorial optimization. *combinatorica*, 1(2):169–197, 1981.

[13] A. Hordijk and L. C. M. Kallenberg. Linear programming and Markov decision chains. *Management Sci.*, 25(4):352–362, 1979.

[14] R.A. Howard. *Dynamic programming and Markov process*. The MIT Press, 1960.

[15] D. Kingston, R.W. Beard, and R.S. Holt. Decentralized perimeter surveillance using a team of UAVs. *IEEE Transactions on Robotics*, 24(6):1394–1404, 2008.

[16] K. Krishnamoorthy, M. Pachter, P. Chandler, D. Casbeer, and S. Darbha. UAV perimeter patrol operations optimization using efficient dynamic programming. In *Proc. American Control Conf.*, pages 462–467, 2011.

[17] K. Krishnamoorthy, M. Pachter, P. Chandler, and S. Darbha. Optimization of perimeter patrol operations using UAVs. *AIAA J. Guidance, Control and Dynamics*, 35(2):434–441, 2012.

[18] K. Krishnamoorthy, M. Pachter, S. Darbha, and P. Chandler. Approximate dynamic programming with state aggregation applied to UAV perimeter patrol. *Internat. J. of Robust and Nonlinear Control*, 21(12):1396–1409, 2011.

[19] K. Krishnamoorthy, M. Park, S. Darbha, P. Chandler, D. Casbeer, and M. Pachter. Lower bounding linear program for the perimeter patrol optimization problem. *AIAA Journal of Guidance, Control, and Dynamics*, in press, 2013.

[20] K. Krishnamoorthy, M. Park, M. Pachter, P. Chandler, and S. Darbha. Bounding procedure for stochastic dynamic programs with application to the perimeter patrol problem. In *Proc. American Control Conf.*, pages 5874–5882, Montreal, QC, CA, June 2012.

[21] A.S. Manne. Linear programming and sequential decisions. *Management Science*, 6(3):259–267, 1960.

[22] J.S. Marier, C. Besse, and B. Chaib-draa. Solving the continuous time multiagent patrol problem. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 941–946, 2010.

[23] A. Marino, L.E. Parker, G. Antonelli, F. Caccavale, and S. Chiaverini. A fault-tolerant modular control approach to multi-robot perimeter patrol. In *Proc. IEEE International Conf. on Robotics and Biomimetics (ROBIO)*, pages 735–740, 2009.

[24] D. A. Paley, L. Techy, and C. A. Woolsey. Coordinated perimeter patrol with minimum-time alert response. In *Proc. Guidance, Navigation and Control Conf.*, number AIAA 2009-6210, Chicago, IL, 2009.

[25] E.L. Porteus. Bounds and transformations for discounted finite Markov decision chains. *Operations Research*, 23(4):761–784, 1975.

[26] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*, volume 414. Wiley. com, 2009.

[27] P. J. Schweitzer and A. Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110(2):568–582, 1985.

[28] M. Trick and S. Zin. Spline approximation to value functions: A linear programming approach. *Macroeconomic Dynamics*, 1:255–277, 1997.

[29] B. Van Roy. Performance loss bounds for approximate value iteration with state aggregation. *Mathematics of Operations Research*, 31(2):234–244, 2006.