ABSTRACT

ALEBRAHIM, EBRAHIM KH E S. Stochastic Dynamic Optimization in Spatial and Network Resource Economic Models. (Under the direction of Paul Fackler).

Explicit spatial optimization models in resource economics, such as control of invasive species, and disease spread, tend to be of high dimensional nature. Efficient optimization methods, such as dynamic programming, suffer from what is known as the curse of dimensionality and hence incapable of solving such problems. This dissertation explores and tests various approximate dynamic programming methods to solve such problems. Our analysis is based on a stylized invasive species model. The model is a stochastic discrete susceptible infected susceptible (SIS) diffusion, or transmission, model where problems such as pest infestation, and disease and viral spread can be modeled similarly. This means our treatment in this dissertation would be useful for a wide variety of resource problems with spatial or network structure. The dissertation consists of four main parts. In the first part, we provide a review of the importance of spatial models in both economics and resource economics. In the second part, we review some fundamentals of dynamic optimization. In the third part, we propose and explore a rank-based policy function approximation method. In the fourth part, we introduce an overview of reinforcement learning methods, as an approximate dynamic programming approaches, illustrate its potential in solving small and high dimensional problems and explore the performance of it using linear in parameters approximation algorithms for the approximation of the state-action value function. Our analysis shows that both approaches have the potential to solve moderate to high dimensional dynamic resource economics problems.

© Copyright 2020 by Ebrahim KH E S Alebrahim

All Rights Reserved

Stochastic Dynamic Optimization in Spatial and Network Resource Economic Models

by
Ebrahim KH E S Alebrahim

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Economics

Raleigh, North Carolina
2020

APPROVED BY:

Krishna Pacifici	Harrison Fell
Daniel Tregeagle	Paul Fackler Chair of Advisory Committee

DEDICATION

To my parents, wife, and family.

BIOGRAPHY

Ebrahim was born in Kuwait. He has a Bachelor of Science in Electrical Engineering from Kuwait University. After his graduation, he has joined the Kuwait Institute of Scientific Research (KISR) as a research assistant. He has worked in KISR for about two years, where he has participated in several renewable energy research projects as both a team member and a team leader. In 2012, Ebrahim moved to North Carolina to pursue his graduate study in Economics.

ACKNOWLEDGEMENTS

I want to express my special thanks to my advisor, Paul Fackler, for advising my dissertation work. His thought-provoking comments, feedback, and continuous support have played an important role in my dissertation work. I would also like to express my gratitude to my committee members for their constructive feedback and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1 INTRODUCTION

1.1 Introduction

This dissertation generally focuses on the problem of resource allocation over space and time. We explore several approximate dynamic programming (ADP) methods (Powell 2008) for controlling a stylized stochastic invasive species dynamic spatial model. We specifically focus on the situation where the dimension of the problem is intractable to solve with exact numerical methods, such as dynamic programming (Bellman 1954). Although our study is based on a stylized invasive species problem, its implications could be useful for other similar problems, such as disease spread.

This chapter consists of two main parts: a literature review and a description of the dissertation outline. The literature review starts with a discussion of the importance of the spatial model models in economics, in general, and resource economics more specifically. The review concludes with a discussion of the literature that has explored approximate dynamic programming methods in resource economics problems.

1.2 Literature

Many fields of sciences use spatial models, including ecology and resource economics. Disease spread (Ferguson et al. 2005), pest infestation (Bianchi et al. 2010), and invasive species (Epanchin-Niell and Hastings 2010) are some examples. Spatial models can be either implicit or explicit.

Explicit models are models that track the dynamics of the state of the locations individually. Bianchi et al. (2010) provides examples of such models. Implicit models are models that track the dynamics of the state of the locations in a categorized aggregate way. Spatial category count models are examples of implicit models (Fackler 2012). To illustrate the two types of models, suppose we have several sites, n, where some pest can infest each of them. Assume that the infestation level represents the state of each site. An explicit model in this example tracks the evolution of the infestation level for each site, while for an implicit model, we keep track of the evolution of the number of sites in each category of infestation level (i.e., low, medium and high).

Spatial models have played an important role in economics for at least a century. For example, Hotelling's location model (Hotelling 1929) and the work by (Sraffa 1926) are considered influential works in the theory of firms' competition where space plays an important role. Schelling's segregation model (Schelling 1971) is an example of the spatially explicit dynamic models and considered a seminal work in the field of urban economics and complexity economics (Arthur 1999). In relatively recent work, the spatial nature of economics problem is contextualized in what is known as the New Economics Geography (Krugman 1998).

In resource economics, space plays an important role. Albers et al. (2010) classifies spatial resource economics studies into two types. The first is positive studies; that is, they focus on studying the effect of policy changes on some economic outcomes. The second is normative or resource management studies.

Positive resource economics studies branch into many categories. Hedonic valuation models (Lancaster 1966; Court 1939; Waugh 1928), and sorting models (Palmquist 2005) are some of the well-known branches where space matters (Ando and Baylis 2014).

Hedonic valuation models are based on the idea that product prices are derived from their characteristics (Court 1939). An example is the valuation of housing attributes such as proximity to open space, farmlands, wetlands, and other environmental and infrastructure amenities (Johnston et al. 2002). Hedonic methods play an important role in resource economics policies (Palmquist and Smith 2001). Taylor (2003) provides an overview of the hedonic valuation approach.

Sorting models are used to study the equilibrium sorting behaviors of heterogeneous agents taken into account the endogeneity of the sorting process (Kuminoff et al. 2010). The work of Klaiber and Phaneuf (2010) is an example. They use sorting models to address household location preference toward open spaces. Kuminoff et al. (2010) provides an overview of sorting models with their implications on policy evaluations.

Space plays an essential role in many of the resource management problems. There are two major approaches to dealing with spatial resource management problems. The first is the dynamic optimization approach. The second is the heuristic policy approach (Chadès et al. 2011).

The dynamic optimization approach is used to solve small tractable problems or to get policy insights for larger problems (Epanchin-Niell and Wilen 2012). For example, Sanchirico and Wilen (1999) studies the equilibrium nature of the dynamics of renewable resources over space. The spatial nature in their model is manifested in the form of interconnected patches of habitats. Costello and Polasky (2004) explores the problem of reserve site selection, where the objective is to maximize the number of conserved species over a fixed planning horizon. Brock and Xepapadeas (2011) derives analytical results of the steady-state spatial patterns due to human actions for biological processes.

Heuristic policy approach has been used in several resource economic problems with a large dimension where dynamic optimization methods may not be tractable. For example, Marcel Salathe (2010) develops an immunization policy and explores it in a social network with community structure (M. Girvan and Newman 2002). Chadès et al. (2011) proposes a rule of thumb to control the spread of invasive species over certain spatial network structures. Perry et al. (2017) proposes a priority policy rule based on network centrality measures to control invasive species over a spatial network structure.

There are a good number of studies of control problems for spatial models in both biology and ecology that are very relevant to resource economics. Hof and Bevers (2002) provides a comprehensive overview of the use of optimization and simulation models in ecological economics applications. They also provide a discussion of why simulation-based-optimizations has drawn less attention by resource economists. DeAngelis and Yurek (2017) provides an extensive review of spatially explicit models in ecology.

Spatial models can be seen, and modeled, as a network model, where a particular connectivity structure connects sites. Chadès et al. (2011); Laber et al. (2018) are some works that represented spatial structure in a network framework. There is a rich literature in the network field that discusses dynamic processes over networks. Newman (2009) and Lang et al. (2018) discuss the spread of a disease over a network. Liu et al. (2016) studies immunization strategies in network structures. Proulx et al. (2005) provides an overview of the network analysis for ecological applications, such as the spread of invasive species. The works mentioned above are of resource economics nature. Newman (2007); Barabási (2016); Strogatz (2001) provide general expositions to networks, including networks coupled with dynamic processes.

The management and the control of invasive species, (Epanchin-Niell and Wilen 2012; Chadès et al. 2011), is considered a critical and important problem. It has attracted the attention of both ecologists and resource economists (Olson and Roy 2002).

Invasive species are estimated to cost more than a hundred billion dollars per year (Olson and Roy 2002). The explicit spatial nature of invasive species management and the uncertainty in the dynamics of the species are critical (Meier et al. 2014; Chalak et al. 2011). Spatially explicit models suffer from the curse of dimensionality, that is they grow exponentially in size with the number of sites. Efficient methods such as Dynamic Programming (Bellman 1954) can only solve small size problems and is intractable for larger ones.

The research effort of explicit spatial modeling of the problem is quite limited (Epanchin-Niell and Wilen 2012). Recent research concerned with relatively large dimensional explicit invasive species spatial model is either based on deterministic models (Epanchin-Niell and Wilen 2012) or development and testing of heuristic rules based on stochastic models (Chadès et al. 2011; Perry et al. 2017).

Our research aim at extending the literature by exploring the potential of several approximate dynamic programming (ADP) methods (Powell 2011) in obtaining good policy rules for stochastic dynamic spatial problems where dynamic programming is deemed infeasible. We explore and apply some ADP methods to the invasive species stochastic spatial model of Chadès et al. (2011). Our focus is more on the computational aspects of the methods, in the sense of what works and what may not work. This distinguishes our research from similar other researches, such as Epanchin-Niell and Wilen (2012), and Chadès et al. (2011). The focus of Chadès et al. (2011) is on proposing a heuristic rule for a spatial stochastic model. Epanchin-Niell and Wilen (2012) focuses on providing policy insights by solving a deterministic larger dimensional spatial model using mixed-integer programming. Our research provides insight into the potential of some approximate dynamic programming methods.

In addition to that, we propose a simple-to-implement novel method. The method exploits prior knowledge of the model, such as potential symmetry in the model, into the structure of a policy rule. The method we propose utilizes some network measures. The use of prior knowledge in function approximation has been shown to have a significant impact on the efficiency of the function estimation (Belbute-Peres et al. 2018). We also illustrate that the policy approximation method under our study can help in deriving simple heuristic rules. Policymakers can apply such rules for problems with very large dimensions.

Economic research on the methods of approximate dynamic programming (ADP) is largely based on solving small and moderate size problems (Judd et al. 2011). Springborn and Faig (2019); Laber et al. (2018) are the only works we are aware of that explores ADP methods for large dimensional resource management problems.

Springborn and Faig (2019) demonstrates ADP on a simple stochastic fishery problem developed by Reed (1979). In their problem, they have one state variable, which is the stock, and one action, which is the harvesting. Both variables are continuous. Their analysis is based on approximating the value function using Gaussian process regression (Rasmussen and Williams 2005).

Laber et al. (2018) work is based on the dynamic treatment regime methods (Murphy 2003). Their novel contribution is in extending it to a spatial situation where the number of possible actions is very large. They demonstrate their method on the control of the white-nose disease in bats.

Our work extends the literature of ADP in resource economics in multiple aspects. First, we explore ADP methods that have not been explored in the mentioned works. Second, we apply those ADP methods to a spatially explicit stochastic invasive species control problem. Third, we show that some ADP methods, such as policy approximation methods, can help in deriving simple heuristic rules that can be applied to very large dimensional problems.

1.3 Dissertation Outline

The dissertation consists of three main parts. The first part, which is the second chapter, provides a general overview of Markov Decision Process (Bellman 1957), Dynamic Programming (Bellman 1954), and Approximate Dynamic Programming (Powell 2011). This part aims to familiarize the reader with fundamental concepts that later chapters rely on. Additionally, we provide a short introduction to the importance of approximate dynamic programming methods and the difficulty associated with them.

The second part, which is the third chapter, presents the first method that we explore. The method is based on policy function approximation, and the estimation method proposed by Smith (1990) and explained in Judd (1998). The method also appears to be independently proposed by Moxnes (2003) in what he names as stochastic optimization in policy space (SOPS). Our exploration is based on the invasive species model presented in

Chadès et al. (2011).

The third part, which is the fourth chapter, discusses and explores some of the value function approximation methods. Our treatment, in this part, has two objectives. The first is to provide an overview of reinforcement learning (Watkins 1989; Sutton and Barto 2018), which is a different terminology of approximate dynamic programming in computer science. The second is to illustrate and explore the method of Least-Square-Policy-Iteration (Lagoudakis and Parr 2003) on our invasive species control problem.

CHAPTER

2

PRELIMINARIES

2.1 Introduction

This chapter provides an overview of some fundamental methods and concepts that are useful for subsequent chapters. The treatment of this chapter will be as follows. First, we introduce the Markov decision process (MDP) as a framework for modeling dynamic decision-making problems. Second, we introduce an overview of dynamic programming (DP) and Bellman equation (Bellman 1954). Third, we provide an overview of the two widely used dynamic programming solution methods: the value function iteration, and the policy function iteration methods. In the last section, we provide a brief introduction to approximate dynamic programming (ADP) (Powell 2008).

2.2 Markov Decision Process

A Markov decision process (MDP) (Bellman 1957; Howard 1960) consists of a set of state and action variables, a reward function, and a transition rule that models the evolution of the state variables for each action. In the MDP, future states depend only on current states, and actions (i.e., history is irrelevant). MDP provides an elegant framework for modeling many practical dynamic decision-making problems of stochastic nature (Howard 1960). Dynamic programming (Bellman 1954) is considered an efficient tool for solving MDP problems (Howard 1960; Bellman 1957). Resource economics is pervasive with such MDP problems where agents, such as resource managers, are in a situation where they seek to take the best actions over some period of time in order to achieve certain goals. Such situations, or problems, are formulated as dynamic optimization problems (Marescot et al. 2013).

2.3 Dynamic Programming

Any dynamic optimization problem generally consists of two main components (Puterman 1994): a system model, and an objective function. The system model is a model that describes how the system behaves, and it consists of three main components: state variables, which describe the state of the system, action variables, which describe the actions that can be taken on the system, and a transition model, which describes how the system evolves.

To state this in a mathematical form and assuming all the variables are discrete, let S_t represent a vector of state variables, A_t a vector of action variables, and ϵ_t represents a vector of random variables, where t represents the period. Note that in the case of an infinite horizon problem, we use t to represent the current period and $t+1$ for the next period. Let $S_{t+1} = g(S_t, A_t, \epsilon_t)$ represents the system model, which can be stochastic, and $r(S_t, A_t)$ represents a reward function. A widely used objective function is the expected discounted sum of the reward function over some time horizon T (Marescot et al. 2013; Puterman 1994). Let β be a discount factor. An agent's problem is to seek the optimal actions that maximize the expected discounted rewards given the current state (S_1). That is

$$\max_{A_t} \sum_{t=1}^{T} \beta^t E[r(S_t, A_t)|S_1] \tag{2.1}$$

$$\text{such that} \tag{2.2}$$

$$S_{t+1} = g(S_t, A_t, \epsilon_t). \tag{2.3}$$

There are several ways to solve such a problem. If the problem is an MDP, which is the case for the model under our study, then dynamic programming, (Bellman 1954), is considered an efficient and elegant way to solve such a problem (Bellman 1957). Let π denote a policy function, which is a mapping from the states to the actions, and let $V(S_t)$ be the value function at S_t, which is the expected discounted sum of rewards following a policy π. The above optimization problem can be written in a recursive form, in what is known as the Bellman equation, as

$$V(S_t) = \max_{A_t} r(S_t, A_t) + \beta E[V(S_{t+1})|S_t, A_t]. \tag{2.4}$$

Bellman equation can be solved in different ways, as illustrated in the subsequent sections. The solution is characterized by an optimal policy function, let it be $\pi_t^*(S_t)$, which gives the optimal action given state S_t, and an optimal value function, let it be $V_t^*(S_t)$. For infinite horizon problems, the policy rule and the value function will be stationary (i.e., time-independent).

2.4 Value Function Iteration

The Bellman equation is of recursive form. The value function iteration method (Bellman 1957) states that the infinite recurrence of the Bellman equation results in the maximum value function. To illustrate this, let V^* denotes the vector of maximum values for every state, V^0 denotes a vector of initial values, which could be set arbitrarily to zero, and V^n be the vector of values at iteration n. Starting with $n=0$, then value function iteration method solves:

$$V^{n+1}(S_t) = \max_{A_t} r(S_t, A_t) + \beta E V^n(S_{t+1}), \tag{2.5}$$

until $|V^{n+1} - V^n| < \epsilon$, where ϵ is some small number. Given certain conditions (Puterman 1994), V^n approaches V^* for large n, that is $\lim_{n\to\infty} V^n = V^*$. For finite horizon problems, one would start with the last period values and substitute backward into the Bellman equation.

2.5 Policy Iteration

Howard (1960) proposed the policy iteration algorithm to solve infinite horizon problems. The algorithm consists of two steps. The first is to improve the policy by using the most recent values of the value function. That is, in iteration n,

$$\pi_n = \operatorname*{argmax}_{\pi} r(S_t, A_t) + \beta E V^n(S_{t+1}). \tag{2.6}$$

The second step is to update the values of the value function given the new policy. That is

$$V^{\pi_n}(S_t) = r(S_t, \pi_n(S_t)) + \beta E V^{\pi_n}(S_{t+1}). \tag{2.7}$$

Equation (2.7) is a system of linear equations that can be solved for the value function at each state. Let P^{π_n} denote the transition matrix, where columns represents current state and rows future states for policy π_n, then $E V^n(S_{t+1}) = P^{\pi_n{}'} V^n(S_{t+1})$, and Equation (2.7) can be written as

$$V^{\pi_n} = r(S, \pi_n(S)) + \beta P^{\pi_n{}'} V^{\pi_n} \tag{2.8}$$

$$\implies V^{\pi_n} = (I - \beta P^{\pi_n{}'})^{-1} r(S, \pi_n(S)). \tag{2.9}$$

Note that we have dropped the time scripts since we are assuming an infinite horizon and hence V^{π_n} is stationary (Howard 1960).

The policy iteration method distinguishes itself from the value function iteration method in two aspects (Howard 1960). First, it converges when no policy improvement occurs in two consecutive iterations. Second, it tends to converge in fewer iterations.

2.6 Approximate Dynamic Programming

Although dynamic programming is a powerful method for solving dynamic optimization problems, it suffers from the curse of dimensionality. Powell (2011) explains this as three curses of dimensionality. The first one is related to the state space, which is all possible outcomes of the state variables. The state-space dimension can easily become very large as it grows exponentially with the number of state variables. The second one is related to the action space, which is all possible actions that can be taken. Similarly, the action space grows exponentially with the number of action variables. The third one is related to the stochasticity of the system, or the outcome (random) variables (i.e., ϵ). The outcome space increases exponentially with outcome variables making the computation of the expectation hard or even impossible. This makes dynamic programming intractable for large dimensional problems, and hence approximate dynamic programming methods come into play.

Approximate dynamic programming (ADP) is a collection of different methodologies that aims at obtaining a near-optimal policy or value function (Powell 2008). Most methods are based on approximating either the policy function or the value function by some parametric or non-parametric functions (Powell 2011). Approximate dynamic programming is also known by other names, such as neurodynamic programming (NP) (Bertsekas and Tsitsiklis 1996) and reinforcement learning (RL) (Sutton and Barto 1998). The success of approximate dynamic programming methods is problem specific (Powell 2011). This means a method that works for a particular type of problem may not work for other types of problems. This requires an exploration of a variety of methods for the problem of interest. For example, a method based on policy function approximation may work better for certain problems, while a method based on value function approximation may work better on others.

In our research, we aim to explore and apply different methods on a stylized spatial invasive species MDP problem. The stylized model is based on Chadès et al. (2011) stochastic discrete susceptible infected susceptible (SIS) diffusion, or transmission, model. Our findings can be useful to other problems based on SIS models, such as disease spread.

CHAPTER

3

A RANK BASED POLICY FUNCTION APPROXIMATION IN SPATIAL MODEL

3.1 Introduction

This chapter aims at introducing and testing the performance of a simple tunable approximate policy function. The discussion in this chapter will be as follows. In the first part, we provide a brief review of stochastic optimization in policy space as an approximate dynamic programming method (Moxnes 2003; Judd 1998; Smith 1990). The second part lays out the spatial test model that we will use in the exploration of the approximate dynamic optimization method followed by a discussion of our approximate policy function. Afterward, we provide an overview of how features of the policy function can be selected. Next, we discuss the different types of approximate dynamic programming evaluation methods. The last part is the analysis part, which explores our rank-based policy on both regular grids situation as well as Chadès et al. (2011) multi-layer test network.

3.2 Policy Function Approximation

As we have mentioned in previous chapters, approximate dynamic programming is a collection of different types of methodologies that aim at obtaining a near-optimal policy or value function. One simple method is parametric policy function approximation. As explained in Judd (1998) and Smith (1990), one would first propose a parametric function for the policy rule, and then estimate the parameters through simulation, such that the objective function is maximized.

To illustrate the method, let $A_t = \pi(S, \alpha)$ be a parametric policy function approximation, where α is a vector of parameters. Let M denote the number of simulation replications of the system model over time T [1]. The optimization problem can be solved by solving for the vector of parameters (α) in

$$\max_{\alpha} \frac{1}{M} \sum_{m=1}^{M} \sum_{t=0}^{T} \beta^t r(S_{mt}, \pi(S_{mt}, \alpha)). \tag{3.1}$$

There are many optimization techniques, or solvers, that can be used to solve the problem in Eq. (3.1). The choice of the solver to be used depends on the continuity and linearity of the objective function (Eq. 3.1). As we show in later sections, in our spatial

[1]Note that we have to choose T large enough such that the solution is close enough to the infinite horizon problem. We discuss the choice of T in subsequent sections.

problems, both the reward and policy functions are discrete. Additionally, our proposed policy function will be non-linear. Thus, we use a robust gradient-free global optimizer such as DIvided RECTangles (DIRECT) optimization algorithms.

Define a feature as a function of state variables that can be used as a predictor in the policy function. The choice of features in the approximate policy function is one significant and critical thing to be considered to obtain a good policy function. It is a very challenging step and is based on both trial and error and the consideration of the structure of the problem.

Once some features are decided on, it is essential to know how good or how close is the approximate policy to the optimal one or some baseline policies. Evaluating the performance of the approximate function is another major challenge. Powell (2008) states three ways for policy evaluation. The first one is to compare the approximate policy to an optimal policy of a simpler problem. The second is to compare it to an optimal policy of a deterministic version of the problem. The third way is to compare it to a myopic or greedy policy. Each approach will be elaborated in more depth in later sections.

3.3 Model

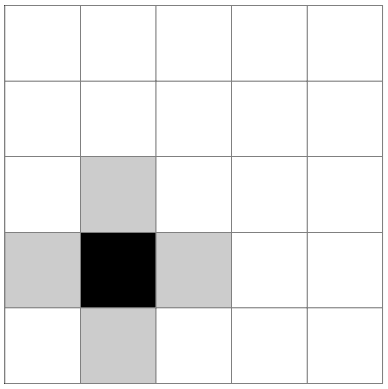


Figure 3.1: An illustration of a 5x5 grid. An example site is black shaded and its neighbors are grey shaded.

Our analysis is based on Chadès et al. (2011) model. We explore both regular grid model and Chadès et al. (2011) multi-layer star network model. In both situations the model consists of multiple sites numbered from 1 to N_s. Each site is represented by a state variable, S_i, that takes two values: 1: site occupied, and 0: site unoccupied. We assume that one site is treated each period, and the action variable represents the number of that site and 0 if no action is to be taken. We also assume that preventive action (treating an uninfected site) is possible. Let S_i denote the state variable for a site i, A the action variable, which is the site to treat, and S_i^+ the next period state value. Given S_i, and A, the state variable of site i evolves to the occupied state according to the probability model:

$$P(S_i^+ = 1|S_i, A) = S_i(1-\mathbb{1}_{A=i})(1-p_n)+S_i\mathbb{1}_{A=i}(1-p_t)+(1-S_i)(1-\mathbb{1}_{A=i})[1-(1-p_0)(1-p_1)^{q_i}], \quad (3.2)$$

where $(1-p_n)$ is the probability that site i will remain occupied in the next period when no action is taken, $(1-p_t)$ is the probability that site i will remain occupied in the next period when an action is taken, $[1-(1-p_0)(1-p_1)^{q_i}]$ is the probability that site i becomes occupied when no action is taken due to having occupied untreated neighbors ($(1-p_1)^{q_i}$) or due to spontaneous or exogenous occupation $(1-p_0)$ where q_i is the number of directly connected sites (neighbors) to site i that are occupied but not treated. The model also assumes that unoccupied treated sites remain unoccupied. The neighbors of a site are defined to be the four directly adjacent neighbors to site i, which is known as the von Neumann neighborhood. Figure 3.1 shows an illustration of a five by five grid, where each cell is a site. An example site is black shaded, and its neighbors are grey shaded.

Now we define the management problem. Let $\pi(S_t, \alpha)$ denote a policy rule which maps state to action at period t where S_t is an n-element vector that represents the state values of the sites at time t, and α is a vector of parameters. The management problem can be characterized by the policy rule $\pi(S_t, \alpha^*)$ that achieves the minimum expected discounted sum of the number of occupied sites, that is

$$\min_{\alpha} E\left[\sum_{t=0}^{T} \beta^t \sum_{i=1}^{n} S_{it} | \pi\right], \tag{3.3}$$

where β is a discount factor, and T is the terminal period. In our analysis, we assume that the time horizon is infinite, where we approximate it by finite time horizon of T periods. We discuss the choice of T in later sections.

3.4 Policy Rule

The functional form of our tunable or parametric policy rule can generally be expressed as follows:

$$SiteIndex_i = \phi(S, \alpha_i), \tag{3.4}$$

where S is a vector of the states variables, ϕ is a function, which can be linear or non-linear, and α_i is vector of parameters to be estimated for site i. The action to be taken is

$$A = \pi(S, \alpha) = \underset{i}{\operatorname{argmax}}(SiteIndex_i), \tag{3.5}$$

that is an action will be taken for the site with the largest index value. It should be noted that the policy rule is non-linear in α. The non-linearity is due to the max operator, even if the site index (ϕ) is linear in α.

3.5 Features Selection For Spatial Models

As stated previously, feature selection is one challenging step in the construction of approximate policy rules. In spatial models, one natural feature is the state of the site itself. That is, our proposed policy rule can be stated as $SiteIndex_i = \alpha_i \cdot S_i$. Although such a simple policy might result in near-optimal policy rules in small problems, it may not be sufficient for larger problems. This requires one to add features that could be functions of the states of other sites. Representing the spatial structure of the model efficiently is very important and helps in the construction of features.

One simple, yet powerful, way to represent a spatial model is to represent it by, what is known in the graph and network theory, an adjacency matrix. In its simplest form, an adjacency matrix is a symmetric matrix where each row represents a site i, and each cell in row i, and column j contains "1" if site i is connected (neighbor) to site j and "0" otherwise.

There are two advantages of using the adjacency matrix. First, it allows exploiting the graph and network theory literature in the construction of features. Second, it allows us to calculate the features in an efficient way, which is very critical in simulation-based optimization as the policy rule is called many times.

To illustrate this, suppose we have a two by two grid model. The model can be represented by the adjacency matrix

$$\zeta = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}. \tag{3.6}$$

Suppose that we plan to add a new feature to the policy rule, such as the total number of neighboring sites with state values $S_i = 1$, and denote it by q_i. The feature can be efficiently calculated as $q_i = \zeta \cdot S$ where S is a column vector of the state values.

3.5.1 Network Centrality

In this section, we discuss how network measures, such as network centrality can be used to construct useful features. Centrality measures define how important a node is in a network (Newman 2007). The simplest centrality measure is known as the degree centrality. The degree centrality for a node (or a site) in a network is defined to be the number of directly connected nodes to that node. For example, under this definition, a node that is directly connected to a larger number of nodes is considered more important or central than one that is directly connected to less number of nodes.

The degree centrality does not reflect a deep measure of centrality for a node (Newman 2007). For example, one could have two types of nodes, one that has a large number of neighbors, hence has large degree centrality, but its neighbors have no other neighbors, and another node that is connected to few neighbors, hence has a small degree of centrality, but its neighbors have a large number of neighbors. Based on degree centrality, one would conclude that the first node is more central than the second. Furthermore, in the grid networks, the degree centrality will be the same, that is four, for all interior sites that are one site away from the edges, and hence will not be a good measure of the position of the sites relative to the edges.

Eigenvector centrality is a centrality measure that is deeper than degree centrality in the sense that it considers not only the directly connected neighbors but also the neighbors of the neighbors. This centrality measure provides a score of how central a node is (Newman 2007).

Following the treatment in Newman (2007) and Bonacich (1972), let x_i represent the eigenvector centrality of node i and ζ an adjacency matrix for the network of interest, then the eigenvector centrality for node i is defined as[2]

$$x_i = \frac{1}{\lambda} \sum_{j \in G_i} \zeta_{ij} x_j, \tag{3.7}$$

or in vector form as

$$\lambda x = \zeta \cdot x \tag{3.8}$$

where $x = [x_1, x_2, ..., x_n]$, and G_i is the set of neighbors for node i. The eigenvector centrality is the vector associated with the largest eigenvalue for the adjacency matrix ζ.

[2]Note that, as explained in (Newman 2007), the definition may seem awkward and tautological.

Eigenvector centrality can also be used in parameter dimension reduction of a policy function. We illustrate this by an example. Suppose we have a five by five regular grid, as shown in Fig. 3.2. Looking at the figure, one can observe six types of locations as distinguished by their eigenvector centrality. In a policy function, one can utilize this symmetry by assuming each type of site has a different set of parameters to be estimated instead of one for each site, which would result in a significant reduction in the dimension of parameters to be estimated. For example, suppose we have a policy function with one parameter for each site, then we would have a total of 25 parameters. If we constrained the parameters to be the same for each site type, then this would reduce the number to 6 parameters.

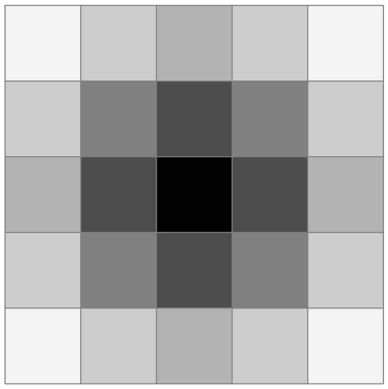


Figure 3.2: A 5x5 grid model. Sites with same eignevector centrality have the same degree of grey shading.

3.6 Issues and Challenges

There are several issues and challenges of the rank-based policy rule, as stated in Eq. (3.5) and the estimation method in Eq. (3.1). The first one is that the policy rule suffers from the identification issue. The Identification issue is a result of the ordinal nature of the policy rule. To illustrate this, and given a specific state S, let j denote the index of the site with the maximum value of ϕ as calculated in Eq. (3.4). For any δ such that $|\delta| < min(\phi_j - \phi_i) \forall i$, perturbation of the values of ϕ_i by δ will still preserve the rank of site j being the maximum. Additionally, given fixed values of ϕ_i, the vector of α_i for each site i is, generally, non-identified. The identification issue should not be deterrent from using such a policy rule. Some widely and successfully used function approximation methods, such as neural networks, suffer from such an issue (Albertini and Sontag 1993). Solving or reducing the severity of it, however, may improve the efficiency of the estimation process.

In certain problems, there could be redundancy in the parameters that contribute to the identification issue. This is the case in our regular grid situation, where the spatial structure is symmetric. We can resolve this redundancy by constraining the sites that have the same eigenvector centralities to have the same parameters, as illustrated in the previous section.

The specification of the policy rule in Eq. (3.5) does not illustrate how ties could be broken-up. In other words, in case two sites have the same values of ϕ, which site do we choose? One way is to break the ties by choosing a site randomly. Another way is to choose the one that appears first in the list of sites ranks. For computational convenience and efficiency, we choose the second away.

Our simulation-based estimation approach requires setting certain hyper-parameters such as the number of replications and the time horizon. We choose the number of replications, such that the variance of the estimated value function is small enough. In our estimation stage, where we use simulation to estimate the parameters of the policy rule, we set it to a hundred replications. In the evaluation stage, we set the number of replications to ten thousand in order to make the simulated time paths smoother. In all our analyses, we use the simulation-based expected number of occupied sites at each time period over the number of replications as our policy performance measure. We use fewer replications in the estimation stage than the evaluation stage as it is very computationally intensive since the optimizer calls our simulator thousands of times.

The other thing to decide on is the time horizon. Note that we can express the infinite horizon value function (V_∞) following a policy π as

$$V_\infty^\pi = \sum_{t=0}^{\infty} \beta^t E R_t^\pi \approx \sum_{t=0}^{T} \beta^t E R_t^\pi + \beta^{T+1} \sum_{t=0}^{\infty} \beta^t \tilde{R}^\pi = \sum_{t=0}^{T} \beta^t E R_t^\pi + \frac{\beta^{T+1} \tilde{R}^\pi}{1-\beta}, \tag{3.9}$$

where R_t^π is the reward at time t following policy π and $\tilde{R}^\pi$ is the expected equilibrium level following policy π. The time horizon (T) is chosen, such that the second term in (3.9) is negligible. In our situation, T is set to 200.

3.7 Policy Evaluation

In order to test or explore the performance of our proposed policy function, we need to use some evaluation methods. The literature seems to lack a thorough treatment of evaluation methodologies for approximate dynamic programming methods. Nevertheless, Powell (2008) states three methods where some or all may apply to the problem of interest.

The first method is to compare the policy obtained using the approximate dynamic programming methods, such as the stochastic optimization in policy space, with optimal policy for a simplified or smaller version of the problem. For example, in a spatial model, we can compare the policy rule obtained using approximate methods with the optimal rule by solving a problem with a number of sites up to a level where it can be solved optimally.

The second method is to solve a deterministic version of the problem optimally and to use approximate methods and compare those two policies. In some problems, the deterministic version of the problem allows one to solve a larger problem using other optimization methods.

The third method is to compare the approximate policy rule with a myopic or greedy rule. For example, in our spatial model, the myopic rule is to take action on the site that minimizes the expected next period damages.

In our analysis, we apply the first and third evaluation methods but not the second one. We avoid the second one as there does not seem to be an obvious way to get a deterministic version of the model without making some critical assumptions. For example, in our model, the diffusion of pests to a site depends on its infected neighbors in a probabilistic way, that is the higher the number of infected neighbors, the higher is the probability of that site to be infected next period. If we are to make this part of the model deterministic, then we need to decide on a threshold such that the site becomes infected if the number of infected neighbors exceeds the threshold. This is tricky and may result in misleading conclusions on how the policy might do in the stochastic situation. Another difficulty is that the way we would be able to solve a larger version of our deterministic model is through integer programming (Epanchin-Niell and Wilen 2012), which means that the solution obtained will be defined as a sequence of actions for every period. It is not apparent how would one convert this, if even possible, to a policy function that is defined for every possible state.

3.8 Analysis

We will start our analysis with an extreme network of four isolated sites. This aims to familiarize the reader with the idea of this research and the methodology. Additionally, we use it to illustrate some concepts.

Next, we aim at exploring the performance of our approximate policy for larger networks. We will explore both regular grid networks and Chadès et al. (2011) multi-layer star network. The objective of this exploration is to seek answers to several main questions. The first is whether the stochastic optimization method with our non-linear policy function would work or not. By this, we mean that even if the approximate policy has a correct functional form in the sense that it could result in the optimal policy, the optimization method may or may not result in good estimates of the parameters. In analogy to a simple linear regression model, suppose we know the correct model or the data generation process. If we do not use a good estimation method (i.e., OLS), then we may not get a good estimate of our correctly specified model. The second point is that if it works, then how good is it compared to other baseline policies, and how would it perform when the problem increases in size.

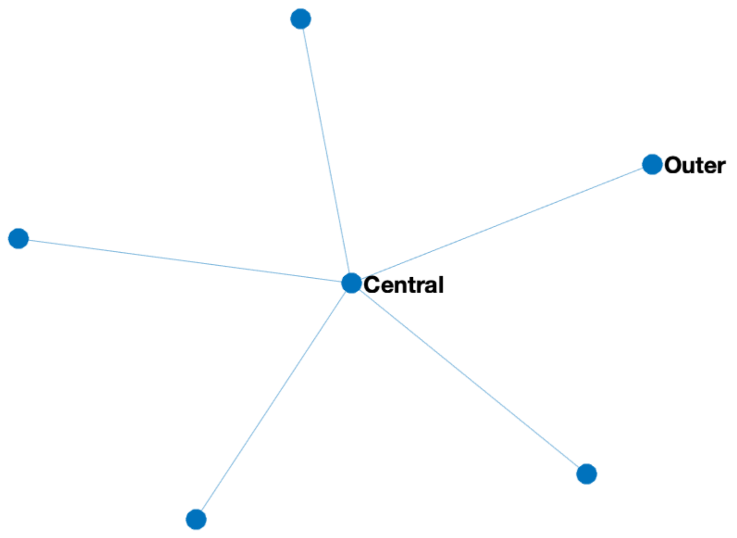


Figure 3.3: A star network.

To be able to answer those questions, we need to define features for the policy rule in Eq. (3.5), and define baseline comparison policy rules. The features that we will use in the policy rule are the state of the site itself, and the number of infected neighbors interacted with the state of the site. That is

$$SiteIndex_i = \alpha_{1i} \cdot S_{1i} + \alpha_{2i} \cdot S_{1i} \cdot q_i, \tag{3.10}$$

where q_i is the number of infected neighbors. We selected those features as they generalize the inside-out and outside-in rules of thumb used in the literature, and explained next

(Chadès et al. 2011). Additionally, we believe that the rule also generalizes the Chadès et al. (2011) rule of thumb. We illustrate this on the star network mentioned in Chadès et al. (2011) and shown in Fig. 3.3. An inside-out rule prioritizes the treatment of the central node over the outer nodes, that is central node is treated first. Let i represents the outer nodes, and j represents the central node, then to implement the inside-out using our rule, we can set α_{1i}, α_{1j} and α_{2i} to 1 and α_{2j} to 1 + a small number. The outside-in rule does the opposite: which is to treat the outer nodes first, then the central ones. This can be implemented by setting α_{2i} to -1 and α_{2j} to -1- a small number, while setting α_{1i} and α_{1j} to 1. Chadès et al. (2011) rule of thumb treat half the outer nodes first then the central node then continuing treating the rest of the outer nodes. To implement it we can set α_{1i} to 1 α_{1j} to 3 , and α_{2i} and α_{2j} to -1.

We will be using four policy rules for comparison. The first one is the optimal policy, and this will only be used in the largest problem that can be solved optimally, which is the four by four grid problem. The second is the myopic or greedy policy, which is to treat the site that would minimize the expected damages in the next period. Mathematically, the greedy policy is

$$Greedy(S) = \min_A E\left[\sum_i^n (S_i^+)|A,S\right] = \min_A \sum_i^n S_i^+ \cdot P(S_i^+ = 1|S,A), \tag{3.11}$$

where S is a vector of the state of each site, S_i is the state of site i, A is an integer number that represents the site to be treated. The third rule is an inside-out rule, where the most central site, according to the eigenvector centrality, is treated first. The fourth is the outside-in rule, where the least central site is treated first. We will use two variations of the third and the fourth rules. The first prioritizes the treatment to the site with the highest number of infected neighbors, while the second prioritizes it to the site with the least number of infected neighbors. The third and fourth policy rules are special cases of the policy in Eq. (3.5) where we set the coefficients at certain values, as has been illustrated in the previous paragraphs.

It should be noted that in our analysis, we assume that cost is irrelevant as we assume that only one action per period is allowed. The one action per period assumption allows us to solve the problem optimally, for comparison purposes, for a relatively larger number of sites (about 16 sites). Model baseline parameters are shown in Table 3.1. In all our analyses, we use the expected number of occupied sites (i.e., the damage) per period as a measure of the performance of the policies. The default initial state for our simulations is the state where all sites are being occupied. We will explicitly state the initial states for other cases that we explore as well.

Table 3.1: Baseline and adjusted model parameters.

Parameter	Base values
p_0	0
p_1	0.15
p_n	0.1
p_t	0.7,1
β	0.95
T	200

3.8.1 Isolated Network

To do a basic check of the proposed solution method and to illustrate and make some points more concrete, we start by solving a simple and trivial problem. The problem consists of four isolated sites with the setup mentioned previously, and the model parameters are set according to the base values in Table 3.1.

This problem is trivial as a preventive action, which treats a site in order to prevent the spread of infection, plays no role. This is because sites are isolated from each other, and hence an optimal solution is trivial, which is to treat one of the occupied sites at each period.

We solve this problem both optimally and using the approximate policy. There are several points to illustrate. First, looking at Table 3.2, we can see that both the optimal and the approximate policy solution give what we expect, which is to treat an infected site.

It should be noted that the approximate policy does not seem to break ties in the order, as mentioned in the previous section. The reason for this is that, although one expects ties for states that have more than one infected site, the coefficients of the approximate policies, as shown in Table 3.3, for some sites are different, resulting in no ties in the rank for some states. This also illustrates the identification issue in our policy rule.

Second, if we look at the long-run distribution in Table 3.2, computed using the presented optimal solutions, we see that complete eradication is certain. This means, in the long run, and under the optimal policy, most states are never visited. It should be noted that eradication is certain since spontaneous infection is ruled out in our setup. This necessitates that we start with some sites being infected in the estimation of the approximate policy. If we do not do so and alternatively started the simulation with all sites being uninfected, then regardless of what policy we implement, the sites will stay uninfected. This is because the objective function, as in Eq. (3.3), will be the same for any parametric values, which is zero.

Table 3.2: Policy rule: optimal versus approximate policy.

				Action		
S_1	S_2	S_3	S_4	Optimal	Approximate Policy	Long Run Distribution
0	0	0	0	0	1	1
0	0	0	1	4	4	0
0	0	1	0	3	3	0
0	0	1	1	3	4	0
0	1	0	0	2	2	0
0	1	0	1	2	2	0
0	1	1	0	2	2	0
0	1	1	1	2	2	0
1	0	0	0	1	1	0
1	0	0	1	1	4	0
1	0	1	0	1	1	0
1	0	1	1	1	4	0
1	1	0	0	1	2	0
1	1	0	1	1	2	0
1	1	1	0	1	2	0
1	1	1	1	2	2	0

Table 3.3: Estimated coefficients of the policy in Eq. 3.10 It should be noted that we have infinite solutions (i.e., any positive coefficients result in the same policy).

Coefficient	S_1	S_2	S_3	S_4
α_1	0.4444	0.6667	0.4444	0.6667
α_2	0	0	0	0

3.8.2 Grid Networks

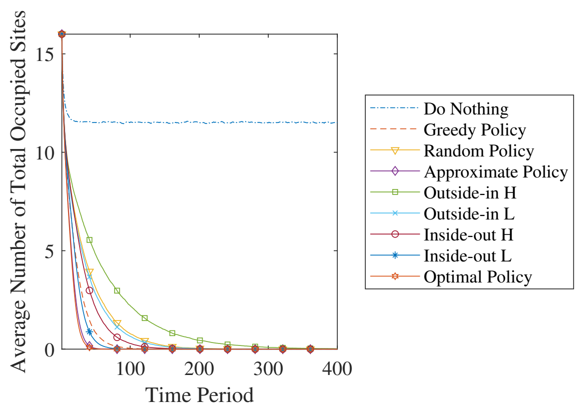
In this section, we analyze the policy rule in Eq. (3.10) on three regular grid networks: 4x4, 6x6, and 8x8. We constraint the coefficients of the policy rule for the sites with the same eigenvector centrality measures to be the same as described in the previous section. We analyze the policy rule in two versions of the system model. The first assumes that action is partially effective , that is $p_t = 0.7$, while the other assumes that action is perfectly effective $p_t = 1$.

Looking at Figs. 3.4 and 3.5, we find that the approximate policy generally does well compared to other policies, which suggests that the simulation-based estimation works for our non-linear policy rule. As the size of the network increases, the gap increases between the approximate policy and the greedy policy.

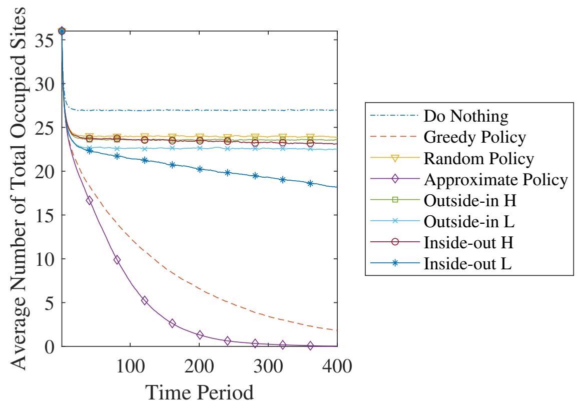
By constraining the number of coefficients to be estimated using the eigenvector centrality, we significantly reduce the dimension of the policy function, and hence the burden on the optimizer. The difference between the dimensions of the constrained and unconstrained policy functions for different grid sizes is illustrated in Table 3.4.

Table 3.4: Dimension of the constrained and unconstrained policy functions for different grid sizes. Exploiting spatial symmetry in the problem by using eigenvector centrality significantly reduces the number of parameters to be estimated.

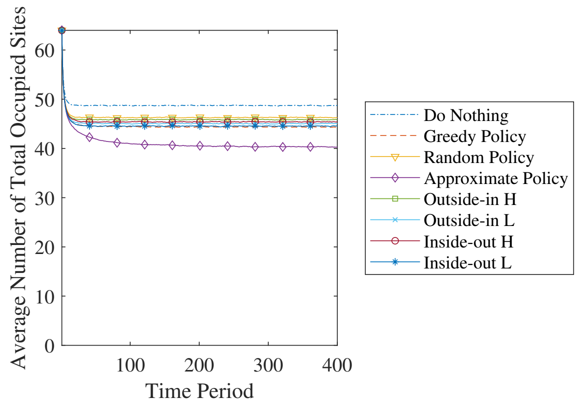
Grid Size	Policy Function Dimension	
	Unconstrained	Constrained
4x4	32	6
6x6	72	12
8x8	128	20



(a) 4x4 grid network.

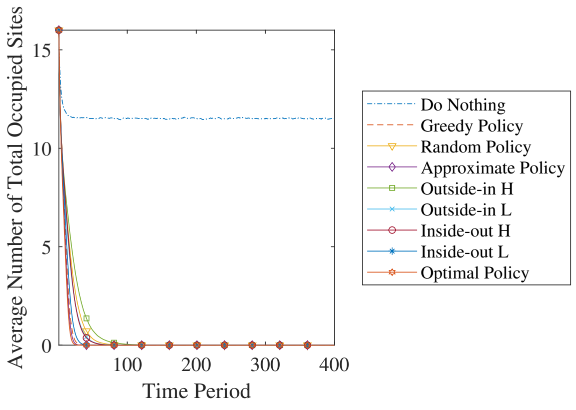


(b) 6x6 grid network.

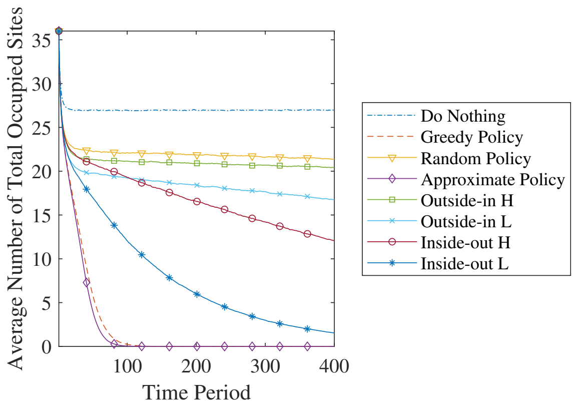


(c) 8x8 grid network.

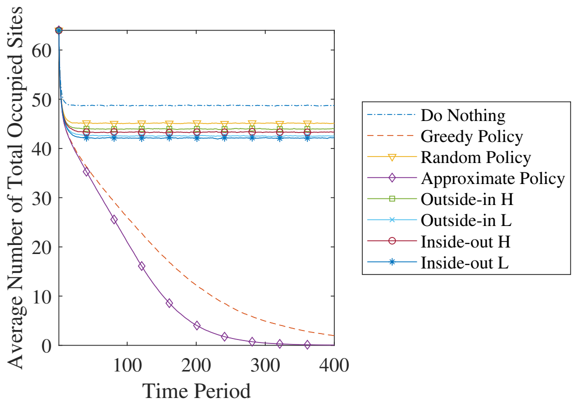
Figure 3.4: Time path simulations for a 4x4, 6x6 and 8x8 grid network where actions is assumed to be partially effective ($p_0 = 0, p_1 = 0.15, p_n = 0.1, p_t = 0.7$). The time paths represent an average over a ten thousands replications. The gap between policies widens as the grid size increases.



(a) 4x4 grid network.



(b) 6x6 grid network.

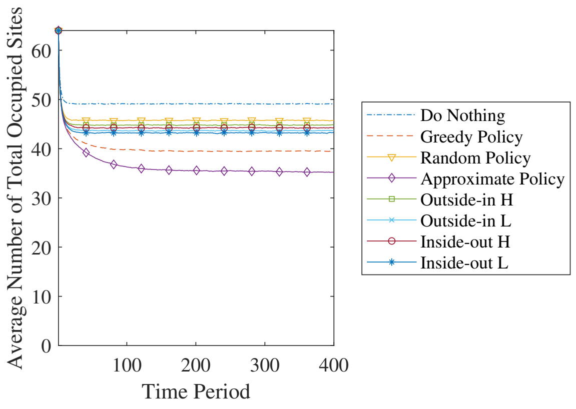


(c) 8x8 grid network.

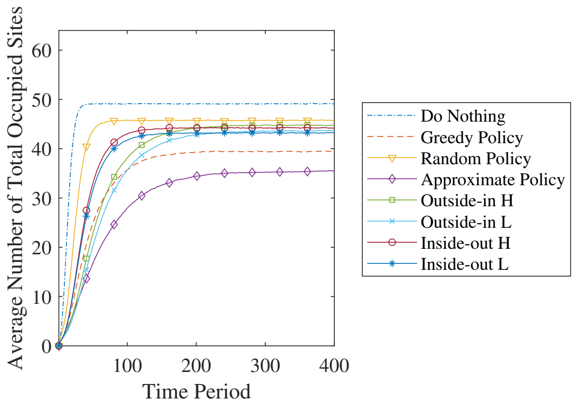
Figure 3.5: Time path simulations for a 4x4, 6x6 and 8x8 grid network where actions is assumed to be completely effective ($p_0 = 0, p_1 = 0.15, p_n = 0.1, p_t = 1$). The time paths represent an average over a ten thousands replications. The gap between policies widens as the grid size increases.

The performance of the policy function crucially depends on the parameters of the system model. For example, introducing some probability of spontaneous infection ($p_0 = 0.01$) eradication to be unachievable, as shown in Fig. 3.6 where the equilibrium level has increased to around 35. This effect is due to the fact that spontaneous infection is propagated and amplified through the system by the diffusion process (i.e., infected site infects their neighbors, which themselves infect their neighbors, and so on).

One might expect that the initial state plays a role in the policy rule's performance in terms of the achieved equilibrium levels. Our analysis shows that this is not necessarily true. Figure 3.6b shows that even if the system starts with an extreme case where there are no infected sites, the equilibrium level is still the same as when it starts with all sites being infected. This suggests that our system is recurrent; that is, the equilibrium distribution of states is independent of the initial state (Howard 1960). Nevertheless, there are situations where the system is not recurrent. This is apparent when we study the sensitivity of the policies and the system to the changes in the spontaneous infection parameter (p_0).



(a) Inital state is where all sites are infected



(b) Inital state where all sites are uninfected

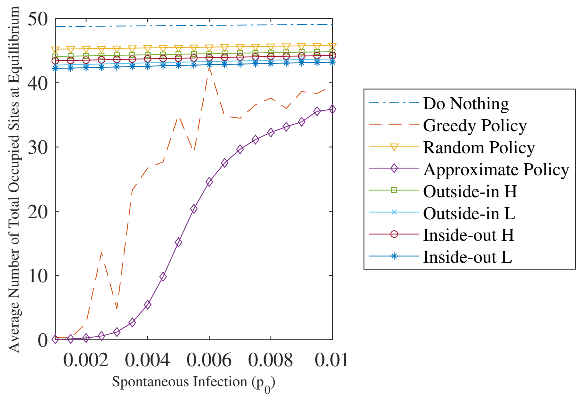
Figure 3.6: Time path simulations for 8x8 grid network where actions is assumed to be completely effective, and spontaneous infection is set to small non-zero value ($p_0 = 0.01, p_1 = 0.15, p_n = 0.1, p_t = 1$). The time paths represent an average over a ten thousands replications. The results show that the system is likely to be recurrent. That is, the initial state doesn't affect the equilibrium level.

Figure 3.7 shows sensitivity analysis of the effect of different values of the spontaneous infection parameter on the average equilibrium of occupied sites. Figure 3.7a is for the situation where the initial state is such that all sites are being infected, while Fig. 3.7b is for the situation where the initial state is such that all sites are being uninfected.

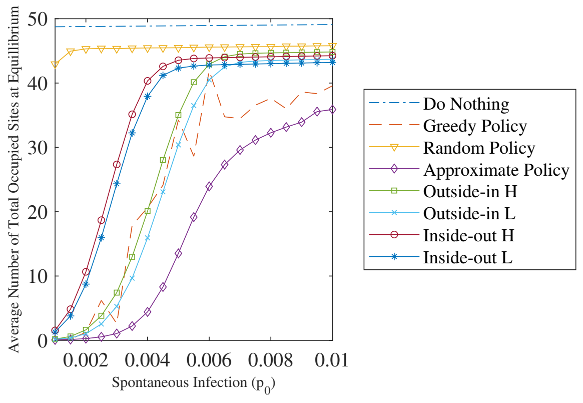
In the first situation, all policies are unable to achieve complete eradication except for the greedy and approximate policies, and that is for certain small values of spontaneous infection. The approximate policies achieve almost a complete eradication for values of p_0 less than 0.003. The gap between the approximate and greedy policy, in terms of equilibrium level, appears to be largest when p_0 is between 0.004 and 0.006. The performance of the greedy and approximate policy, in terms of achieved equilibrium levels, degrades as p_0 increases. The equilibrium level starts to level off for values of p_0 greater than 0.006.

In the second situation, the performance of both the approximate and the greedy policies are comparable to the first situation. An interesting thing to note is that both the outside-in and inside-out policies are now effective in some range of the spontaneous infection parameter (p_0). Both the outside-in policies, the ones that prioritize sites with less infected neighbors (outside-in L), and high infected neighbors (outside-in H), do better than the inside-out policies. The rationale is that the outside sites have lower probabilities of being infected in the subsequent periods. Additionally, clearing the outside sites first will also lower the probability of the inner sites to get infected once clear.

An important thing to note is that while all policies are monotonic with changes in spontaneous infection parameter (p_0), the greedy policy is not. The greedy policy, as shown in Eq. (3.11), is a direct function of the model parameters. We examined it for the different values of p_0, given specific states, and have found that it results in jumps in actions. Our investigations of the approximate policy reveal that the estimated coefficients are equivalent for all parametric values of the spontaneous infection (p_0).



(a) Inital state is where all sites are infected.



(b) Inital state where all sites are uninfected.

Figure 3.7: Sensitivity analysis of the spontaneous infection parameter(p_0) for an 8x8 grid case. Other parameters are set to the baseline ($p_1 = 0.15, p_n = 0.1, p_t = 1$). The average is calculated over a ten thousands replications. Equilibrium levels increase as p_0 increases. The initial state appears to play an important role in the effectiveness of both the outside-in and inside-out heuristic rules.

Another thing to note is that the performance of the policies degrades as the grid size increases. This is expected, as we only take one action per period. The effectiveness of action (p_t) is also very crucial. Eradication takes noticeably longer when action is partially effective ($p_t = 0.7$). In the 8 x 8 grid situation, eradication is not achievable. However, when action is completely effective ($p_t = 1$), eradication is achieved in all grid sizes under study.

The variability of the number of occupied sites given a specific control policy is an important aspect. Our results represent an average of the total number of occupied sites over ten thousand time replications. This means that the number of occupied sites can vary across the time paths for each implemented policy. Figure (3.8) shows that the standard deviation of the total number of occupied sites for effective policies, such as the approximate policy, tends to increase during eradication periods and decreases as it gets close to complete eradication.

Variability is greatest when there are a moderate number of occupied sites, and this gives the greatest scope for change. For example, when the system starts with all sites being infected, then in the next period, we can only have at most one site being cleared due to treatment. The variance in this situation will be minimal. However, when we start with a situation where more sites have already been cleared, then in the next period, we can clear at most one site. Additionally, the already cleared sites can get infected due to the neighboring infected sites, which results in a higher variance. When most sites are cleared, then fewer sites can be infected due to neighboring infected sites, and this results in a reduction in the variance.

Our analysis shows evidence that our proposed policy, generally, outperforms other common heuristic policy rules used in the literature. The sensitivity analysis in the appendix shows that it consistently outperforms other comparison policies. In most situations, most of the comparison policies cannot achieve eradication except in the case where the diffusion

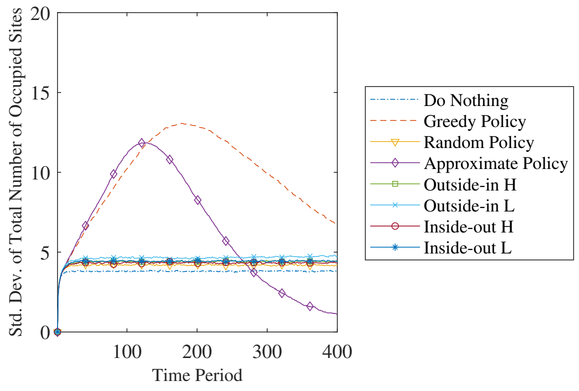


Figure 3.8: The standard deviation for the number of occupied sites. Model parameters values are $p_0 = 0, p_1 = 0.15, p_n = 0.1, p_t = 1$. The time paths represent an average over a ten thousands replications.

probability due to infected neighbors is relatively low ($p_1 = 0.1$), and the probability of a site to naturally clear is high ($p_n = 0.1$). In such a situation, both versions of the inside-out and outside-in policies can achieve eradication. The outside-in policies outperform inside-out policies.

Although we have tested our policy rule on a specific stylized model, its flexible nature makes it potentially suitable for many other applications of network or spatial nature problems. Additionally, it can help in deducing a heuristic rule that applies to a larger problem, as we illustrate in the following section.

3.9 Heuristic Rule For Grid Space

Our analysis in the previous sections suggests a simple heuristic policy rule for a regular grid space. The policy rule is a priority that first treats the upper left corner of the first column moving downward, then once the first column is clear, we move to the next one and so on. This is illustrated in Fig. 3.9.

There are several reasons why this rule appears to work well. First, the corner sites have the least number of neighbors, which is two. This means that once they are treated, they have the lowest probability of being infected in the next period. Second, once a corner site is cleared, the next site below will will have the least number of infected neighbors. Once all sites are cleared in the first column, each will have only one infected neighbor.

When a rectangular grid (i.e., non-square) is considered, then the rule must be modified. It is important that we start with the shorter side of the grid. For example, in a 4x6 grid, as in Fig. 3.9, we start with the side that has four sites rather than the one with six sites. The reason for this is that once the side is cleared, then the probability of at least one site getting reinfected in the longer side is higher than the shorter side.

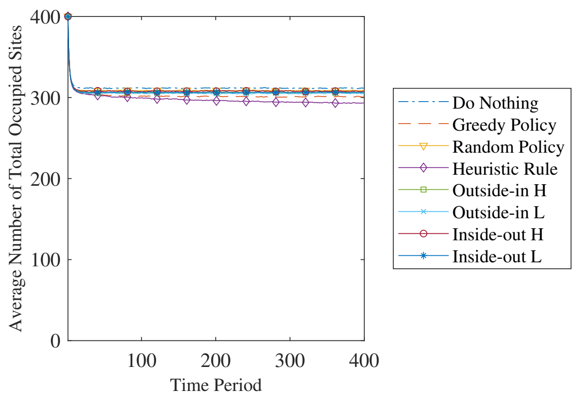
To explain this, first assume that we start with all sites are being infected. Once all sites are cleared on one side, then all of them will have the same probability of being infected in the next period. Let p_o represents the probability that a cleared site is infected in the next period after all sites on that side are cleared. The probability that at least one of the cleared sites gets reinfected is $1-(1-p_o)^{N_l}$, where N_l is the number of sites of side l. The probability is increasing with the number of sites on side l (N_l). It should also be noted that a site that gets reinfected increases the probability of the neighbors to become infected, as the cleared neighbors will now have two infected neighbors instead of one.

1	5	9	13	17	21
2	6	10	14	18	22
3	7	11	15	19	23
4	8	12	16	20	24

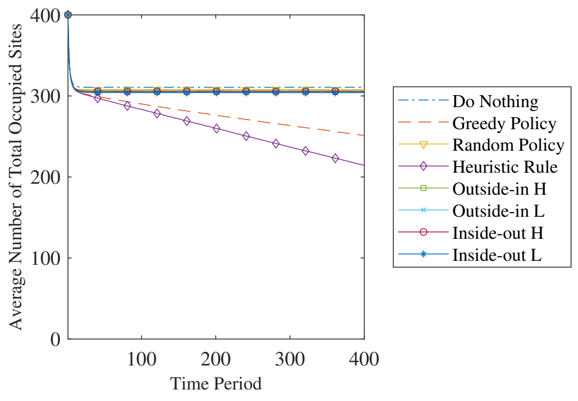
Figure 3.9: An example 4x6 grid that illustrates the heuristic priority rule. It also illustrates how sites are ordered in our simulation code.

3.9.1 Simulation Results

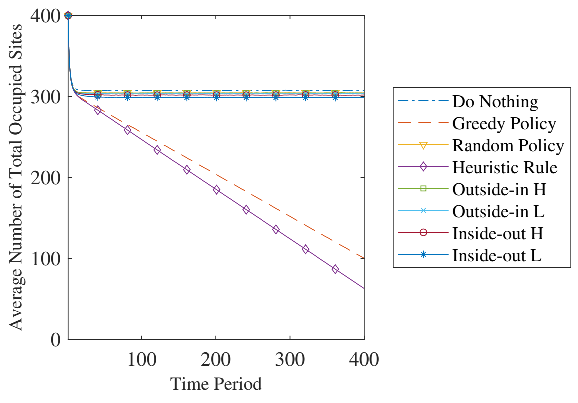
We simulate our heuristic rule for a 10x40, 8x50, and 5x80. All cases consist of four hundred sites. We conclude several points. First, the heuristic rule does better than all comparison policy rules in all cases. Second, the length of the grid side where we start the treatment (the shortest side in case) plays a critical rule in the performance of the policy. This also shows evidence of our argument, where prioritizing treatment of shorter side is better than prioritizing treatment of the longer one. Third, the greedy policy appears to be affected by the dimension of the grid. It should be noted that the greedy policy takes action that minimizes the damages in the next period. In some states, such as when all sites are infected, the greedy policy can result in multiple equivalent actions. We break the ties by choosing the site that appears first in the list of greedy actions, starting from the left side, as shown in Fig. 3.9. Hence, the grid dimension can affect the greedy policy as well.



(a) 10x40 grid network.



(b) 8x50 grid network.



(c) 5x80 grid network.

Figure 3.10: Time path simulations for a 5x80, 8x50 and 10x40 grid network where actions is assumed to be completely effective ($p_0 = 0, p_1 = 0.15, p_n = 0.1, p_t = 1$). The heuristic policy performance improves as the length of the shorter side decreases.

3.9.2 Heuristic Rule in Clustered Situation

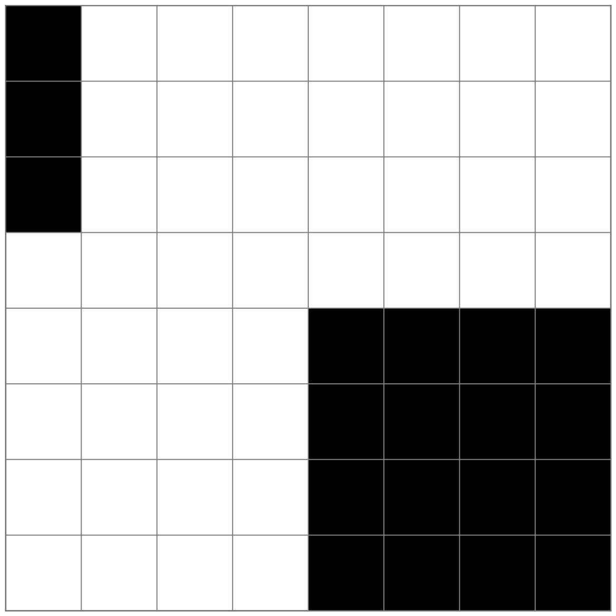
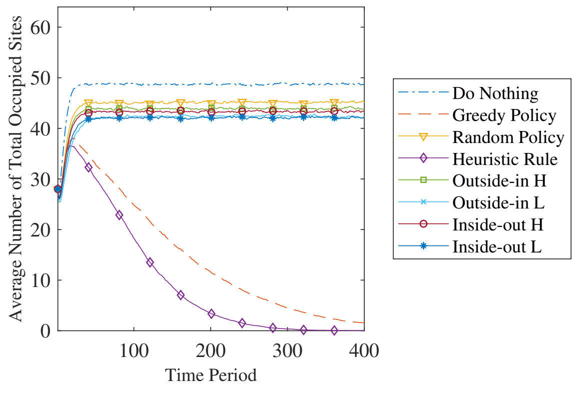


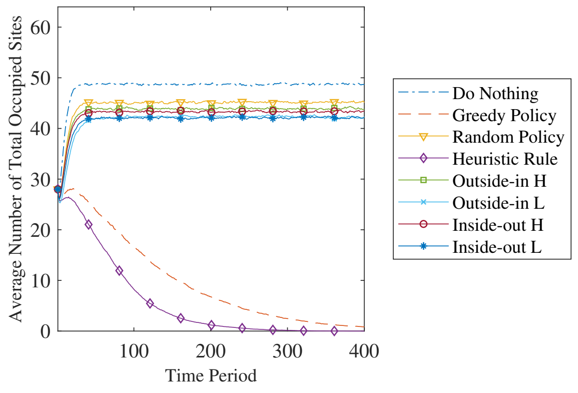
Figure 3.11: An illustration of an 8x8 grid with clustered infected sites. Infected sites are black shaded.

We explore our heuristic rule in the situation where we have a clustered infested grid. We explore two versions of the rule. The first prioritizes the treatment of the least clustered ones. The second prioritize it to the most clustered ones. The first case can be implemented by prioritizing our heuristic rule, starting from the most northwest site while the second one prioritizes it from the most southeast site. It should be noted that in general situations, one needs to apply some clustering methods.

Simulation results are shown in Fig. 3.12. Our heuristic rule outperforms other comparison rules. Estimating our approximate policy rule using the clustered situation as an initial state gives the same results. Our heuristic rule can be improved for clustered situations, and that is by prioritizing a less dense cluster. The rationale is that sites in less dense clusters tend to have a lower probability of being infected in subsequent periods once treated. Prioritization over clusters is a problem that deserves a detailed exploration in future research.



(a) Large cluster prioritization case.



(b) Small cluster prioritization case.

Figure 3.12: Effect of clustered initial infestation in an 8x8 grid case. Parameters are set to the baseline ($p_0 = 0, p_1 = 0.15, p_n = 0.1, p_t = 1$). The average is calculated over a one thousand replications. Prioritization of the small cluster results in a slightly better performance.

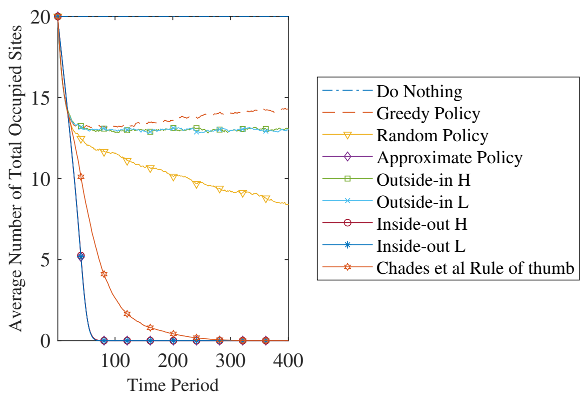
3.10 Star Network

Star networks are considered the fundamental element in the analysis of Chadès et al. (2011) work. In this section, we first analyze different sizes of simple star networks with the model parameters used in Chadès et al. (2011). Table 3.5 shows the model parameters. We then analyze Chadès et al. (2011) two-layers test network. We compare our approximate policy with our comparison policies in addition to Chadès et al. (2011) rule of the thumb. Their rule of thumb states that we first prioritize the treatment of half of the satellite infected nodes then the central node.

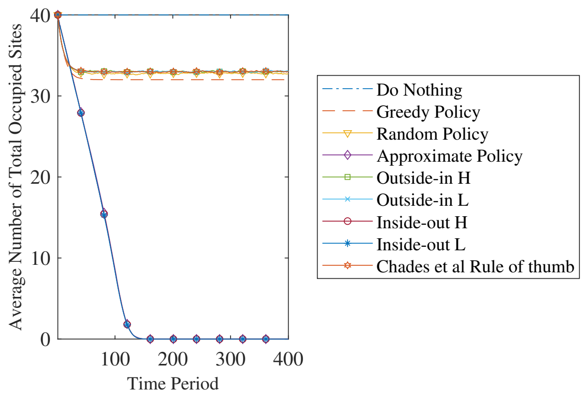
Table 3.5: Chadès et al. (2011) star network baseline model parameter.

p_0	p_1	p_n	p_t
0	0.1	0	0.7

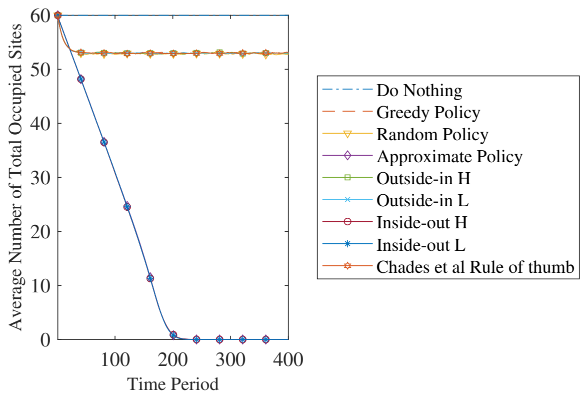
Our analysis on a star network with 20, 40, and 60 nodes shows that our approximate policy results in an inside-out rule that outperforms other comparison policies. Our policy also achieves eradication, while others do not achieve it except for Chadès et al. (2011) rule in the 20 nodes case. Table 3.6 shows the estimated policy coefficients as shown in Equation (3.10). Figure 3.13 shows the time path simulation results. Our approximate rule prioritizes the treatment of the central site. In other words, when the central node is infected, we treat it; otherwise, we treat the satellite nodes. By doing so, we ensure that satellite nodes will not have infected neighbors (i.e., the central node) and hence cannot get reinfected as we treat them.



(a) Twenty nodes case.



(b) Forty nodes case.



(c) Sixty nodes case.

Figure 3.13: Time path simulations for a 20, 40, and 60 nodes star network ($p_0 = 0, p_1 = 0.1, p_n = 0, p_t = 0.7$).

Table 3.6: Estimated coefficients of the policy in Eq. 3.10. The policy rule prioritizes the treatment of the central sites (i.e., it is an inside-out rule).

Node	α_1	α_2
Satellite	0.6667	-0.6667
Central	0.6667	0

3.10.1 Multi-layer Star Network

Chadès et al. (2011) has used the multi-layer start network with 101 nodes, as shown in Fig. 3.14 as the main test case for their rule of thumb. We analyze our approximate policy on the same network with their baseline model parameters. Figure 3.15 shows the time path simulation results for the baseline model.

Our approximate policy outperforms all other comparison policies, including Chadès et al. (2011) rule of thumb. Table 3.7 shows the estimated coefficients. It should be noted that because of the network symmetry, there are only three different types of nodes. By using the eigenvector centrality, we reduce the number of parameters from two hundred (two per node) to only six parameters (two per type). This reduction in parameters can significantly reduce the computational burden on the optimizer. Our sensitivity results in the appendix show that our approximate policy consistently outperforms other comparison policies.

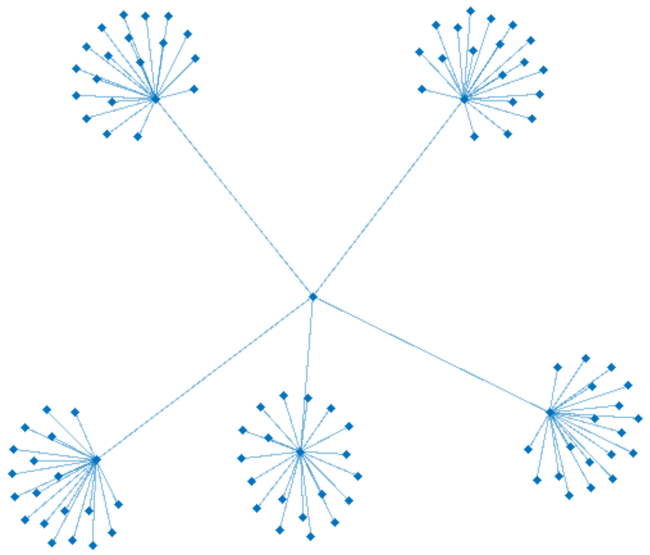


Figure 3.14: Chadès et al. (2011) test network.

Table 3.7: Estimated coefficients of the policy in Eq. 3.10. The rule prioritizes the outer central node over the satellite node, and the most central one over the outer central one as long as it has no more than one neighbor.

Node	α_1	α_2
Satellite	0.6667	-0.2222
Outer Central	0.6667	0
Most Central	0.9630	-0.2222

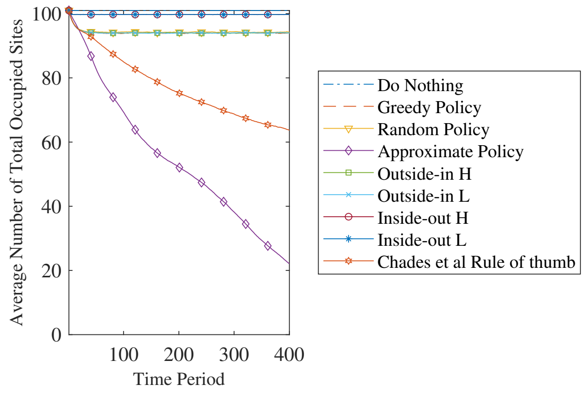


Figure 3.15: The average number of occupied sites per period for a multi-layer star network with 101 nodes ($p_0 = 0, p_1 = 0.1, p_n = 0, p_t = 0.7$).

3.11 Conclusion

Our analysis of our proposed policy shows that it consistently does better than our comparison policies. In the grid situation, both the greedy policy and our approximate policy can achieve eradication when there is no or very low probability of spontaneous infection. The differences between the two policies widen as the grid size increases. Our results for the grid case show that the approximate policy can be expressed in a simple heuristic rule that only depends on the state of the sites and the treatment's order. While this may be true for the densely infected unclustered grid, our further analysis of clustered situations shows that our rule consistently outperforms our comparison rules.

Our approximate rule in Chadès et al. (2011) multi-layer star network case outperforms the comparison rules. Both Chadès et al. (2011) rule of thumb, and our approximate policy can achieve eradication. However, the gap between the two policies is very wide Other policies, including the greedy policy, cannot achieve eradication. Our approximate policy rule in the star network model is more complex and depends on both the state of the node and the number of infected neighbors.

Our sensitivity analysis illustrates several points. First, the policies' performance improves as the probability of sites to naturally clear (p_n) or the action effectiveness (p_t) increases. Second, the performance degrades as the probability of a site to get infected due to neighboring sites (p_1) increases. Third, spontaneous (p_0) infections increase the equilibrium level of the average number of occupied sites.

Despite the good performance we attain in our analysis, the stochastic optimization in the policy space approach, combined with our rank-based policy rule, has some potential issues and problems. First, the success of the approach relies on the optimizer used. This means that even if we have a rank rule with a good set of features, the optimizer may not necessarily converge to a good solution, or it may take prohibitively substantial time before it converges to a good solution. Hence, it is important that one uses a robust optimizer global optimizer. Second, the choice of features plays an important rule and can be complicated, especially for a spatial problem with discrete variables. For example, incorporating a feature that prioritizes the treatment based on the clustering of the occupied sites is not obvious. In addition to that, features have to be calculated in a very efficient way.

A major contribution in this research, in addition to the testing of the stochastic optimization in the policy space method, is proposing a method of exploiting symmetry in spatial and network models. Our approach relies on the use of eigenvector centrality measure. We have shown that it can result in a significant reduction in the number of parameters in the policy rule and hence decrease the computational burden on the optimizer.

CHAPTER

4

REINFORCEMENT LEARNING METHODS

4.1 Introduction

Dynamic programming is a widely used optimization method in the resource economics and ecology fields. Marescot et al. (2013) provides a primer and some applications of the dynamic programming from the literature. There are two issues with dynamic programming. First, it suffers from what is known as the curse of dimensionality; that is, the state space increases exponentially with the number of state variables. This makes the dynamic programming problem with large state variables unsolvable. The second problem is that dynamic programming requires the calculation of the expectation of the value function of the future states, which may not be possible with large dimensional problems.

One way to get around these issues is to use approximate dynamic programming techniques. Powell (2011) provides a detailed exposition to such techniques. This chapter focuses on reinforcement learning (RL) methods, such as Q-learning, which is widely used in robotics and artificial intelligence (AI), but less in resource economics and ecology fields. Approximate dynamic programming methods are know as reinforcement learning (RL) methods in computer science (Bertsekas 2019).

The treatment of this chapter will be as follows. First, we provide an overview of reinforcement learning, describe the Q-learning algorithms, and illustrate how it works using a simple single site pest infestation MDP model. This will familiarize the reader with the method as well as illustrate some key points and issues that apply to many other RL algorithms as well. Second, we provide a review of recent variations of the RL algorithms in the literature. The literature review will focus more on the algorithms that are likely to be suitable for planning problems, where one seeks an optimal or near-optimal policy rule in order to achieve a specific objective over time without the need to update the rule frequently. Third, we explore and test the Least Square Policy Iteration (LSPI) (Lagoudakis and Parr 2003) on our spatial problem that has been introduced in the previous chapter.

4.2 Overview

The fundamental idea of reinforcement learning traces back to psychology, more specifically in the study of how animals learn by trial and error (Sutton and Barto 1998). This idea is attributed to Thorndike (1911), which basically states that actions by animals that result in good outcomes are more likely to be taken when the situation occurs again, while ones that result in bad outcomes are less likely to be taken. In his dissertation, Watkins (1989) provided a unified view of reinforcement learning. He also proposed an algorithm that solves dynamic control problems, which he calls it Q-learning. Since the introduction of Q-learning, reinforcement learning techniques has witnessed a large use and research in the area of artificial intelligence (AI) and machine learning (Sutton and Barto 2018).

This raises a question: why reinforcement learning methods have witnessed a large use in the field of artificial intelligence but much less in resource economics, given that both fields involve some types of control problems? In other words, what are the differences between control problems in artificial intelligence and resource economics that potentially slows down the penetration of the reinforcement learning method to resource economics?

One significant difference is that resource economists tend to have an explicit mathematical system model of their problems, while in artificial intelligence problems, an explicit system model may not exist. For example, the environment of a robot may not be known or may change depending on where the robot is deployed. This makes solution methods, such as dynamic programming, which requires an explicit model, in-feasible. Reinforcement learning methods, such as Q-learning, is a way that mitigates this issue (Sutton and Barto 2018).

A second difference is that online problems where one needs to frequently update the policy rule are more common in the field of artificial intelligence. For example, a robot in a new environment would have to update its policy frequently as it explores the new environment. However, in economics, the environment is assumed to be known. Additionally, observations from the environment occur less frequently. An important aspect of many economics problems is that interaction with the environment for learning purposes can result in irreversible situations. For example, relaxing some hunting or fishing policies in order to learn better management policies can drive certain species to extinction.

4.3 Literature Review

The fact that many of the RL methods have been developed within a community interested in online problems, where one would follow a policy rule and frequently improve it, has resulted in many algorithms that update control rules on an observation by observation basis, such as the Q-learning (Watkins and Dayan 1992). This may not necessarily be efficient for planning problems (Lagoudakis and Parr 2003). It is not until recently where more effort has been put to improve the efficiency of such methods that make them more appealing for planning problems (van Otterlo and Wiering 2012).

Rather than updating the policy on an observation by observation basis, the recent methods update the policy using batches of observations. The Least Square Policy Iteration (LSPI) (Lagoudakis and Parr 2003) algorithm is an example of that. The LSPI assumes that the value function is approximated by a linear in parameters function, and it estimates the parameters of the approximate function using a batch of data, or observations generated by a random policy using a simulator. Ernst et al. (2005) proposes an alternative algorithm to estimate value function approximated by a broader class of approximate functions, such as tree-based functions, in what he names as Fitted Q-iteration (FQI) algorithm. Riedmiller (2005) extends that to a neural network value function approximation, in what is know as Neural-Fitted-Q (NFQ). More recently, Mnih et al. (2013) proposed a similar algorithm to NFQ but argued to be more efficient, called Deep Q-Network (DQN). The DQN differs from the NFQ in two main aspects. First, it stores the simulated observations in memory, and randomly (uniformly) samples small batches of observations from memory for estimation of the parameters. Second, the value function is updated less frequently. They have applied their method using deep convolution neural networks to several Atari games, where the state variables are the raw screen pixels of the games. Their algorithm was able to outperform a professional player in about 29 games out of 49.

Among the mentioned algorithms, the Least Square Policy Iteration and FQI are the only ones that are purely batch-based and generally do not require setting any hyperparameters. It should be noted that both methods can be adapted to work on small batches of observations. Such a situation may arise in large dimensional problems where working on large batches is not feasible. In such problems, one may use smaller batches of observations. Neural Fitted Q requires setting some hyper-parameters such as learning rate and batch size for the neural network estimation step. The DQN is not a pure batch method and requires setting several hyper-parameters. Our analysis in this research will focus on pure batch methods. We are, more specifically, interested in the LSPI method with linear in parameter approximation of the value function.

4.4 Q-Learning

Q-learning, as introduced in Watkins (1989), and Watkins and Dayan (1992), is considered an incremental dynamic programming. Unlike dynamic programming, Q-learning does not require knowledge of the system model; that is, it is a model-free method. Q-learning is a direct method; that is, it learns the optimal policy without the need to learn the underlying model. Compared to indirect methods, which obtain an optimal policy through the learning of the system model, such as the method by Sato et al. (1988), Q-learning, generally, tends to be more efficient (Barto and Singh 1990). The Q-learning method is characterized by what is known as the Q-value (Watkins and Dayan 1992). The Q-value can be expressed as

$$Q^{\pi}(S,A) = R(S,A) + \beta E[V^{\pi}(S^{+})|S,A], \tag{4.1}$$

where S is a vector of state variables for the current period, S^+ is a vector of state variables for the next period, A is a vector of action variables, $R(S,A)$ is some reward function, π is some policy rule, and β is a discount factor. Additionally, and given some Q-values for some policy π, the state value function ($V(S)$) can be expressed as

$$V^{\pi} = \max_{A} Q^{\pi}(S,A). \tag{4.2}$$

The optimal Q-values can be estimated incrementally (Watkins 1989) from a sequence of (S_i, A, R, S_{i+1}) observations by the equation

$$Q_{i+1}(S_i, A_i) = (1-\alpha_i)Q_i(S_i, A_i) + \alpha_i[R(S_i, A_i) + \max_{A} \beta Q_i(S_{i+1}, A)], \tag{4.3}$$

where i represents the time step, α_i is the learning rate, S_i is the current state realization, S_{i+1} is the future state, and A is the action to be taken. A large learning rate implies that the updated value will put more weight on the predicted maximized value and less on the previous value. The algorithm in its simplest form is shown in algorithm 1.

Algorithm 1 Primitive Q-Learning

```
Initialize $S_0$ and $\alpha$
$i \leftarrow 0$
repeat
    Choose a random action $A_i$
    Calculate $R(S_i, A_i)$
    Simulate $S_{i+1}$ given $S_i$ and $A_i$
    Calculate $Q_i(S_i, A_i)$
    $i \leftarrow i + 1$
until Certain criteria is satisfied
```

4.5 Illustration of Q-learning on a Simple Pest Infestation Problem

We illustrate how Q-learning can be used to solve a dynamic problem with a simple pest infestation problem. The problem is based on a demonstration in the MDPSOLVE (Fackler 2011) package for Matlab. The problem is as follows: there is one state variable, S, that takes three values representing the infestation level. The levels are 1: low infestation, 2: medium infestation, 3: high infestation. There is one action variable, A. The action variable takes two values: 0 if no treatment is taken, and 1 if treatment is to be taken. The transition matrix for no action case is

$$P_0 = \begin{bmatrix} 0.65 & 0.15 & 0.05 \\ 0.25 & 0.4 & 0.2 \\ 0.1 & 0.45 & 0.75 \end{bmatrix}, \tag{4.4}$$

and for the action case is

$$P_1 = \begin{bmatrix} 0.85 & 0.45 & 0.35 \\ 0.15 & 0.5 & 0.5 \\ 0 & 0.05 & 0.15 \end{bmatrix}. \tag{4.5}$$

The reward function is

$$R(S, A) = -D(S) - C \cdot A, \tag{4.6}$$

where D=0, 5 and 20 for S =1,2, and 3 respectively, and $C = 10$. The problem is to find the best policy, let it be $\pi^*(S)$, that maximizes the expected discounted rewards, that is

$$\max_{\pi(S)} \mathbb{E} \sum_{i=0}^{\infty} \beta^i R(S_i, \pi(S_i)). \tag{4.7}$$

We have solved the problem using MDPSOLVE (Fackler 2011) package in Matlab, and the optimal policy is shown in Table 4.1. We have also solved the problem using the $\epsilon-greedy$ Q-learning algorithm with $S_o = 1$, $\epsilon = 0.5$. The $\epsilon-greedy$ policy assumes that action will be taken according to the Q-function with a probability ϵ and randomly (uniformly) with probability $1-\epsilon$. Figure 4.1 shows the Q-values at state $S = 1$ for different values of learning rates along with the optimal value.

There are two things to note. First, a large learning rate results in a fast learning but high variance in the Q-values, while a low learning rate results in slow learning but a low variance in the Q-values. Second, learning rate plays a critical role in the convergence of the Q-function to the optimal one. Watkins and Dayan (1992) have proved that given certain conditions for the learning rate, the algorithm converges to the optimal Q-function. The conditions require that the learning rate goes to zero as the number of steps (i) goes to infinity.

Table 4.1: Optimal policy.

A	S
0	1
1	2
1	3

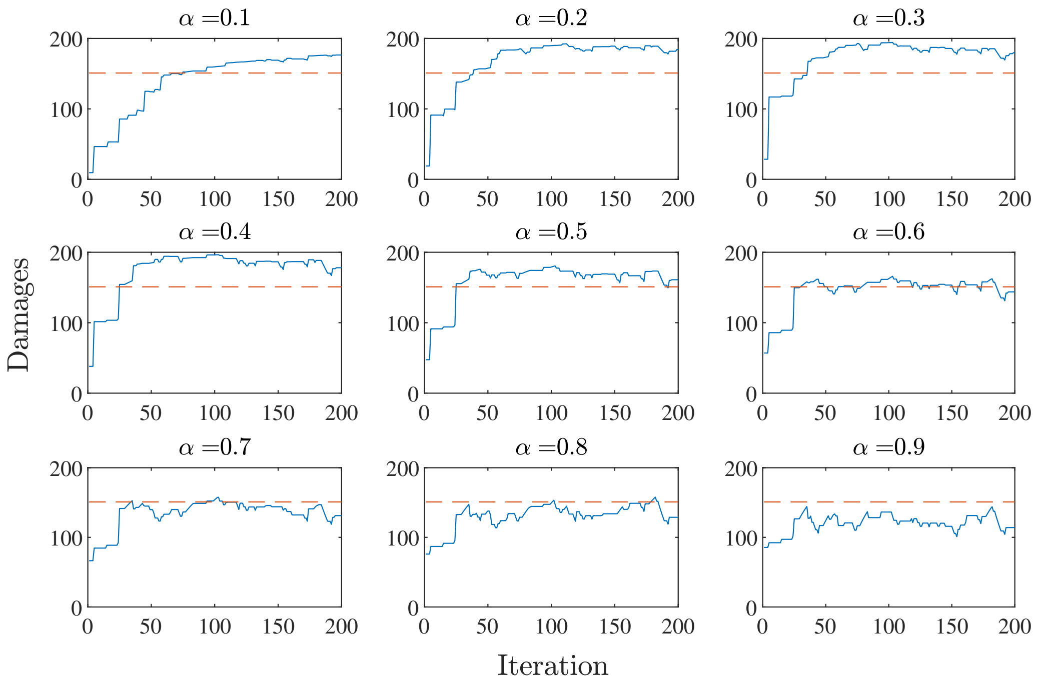


Figure 4.1: Optimal Q-values at S=1 for different learning rates. The red dashed line represents the optimal value at S=1, calculated using dynamic programming.

4.6 Least Square Policy Iteration

Suppose the Q-function is approximated with

$$Q(S,A|\theta) = \sum_k \phi_k(S,A) \cdot \theta_k, \tag{4.8}$$

where ϕ_k are some basis function, and θ_i are parameters to be estimated. Suppose that we have a set D of tuples (S, A, R, S') where S is the current state, A is the action, R is the reward and R' is the future state. The Least-Square Policy Iteration (LSPI) (Lagoudakis and Parr 2003) algorithm consists of two alternating steps. The first updates the Q-function, and that is through updating the vector of parameters θ. The parameters update equation is

$$\theta = \left[\sum_j \phi(S_j, A_j)(\phi(S_j, A_j) - \beta \cdot \phi(S'_j, \pi(S'_j))^T \right]^{-1} \sum_j \left[\phi(S_j, A_j) R_j \right] \tag{4.9}$$

where β is the discount factor and ϕ is a column vector of the basis functions. The second is the policy updating step, which depends on the updated Q-function. The policy updating step is

$$\pi(S) = \underset{A}{\operatorname{argmax}}\, Q(S,A|\theta). \tag{4.10}$$

The LSPI algorithm alternates between the policy evaluation step (4.9), and the policy improvement step (4.10), until there is no change in the policy. There are two advantages of LSPI. First, it does not require setting a hyperparameter, such as the learning rate. Second, it is has been proved to be stable (Lagoudakis and Parr 2003); that is, it either converges to some Q-values or fluctuates within a bound of Q-values (i.e., it does not diverge).

4.7 Fitted Q Iteration

Like LSPI, Fitted Q Iteration (FQI) (Ernst et al. 2005) is a pure batch RL algorithm that estimates an approximate Q-function. Unlike the LSPI, the FQI can fit a broader class of approximating functions, such as kernel-based, and tree-based methods. The FQI algorithm, as shown in algorithm 2, is analog to the value function iteration algorithm in dynamic programming.

Algorithm 2 Fitted Q Iteration

```
Given a set of tuples $(S, A, R, S^+)$,
Initialize the approximated Q-function (let be $\hat{q}$) to zero
repeat
    for j=1 to n do
        $\tilde{q}_j \leftarrow R(S_j, A_j) + \beta \max_A \hat{q}(S_j^+, A)$
    end for
    fit $\hat{q}$ using $\tilde{q}$, S,and A by some regression/fitting method
until Certain criteria is satisfied
```

4.8 llustration of LSPI and FQI on a Simple Pest Infestation Problem

We re-solve the previously introduced simple pest infestation problem using both LSPI and FQI algorithms. We estimate the Q-value with a parameter for each state-action combination, that is the Q-function to be estimated is

$$\hat{Q}(S,A) = \sum_{i=1}^{3}\sum_{j=0}^{1} \theta_{ij} \cdot \mathbb{1}_{S=i,A=j}. \tag{4.11}$$

The Q-values at S=1 for each iteration is shown in Fig. 4.2. The LSPI converges in the first few iterations while the FQI converges after about a hundred iterations. We have repeated the estimation process a hundred times for sample sizes of 100, 1000, and 10000. Both FQI and LSPI have been able to find the optimal policy 51, 90, and 100 percent of times for the three different sample sizes, respectively. It should be noted that in order to get an optimal or near-optimal policy, we need a sample size of more than a thousand observations.

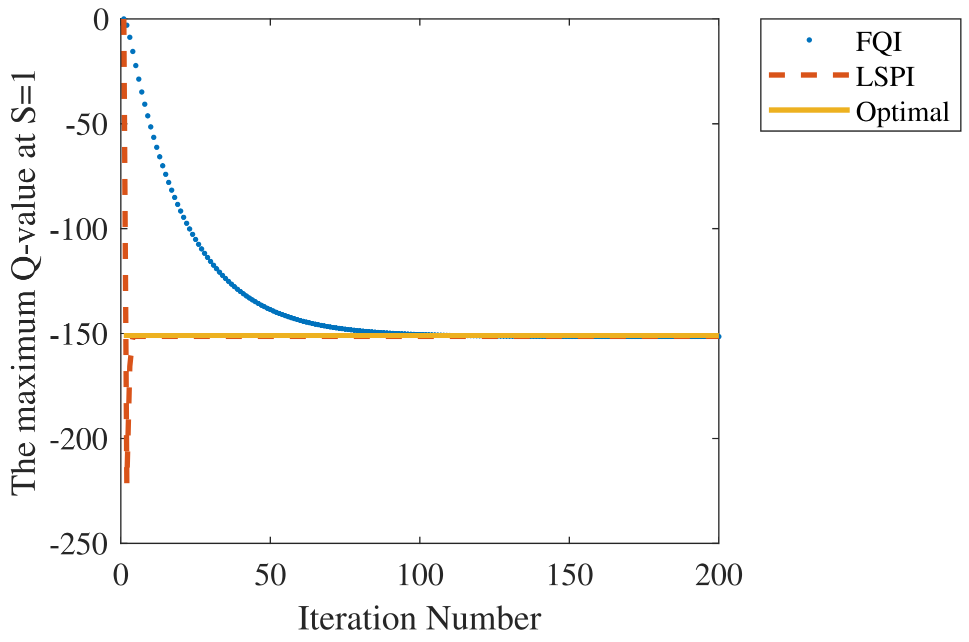


Figure 4.2: Q-values for S=1 for each iteration using the optimal action according to the Q-function using ten thousand simulated observations.

4.9 Analysis

In this section, we aim to explore and analyze linear in parameters approximate Q-functions and estimate them using the Least Square Policy Iteration. In the first part, we identify a good parametric approximation of the Q-function. In the second part, we estimate our approximate function using the LSPI algorithm for 4x4, 6x6, and 8x8 grid size problems of the invasive species model introduced in the previous chapter.

4.9.1 Approximate Q-function Specification

Deciding on the functional form of the Q-function is an essential step in the value function (or Q function) approximation methods. This step relies on both trial and error and, more importantly, the understanding of the problem structure. In this part, we plan to solve a 3x3 grid size problems and compute the optimal Q-function. We use the Q-optimal values to find a good functional approximation of the Q-function. We do so by exploring several features and fitting the Q-values using the Ordinary-Least-Square Method. We find that a good specification of the Q-function is:

$$Q = \sum_{i=1}^{N} (\alpha_{1i} \cdot S_i \cdot + \alpha_{2i} \cdot S_i \cdot q_i) \cdot A_i + \alpha_3 + \alpha_4 \cdot QNNeighInf \\ + \alpha_5 \cdot TotInf + \alpha_6 \cdot TotInf \cdot QNNeighInf, \quad (4.12)$$

where S_i is the state of site i, q_i is the number of infected neighbors of site i, A_i is an action variable that equals 1 if site i is to be treated, $QNNeighInf$ is the total number of infected neighbors for the uninfected sites, $TotInf$ is the total number of infected sites, and N is the total number of sites. We restrict α_{1i} and α_{2i} by the eigenvector centrality of the sites, as described in the previous chapter. That is, sites that have the same eigenvector centrality will have the same coefficients. Our approximate Q-function consists of two parts. The first describes the variation in action for a given state. The second describes the variation in the value function over different states. The rule predicts the optimal policy for 85% of state space.

Our investigation of the states where the model in Eq. (4.12) fails to predict the optimal actions show that the model will predict the second-best (see Table 4.2). Each row in Table 4.2 shows the value function value for a specific state. The value at the optimal action is dark grey shaded, and the action resulting from the approximate Q function is light gray shaded. The difference between the best and second-best Q-values in the states where the model fails to predict the optimal action is very small. This is summarized in the box and whisker plot in Fig. 4.3.

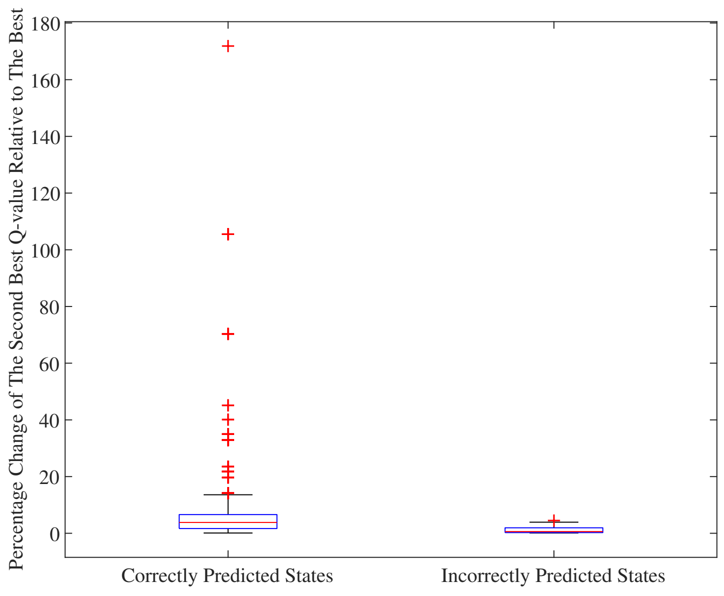


Figure 4.3: Box and whiskers plot of the percentage change of the second best Q-values relative to the best for both the states where the model predicts the optimal actions (correctly predicted) and the states where it predicts sub-optimal actions (incorrectly predicted). The whiskers length are 1.5 times the interquartile, which is the difference between the 75th and 25th percentiles values. The red pluses represent outliers. It should be noted that the percentage change of the second best Q-values relative to the best for the correctly predicted states are quite wide compared to the incorrectly predicted states.

Table 4.2: The Q-values for both the optimal action and predicted action using Equation (4.12). The parameters for the model are estimated using OLS with the optimal Q-values. The Q-values are for the states where the model fails to predict the optimal action. The dark gray shaded values represent the optimal ones while the light gray shaded ones represent the ones predicted by the model.

State									Action										
S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	1	2	3	4	5	6	7	8	9	Q-func. Action	Opt. Action
0	0	0	0	1	1	0	1	0	-22.5484	-21.2363	-21.5369	-21.2363	-17.0329	-17.6898	-21.5369	-17.6898	-21.4029	8	5
0	0	0	1	1	0	0	1	0	-21.5369	-21.2363	-22.5484	-17.6898	-17.0329	-21.2363	-21.4029	-17.6898	-21.5369	8	5
0	0	0	1	1	1	0	1	0	-30.7112	-30.5574	-30.7112	-26.6099	-27.9871	-26.6099	-30.4624	-27.4402	-30.4624	8	4
0	0	1	0	0	0	1	0	1	-22.4495	-21.1547	-17.0315	-21.1547	-22.4495	-20.8018	-17.0315	-20.8018	-17.7977	9	3
0	0	1	1	0	0	1	0	1	-31.1013	-30.9961	-26.3015	-26.4707	-31.0462	-30.5238	-28.3594	-30.6942	-27.1916	9	3
0	0	1	1	0	1	1	1	0	-37.8419	-37.7949	-34.2743	-34.0682	-36.7155	-34.4662	-36.3997	-34.7810	-37.6109	8	4
0	0	1	1	1	0	0	0	1	-32.8219	-32.1815	-29.3375	-27.8691	-28.5303	-32.3423	-32.8219	-32.1815	-29.3375	5	4
0	0	1	1	1	0	0	1	1	-38.4038	-37.7263	-34.7774	-34.2088	-35.5030	-37.9070	-37.9677	-36.2958	-36.6089	3	4
0	0	1	1	1	0	1	0	1	-39.8095	-39.0311	-35.7699	-36.8868	-35.9196	-38.9511	-37.4487	-39.2202	-36.6680	5	3
0	0	1	1	1	1	0	1	0	-37.7864	-37.1226	-35.2135	-33.5185	-35.6917	-35.8096	-37.1099	-34.1675	-37.5213	8	4
0	0	1	1	1	1	1	0	1	-43.2390	-42.5886	-40.5347	-40.2731	-40.3508	-42.3357	-40.7104	-42.6989	-41.3808	5	4
0	0	1	1	1	1	1	1	1	-46.2976	-45.6132	-43.5919	-43.5433	-44.3085	-45.4212	-44.9131	-45.7001	-45.5061	3	4
0	1	0	0	1	0	1	0	1	-32.8219	-27.8691	-32.8219	-32.1815	-28.5303	-32.1815	-29.3375	-32.3423	-29.3375	5	2
0	1	0	0	1	1	0	0	0	-21.5369	-17.6898	-21.4029	-21.2363	-17.0329	-17.6898	-22.5484	-21.2363	-21.5369	2	5
0	1	0	0	1	1	1	0	1	-38.4038	-34.2088	-37.9677	-37.7263	-35.5030	-36.2958	-34.7774	-37.9070	-36.6089	7	2
0	1	0	1	0	0	0	1	0	-24.9432	-20.0779	-25.1084	-20.8631	-23.7743	-26.2538	-24.9432	-20.0779	-25.1084	4	2
0	1	0	1	0	0	0	1	1	-32.3190	-27.5439	-32.9471	-28.2449	-31.7873	-32.9808	-32.7654	-29.9359	-29.9264	4	2
0	1	0	1	0	1	1	0	0	-32.7654	-28.2449	-32.3190	-29.9359	-31.7873	-27.5439	-29.9264	-32.9808	-32.9471	2	6
0	1	0	1	1	0	0	0	0	-21.4029	-17.6898	-21.5369	-17.6898	-17.0329	-21.2363	-21.5369	-21.2363	-22.5484	2	5
0	1	0	1	1	0	1	0	1	-37.9677	-34.2088	-38.4038	-36.2958	-35.5030	-37.7263	-36.6089	-37.9070	-34.7774	9	2
0	1	0	1	1	1	0	0	0	-30.4624	-27.4402	-30.4624	-26.6099	-27.9871	-26.6099	-30.7112	-30.5574	-30.7112	2	6
0	1	0	1	1	1	0	0	1	-37.1099	-34.1675	-37.5213	-33.5185	-35.6917	-35.8096	-37.7864	-37.1226	-35.2135	2	4
0	1	0	1	1	1	1	0	0	-37.5213	-34.1675	-37.1099	-35.8096	-35.6917	-33.5185	-35.2135	-37.1226	-37.7864	2	6
0	1	1	0	0	0	1	0	1	-31.1013	-26.4707	-28.3594	-30.9961	-31.0462	-30.6942	-26.3015	-30.5238	-27.1916	9	7
0	1	1	0	1	0	1	0	1	-39.8095	-36.8868	-37.4487	-39.0311	-35.9196	-39.2202	-35.7699	-38.9511	-36.6680	5	7
0	1	1	0	1	0	1	1	1	-43.2390	-40.2731	-40.7104	-42.5886	-40.3508	-42.6989	-40.5347	-42.3357	-41.3808	5	2
0	1	1	0	1	1	1	1	1	-46.2976	-43.5433	-44.9131	-45.6132	-44.3085	-45.7001	-43.5919	-45.4212	-45.5061	7	2
0	1	1	1	0	0	0	1	0	-32.7654	-29.9359	-29.9264	-28.2449	-31.7873	-32.9808	-32.3190	-27.5439	-32.9471	4	8
0	1	1	1	0	0	1	1	0	-37.6109	-34.4662	-34.2743	-34.7810	-36.7155	-37.7949	-36.3997	-34.0682	-37.8419	2	8
0	1	1	1	0	1	1	0	0	-37.6109	-34.7810	-36.3997	-34.4662	-36.7155	-34.0682	-34.2743	-37.7949	-37.8419	2	6
0	1	1	1	0	1	1	0	1	-44.1297	-41.4354	-43.3227	-41.2187	-43.7152	-42.8925	-41.7155	-44.2196	-42.2496	2	4
0	1	1	1	1	0	0	0	1	-37.9677	-36.2958	-36.6089	-34.2088	-35.5030	-37.9070	-38.4038	-37.7263	-34.7774	9	4
0	1	1	1	1	0	1	1	1	-48.1853	-46.3398	-46.1838	-46.5702	-47.1813	-47.9868	-47.3788	-47.8582	-46.6760	9	3
0	1	1	1	1	1	1	0	1	-48.1853	-46.5702	-47.3788	-46.3398	-47.1813	-47.8582	-46.1838	-47.9868	-46.6760	9	7
1	0	0	0	0	1	1	0	1	-26.3015	-30.9961	-31.1013	-30.5238	-31.0462	-26.4707	-27.1916	-30.6942	-28.3594	7	1
1	0	0	0	1	1	1	0	0	-29.3375	-32.1815	-32.8219	-32.3423	-28.5303	-27.8691	-29.3375	-32.1815	-32.8219	5	6
1	0	0	0	1	1	1	0	1	-35.7699	-39.0311	-39.8095	-38.9511	-35.9196	-36.8868	-36.6680	-39.2202	-37.4487	5	1
1	0	0	0	1	1	1	1	0	-34.7774	-37.7263	-38.4038	-37.9070	-35.5030	-34.2088	-36.6089	-36.2958	-37.9677	1	6
1	0	0	1	0	1	0	1	0	-29.9264	-32.9808	-32.9471	-29.9359	-31.7873	-27.5439	-32.7654	-28.2449	-32.3190	8	6
1	0	0	1	0	1	0	1	1	-34.2743	-37.7949	-37.8419	-34.4662	-36.7155	-34.0682	-37.6109	-34.7810	-36.3997	8	6
1	0	0	1	1	1	0	1	0	-35.2135	-37.1226	-37.7864	-35.8096	-35.6917	-33.5185	-37.5213	-34.1675	-37.1099	8	6
1	0	0	1	1	1	1	0	1	-40.5347	-42.5886	-43.2390	-42.3357	-40.3508	-40.2731	-41.3808	-42.6989	-40.7104	5	6
1	0	0	1	1	1	1	1	1	-43.5919	-45.6132	-46.2976	-45.4212	-44.3085	-43.5433	-45.5061	-45.7001	-44.9131	1	6
1	0	1	0	0	0	0	1	1	-26.3015	-30.5238	-27.1916	-30.9961	-31.0462	-30.6942	-31.1013	-26.4707	-28.3594	3	1
1	0	1	0	1	0	0	1	0	-29.3375	-32.3423	-29.3375	-32.1815	-28.5303	-32.1815	-32.8219	-27.8691	-32.8219	5	8
1	0	1	0	1	0	0	1	1	-35.7699	-38.9511	-36.6680	-39.0311	-35.9196	-39.2202	-39.8095	-36.8868	-37.4487	5	1
1	0	1	0	1	0	1	1	0	-36.6680	-38.9511	-35.7699	-39.2202	-35.9196	-39.0311	-37.4487	-36.8868	-39.8095	5	3
1	0	1	0	1	0	1	1	1	-42.8107	-45.2599	-42.8107	-45.5085	-42.9120	-45.5085	-44.0826	-45.3378	-44.0826	5	3
1	0	1	0	1	1	0	1	0	-34.7774	-37.9070	-36.6089	-37.7263	-35.5030	-36.2958	-38.4038	-34.2088	-37.9677	1	8
1	0	1	0	1	1	1	0	0	-36.6680	-39.2202	-37.4487	-38.9511	-35.9196	-36.8868	-35.7699	-39.0311	-39.8095	5	7
1	0	1	0	1	1	1	0	1	-42.8107	-45.5085	-44.0826	-45.2599	-42.9120	-45.3378	-42.8107	-45.5085	-44.0826	5	7
1	0	1	1	0	1	0	1	1	-41.7155	-44.2196	-42.2496	-41.2187	-43.7152	-42.8925	-44.1297	-41.4354	-43.3227	8	4
1	0	1	1	0	1	1	1	0	-42.2496	-44.2196	-41.7155	-42.8925	-43.7152	-41.2187	-43.3227	-41.4354	-44.1297	8	6
1	0	1	1	1	0	0	0	1	-37.4487	-39.2202	-36.6680	-36.8868	-35.9196	-38.9511	-39.8095	-39.0311	-35.7699	5	9
1	0	1	1	1	0	0	1	0	-36.6089	-37.9070	-34.7774	-36.2958	-35.5030	-37.7263	-37.9677	-34.2088	-38.4038	3	8
1	0	1	1	1	0	1	0	1	-44.0826	-45.5085	-42.8107	-45.3378	-42.9120	-45.2599	-44.0826	-45.5085	-42.8107	5	3
1	0	1	1	1	1	0	0	1	-40.7104	-42.6989	-41.3808	-40.2731	-40.3508	-42.3357	-43.2390	-42.5886	-40.5347	5	4
1	0	1	1	1	1	0	1	1	-46.1838	-47.9868	-46.6760	-46.3398	-47.1813	-47.8582	-48.1853	-46.5702	-47.3788	3	1
1	0	1	1	1	1	1	0	0	-41.3808	-42.6989	-40.7104	-42.3357	-40.3508	-40.2731	-40.5347	-42.5886	-43.2390	5	6
1	1	0	0	0	1	0	1	1	-34.2743	-34.4662	-37.6109	-37.7949	-36.7155	-34.7810	-37.8419	-34.0682	-36.3997	2	8
1	1	0	0	1	0	1	0	1	-37.4487	-36.8868	-39.8095	-39.2202	-35.9196	-39.0311	-36.6680	-38.9511	-35.7699	5	9
1	1	0	0	1	0	1	1	1	-40.7104	-40.2731	-43.2390	-42.6989	-40.3508	-42.5886	-41.3808	-42.3357	-40.5347	5	2
1	1	0	0	1	1	1	0	0	-36.6089	-36.2958	-37.9677	-37.9070	-35.5030	-34.2088	-34.7774	-37.7263	-38.4038	7	6
1	1	0	0	1	1	1	1	1	-46.1838	-46.3398	-48.1853	-47.9868	-47.1813	-46.5702	-46.6760	-47.8582	-47.3788	7	1
1	1	0	1	0	1	0	0	1	-36.3997	-34.7810	-37.6109	-34.0682	-36.7155	-34.4662	-37.8419	-37.7949	-34.2743	2	4
1	1	0	1	0	1	1	0	1	-43.3227	-41.4354	-44.1297	-42.8925	-43.7152	-41.2187	-42.2496	-44.2196	-41.7155	2	6
1	1	0	1	1	0	1	1	1	-44.9131	-43.5433	-46.2976	-45.7001	-44.3085	-45.6132	-45.5061	-45.4212	-43.5919	9	2
1	1	1	0	1	0	0	1	1	-40.5347	-42.3357	-41.3808	-42.5886	-40.3508	-42.6989	-43.2390	-40.2731	-40.7104	5	8
1	1	1	0	1	0	1	0	1	-44.0826	-45.3378	-44.0826	-45.5085	-42.9120	-45.5085	-42.8107	-45.2599	-42.8107	5	7
1	1	1	0	1	0	1	1	0	-41.3808	-42.3357	-40.5347	-42.6989	-40.3508	-42.5886	-40.7104	-40.2731	-43.2390	5	8
1	1	1	0	1	1	0	1	1	-43.5919	-45.4212	-45.5061	-45.6132	-44.3085	-45.7001	-46.2976	-43.5433	-44.9131	1	8
1	1	1	1	1	0	1	1	0	-45.5061	-45.4212	-43.5919	-45.7001	-44.3085	-45.6132	-44.9131	-43.5433	-46.2976	3	8
1	1	1	1	1	1	0	0	1	-44.9131	-45.7001	-45.5061	-43.5433	-44.3085	-45.4212	-46.2976	-45.6132	-43.5919	9	4
1	1	1	1	1	1	1	0	0	-45.5061	-45.7001	-44.9131	-45.4212	-44.3085	-43.5433	-43.5919	-45.6132	-46.2976	7	6

4.10 Analysis Results

We estimate the approximate Q-function in Eq. (4.12) by the LSPI algorithm using a sample size of one million observations. The sample is generated by simulating forward for hundred-time steps using a random policy. We do this for ten thousand randomly (uniform) generated starting points. Simulation results for different grid size cases and different action effectiveness are shown in Figs (4.4) and (4.5).

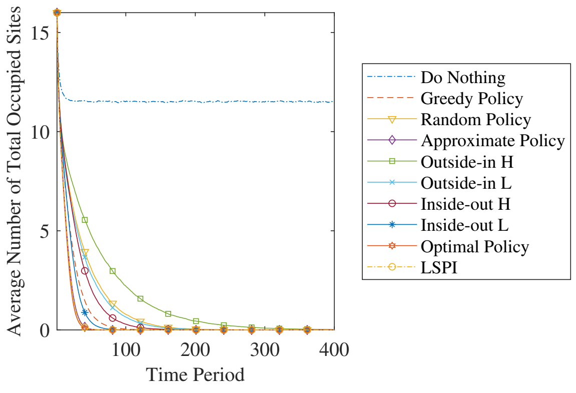
The LSPI policy does well compared to the optimal policy in the 4x4 grid case. Its performance is better than any other policies in the situation where the action is partially effective ($p_t = 0.7$), and does as well as the optimal and approximate policies from the previous chapter. The performance gaps between the different policies shrink as the action becomes more effective (i.e., as we increase p_t from 0.7 to 1).

The performance gaps between policies increase as the grid size increases. Eradication starts to become harder to achieve as the grid size increases. This is more apparent when one compares the 8x8 grid case for the situations when action is partially effective ($p_t = 0.7$) and fully effective ($p_t = 1$). While in the 4x4 grid size, eradication is achieved in both situations, it is not the case for the 8x8 grid size.

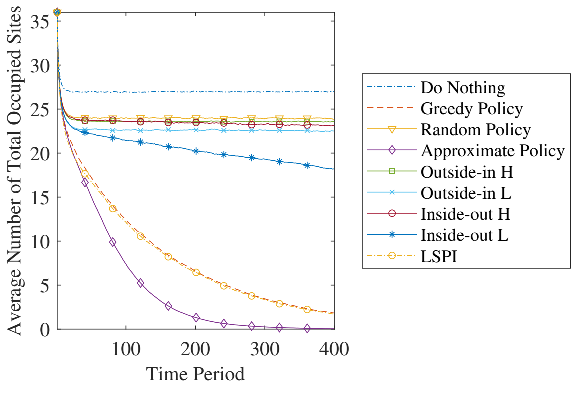
The LSPI policy does not perform as well as the approximate policy in the 6x6 grid size, where the action is partially effective ($p_t = 0.7$) and in the 8x8 grid size when action is fully effective ($p_t = 1$). While there could be many factors that result in this, two main factors are the sample size used in the estimation and the functional form of the estimated Q-function. We have replicated the estimation for sample sizes up to a hundred million observations for the 6x6 grid size case and have noticed no improvement in the policy performance. This suggests that the functional form is likely to be the issue. We have experimented with additional features, such as polynomial terms of the basic features and their interactions.

The additional features, however, did not result in an improvement.

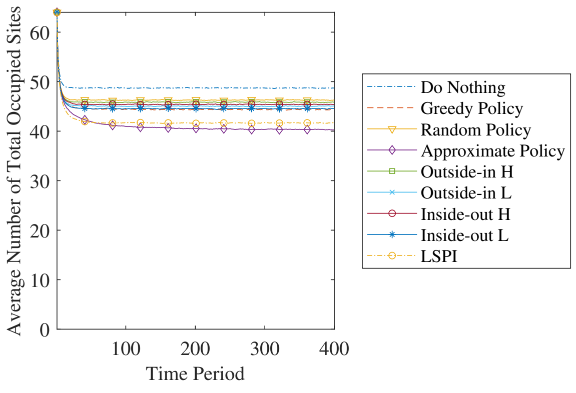
Nevertheless, and aside from the approximate policy and greedy policies, the LSPI policy appears to do better than other policies in all situations. Additionally, it has the potential to do better than even the approximate and greedy policies in some situations. This is illustrated in the sensitivity analysis in the appendix.



(a) 4x4 grid network.

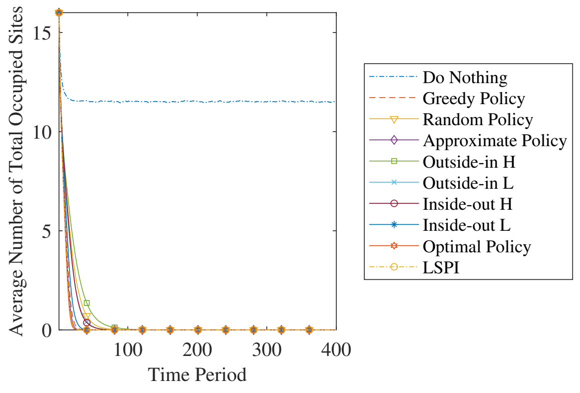


(b) 6x6 grid network.

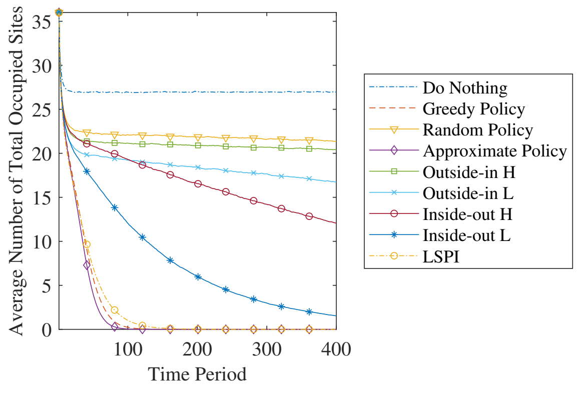


(c) 8x8 grid network.

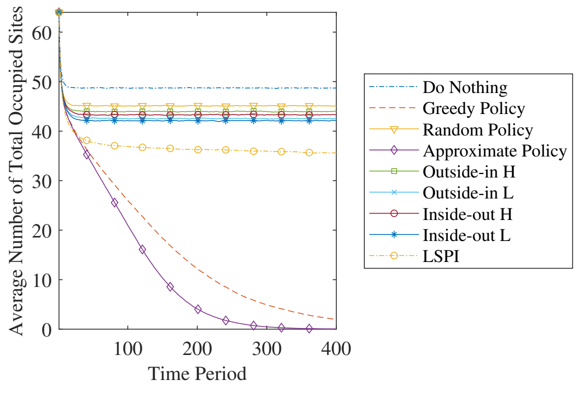
Figure 4.4: Time path simulations for a 4x4, 6x6 and 8x8 grid network where actions is assumed to be partially effective ($p_0 = 0, p_1 = 0.15, p_n = 0.1, p_t = 0.7$). The time paths represent an average over a ten thousand replications.



(a) 4x4 grid network.



(b) 6x6 grid network.



(c) 8x8 grid network.

Figure 4.5: Time path simulations for a 4x4, 6x6 and 8x8 grid network where actions is assumed to be completely effective ($p_0 = 0, p_1 = 0.15, p_n = 0.1, p_t = 1$). The time paths represent an average over a ten thousand replications.

4.10.1 Multi-layer Star Network

We analyze our approximate policy on Chadès et al. (2011) 101 nodes multi-layer star network with their baseline model parameters. Figure 3.15 shows the time path simulation results for the baseline model. Our LSPI policy outperforms our approximate policy from the previous chapter. It also outperforms all other comparison policies, including Chadès et al. (2011) rule of thumb. The sensitivity analysis results in the appendix show that the LSPI policy's performance is comparable to our approximate policy and consistently outperforms other comparison policies.

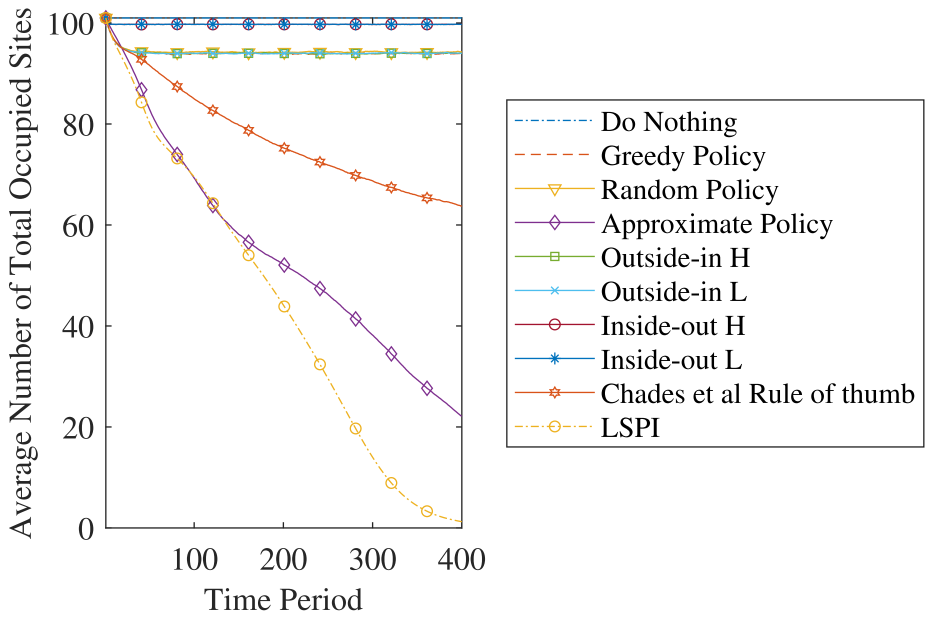


Figure 4.6: Simulations results on Chadès et al. (2011) 101 nodes multi-layer star network ($p_0 = 0, p_1 = 0.1, p_n = 0, p_t = 0.7$).

4.11 Conclusion

Our analysis shows that the LSPI algorithm results in a near-optimal policy rule for a small dimensional grid spatial problem that can be solved using dynamic programming. However, as the dimension of the problem increases, the LSPI method policy performance degrades. Nevertheless, the LSPI policy appears to do well in Chadès et al. (2011) 101 nodes multi-layer star network.

The choice of features is a main factor. Unlike problems with continuous state variables, the use of a large number of polynomials terms and basis functions as features is not applicable when the state variables are discrete. This makes features selection a more complicated process as one needs to construct features based on the understating of the problems. Things are even more complicated with spatial grid problems where spatial aspects can play critical parts in the features construction. This is especially true as the number of sites gets larger.

In higher-dimensional problems, our objective is to construct a set of features for the approximate Q-function that can generalize out of sample states. This is different from having features that fit the simulated observations well. For example, a functional form that fits the observed states well, may not necessarily generalize well over the unobserved states.

Despite that, our analysis shows that the LSPI algorithm can result in a good policy for a large-dimensional problem if one can determine a good set of features for the Q-function. Additionally, it can be handy when one does not have an explicit model that can be solved using dynamic programming. Such problems arise when a sophisticated black-box simulator represents the model of the problem. Both policy and value approximation methods deserve further exploration in realistic resource management settings.

REFERENCES

Albers, H. J., Ando, A., and Shogren, J. F. (2010). Introduction to spatial natural resource and environmental economics. *Resource and Energy Economics*, 32(2):93–97.

Albertini, F. and Sontag, E. (1993). Uniqueness of weights for neural networks.

Ando, A. W. and Baylis, K. (2014). *Spatial Environmental and Natural Resource Economics*, pages 1029–1048. Springer Berlin Heidelberg, Berlin, Heidelberg.

Arthur, W. B. (1999). Complexity and the economy. *Science*, 284(5411):107–109.

Barabási, A.-L. (2016). *Network science.* Cambridge University Press.

Barto, A. G. and Singh, S. P. (1990). On the Computational Economics of Reinforcement Learning. *Connectionist Models: Proceedings of the 1990 Summer School*, pages 35–44.

Belbute-Peres, F. D. A., Smith, K. A., Allen, K. R., Tenenbaum, J. B., and Kolter, J. Z. (2018). End-to-End Differentiable Physics for Learning and Control. Technical report, 32nd Conferenceon Neural Information Processing Systems.

Bellman, R. (1957). A Markovian Decision Process.

Bellman, R. E. (1954). The Theory of Dynamic Programming.

Bertsekas, D. P. (2019). *Reinforcement learning and optimal control.* Athena Scientific.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming.* Athena Scientific, 1st edition.

Bianchi, F. J., Schellhorn, N. A., Buckley, Y. M., and Possingham, H. P. (2010). Spatial variability in ecosystem services: Simple rules for predator-mediated pest suppression. *Ecological Applications*, 20(8):2322–2333.

Bonacich, P. (1972). Technique for Analyzing Overlapping Memberships. *Sociological Methodology*, 4(1972):176–185.

Brock, W. A. and Xepapadeas, A. (2011). Optimal Control and Spatial Heterogeneity: Pattern Formation in Economic-Ecological Models. pages 1–44.

Chadès, I., Martin, T. G., Nicol, S., Burgman, M. A., Possingham, H. P., and Buckley, Y. M. (2011). General rules for managing and surveying networks of pests, diseases, and endangered species. *Proceedings of the National Academy of Sciences of the United States of America*, 108(20):8323–8328.

Chalak, M., Ruijs, A., and Van Ierland, E. C. (2011). Biological control of invasive plant species: A stochastic analysis. *Weed Biology and Management*, 11(3):137–151.

Costello, C. and Polasky, S. (2004). Dynamic reserve site selection. *Resource and Energy Economics*, 26(2):157–174.

Court, A. T. (1939). Hedonic price indexes with automotive examples. *The dynamics of automobile demand*, pages 99–119.

DeAngelis, D. L. and Yurek, S. (2017). Spatially Explicit Modeling in Ecology: A Review. *Ecosystems*, 20(2):284–300.

Epanchin-Niell, R. S. and Hastings, A. (2010). Controlling established invaders: Integrating economics and spread dynamics to determine optimal management. *Ecology Letters*, 13(4):528–541.

Epanchin-Niell, R. S. and Wilen, J. E. (2012). Optimal spatial control of biological invasions. *Journal of Environmental Economics and Management*, 63(2):260–270.

Ernst, D., Glavic, M., Geurts, P., and Wehenkel, L. (2005). Approximate value iteration in the reinforcement learning context. application to electrical power system control. *International Journal of Emerging Electric Power Systems*, 3(1):1–35.

Fackler, P. L. (2011). Mdpsolve: Matlab tools for solving markov decision problems.

Fackler, P. L. (2012). Category count models for resource management. *Methods in Ecology and Evolution*, 3(3):555–563.

Ferguson, N. M., Cummings, D. A. T., Cauchemez, S., Fraser, C., Riley, S., Meeyai, A., Iamsirithaworn, S., and Burke, D. S. (2005). Strategies for containing an emerging influenza pandemic in SE Asia . Supplementary Information. *Nature*, 437(7056):1–25.

Hof, J. and Bevers, M. (2002). *Spatial Optimization in Ecological Applications*. Columbia University Press.

Hotelling, H. (1929). Stability in Competition. *The Economic Journal*, 39(153):41–57.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA.

Johnston, R. J., Grigalunas, T. A., Opaluch, J. J., Mazzotta, M., and Diamantedes, J. (2002). valuing Estuarine Resource Services-Robert J Johnston. pages 47–65.

Judd, K. (1998). *Numerical Methods in Economics*, volume 1. The MIT Press, 1 edition.

Judd, K. L., Maliar, L., and Maliar, S. (2011). Numerically stable and accurate stochastic simulation approaches for solving dynamic economic models. *Quantitative Economics*, 2(2):173–210.

Klaiber, A. and Phaneuf, D. J. (2010). Valuing open space in a residential sorting model of the Twin Cities. *Journal of Environmental Economics and Management*, 60(2):57–77.

Krugman, P. (1998). What's new about the new economic geography? *Oxford Review of Economic Policy*, 14(2):7–17.

Kuminoff, N. V., Smith, V. K., and Timmins, C. (2010). The new economics of equilibrium sorting and its transformational role for policy evaluation.

Laber, E. B., Meyer, N. J., Reich, B. J., Pacifici, K., Collazo, J. A., and Drake, J. M. (2018). Optimal treatment allocations in space and time for on-line control of an emerging infectious disease. *Journal of the Royal Statistical Society. Series C: Applied Statistics*, 67(4):743–789.

Lagoudakis, M. and Parr, R. (2003). Least-Squares Policy Iteration. *Journal of Machine Learning Researh*, pages 1107–1149.

Lancaster, K. J. (1966). A new approach to consumer theory. *Journal of political economy*, 74(2):132–157. Date revised - 2013-06-12; Last updated - 2013-09-16; SubjectsTermNotLitGenreText - 3372 3883 971; 2777 2803 3874 556 3889 6071 1542 11325.

Lang, J. C., De Sterck, H., Kaiser, J. L., and Miller, J. C. (2018). Analytic models for SIR disease spread on random spatial networks. *Journal of Complex Networks*, 6(6):948–970.

Liu, Y., Deng, Y., Jusup, M., and Wang, Z. (2016). A biologically inspired immunization strategy for network epidemiology. *Journal of Theoretical Biology*, 400:92–102.

M. Girvan and Newman, M. E. J. (2002). Community structure in social and biological networks. *PNAS*.

Marcel Salathe, J. H. J. (2010). Dynamics and Control of Diseases in Networks with Community Structure. *PloS Computational Biology*.

Marescot, L., Chapron, G., Chadès, I., Fackler, P. L., Duchamp, C., Marboutin, E., and Gimenez, O. (2013). Complex decisions made simple: A primer on stochastic dynamic programming. *Methods in Ecology and Evolution*, 4(9):872–884.

Meier, E. S., Dullinger, S., Zimmermann, N. E., Baumgartner, D., Gattringer, A., and Hülber, K. (2014). Space matters when defining effective management for invasive plants. *Diversity and Distributions*, 20(9):1029–1043.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *arXiv e-prints*, page arXiv:1312.5602.

Moxnes, E. (2003). Uncertain measurements of renewable resources: Approximations, harvesting policies and value of accuracy. *Journal of Environmental Economics and Management*, 45(1):85–108.

Murphy, S. A. (2003). Optimal dynamic treatment regimes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(2):331–355.

Newman, M. E. J. (2007). The mathematics of networks. *The New Palgrave Encyclopedia of Economics*, 2:1–12.

Newman, M. E. J. (2009). The spread of epidemic disease on networks. *Physical Review E*, 66(1).

Olson, L. J. and Roy, S. (2002). The economics of controlling a stochastic biological invasion. *American Journal of Agricultural Economics*, 84(5):1311–1316.

Palmquist, R. B. (2005). Chapter 16 Property Value Models. *Handbook of Environmental Economics*, 2(05):763–819.

Palmquist, R. B. and Smith, V. K. (2001). The Use of Hedonic Property Value Techniques for Policy and Litigation. *International Yearbook of Envrionmental and Resources Economics*, 6:78.

Perry, G. L., Moloney, K. A., and Etherington, T. R. (2017). Using network connectivity to prioritise sites for the control of invasive species. *Journal of Applied Ecology*, 54(4):1238–1250.

Powell, W. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley Series in Probability and Statistics. Wiley.

Powell, W. B. (2008). What You Should Know About Approximate Dynamic Programming. *Naval Research Logistics*, 55(Dec 2009):240–249.

Proulx, S. R., Promislow, D. E., and Phillips, P. C. (2005). Network thinking in ecology and evolution.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley Sons, Inc., USA, 1st edition.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Reed, W. J. (1979). Optimal escapement levels in stochastic and deterministic harvesting models. *Journal of Environmental Economics and Management*, 6(4):350–363.

Riedmiller, M. (2005). Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method BT - Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings. *Ecml 2005*, pages 317–328.

Sanchirico, J. and Wilen, J. (1999). Bioeconomics of Spatial Exploitation in A Patchy Environment. *Journal of Environmental Economics and Management*, (37):129–150.

Sato, M., Abe, K., and Takeda, H. (1988). Learning control of finite Markov chains with an explicit trade-off between estimation and control. *IEEE Transactions on Systems, Man and Cybernetics*, 18(5):677–684.

Schelling, T. C. (1971). Dynamics Model of Segregation. *Journal of Mathematical Sociology*, 1(May 1969):143–186.

Smith, A. A. (1990). *Three essays on the solution and estimation of dynamic macroeconomic models*. Duke University. Dissertation.

Springborn, M. R. and Faig, A. (2019). Moving Forward: A Simulation-Based Approach for Solving Dynamic Resource Management Problems. Technical report.

Sraffa, P. (1926). The Laws of Returns under Competitive Conditions. *The Economic Journal*, 36(144):535.

Strogatz, S. H. (2001). Exploring complex networks. *Nature*, 410(6825):268–276.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Taylor, L. O. (2003). *The Hedonic Method*, pages 331–393. Springer Netherlands, Dordrecht.

Thorndike, E. L. (1911). *Animal intelligence; experimental studies,*. New York,The Macmillan company,. https://www.biodiversitylibrary.org/bibliography/1201 — The study of consciousness and the study of behavior.–Animal intelligence.–The instinctive reactions of young chicks.–A note on the psychology of fishes.–The mental life of the monkeys.–Law and h.

van Otterlo, M. and Wiering, M. (2012). *Reinforcement Learning and Markov Decision Processes*, pages 3–42. Springer Berlin Heidelberg, Berlin, Heidelberg.

Watkins, C. J. and Dayan, P. (1992). Technical Note: Q-Learning. *Machine Learning*, 8(3):279–292.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK.

Waugh, F. V. (1928). Quality factors influencing vegetable prices. *Journal of Farm Economics*, 10(2):185–196.

APPENDIX

APPENDIX

A

SENSITIVITY ANALYSIS

The sensitivity analysis are conducted for three grid sizes: 4x4, 6x6, and 8x8 and 8x50. In each grid size we conduct a sensitivity for three parameters: action effectiveness (p_t), diffusion due to infected neighbors (p_1), and natural remission or clearance (p_n). Each panel shows a sensitivity analysis for the combination of p_n and p_1 values where each can take three possible values given a fixed value of action effectiveness (p_t). We follow the treatment of Chadès et al. (2011) for the sensitivity of p_t, which we do it for $p_t = 0.7$ and $p_t = 1$. Table A.1 lists the different sensitivity analysis cases.

Table A.1: List of sensitivity analysis cases.

Figure Number	Grid(or network) Dimension	Fixed Parameters Value	Description
A.1	4x4	$p_0 = 0, p_t = 0.7$	Mean of the occupied sites
A.2	4x4	$p_0 = 0, p_t = 0.7$	Std. Dev. of the occupied sites
A.3	4x4	$p_0 = 0, p_t = 1$	Mean of the occupied sites
A.4	4x4	$p_0 = 0, p_t = 1$	Std. Dev. of the occupied sites sites
A.5	6x6	$p_0 = 0, p_t = 0.7$	Mean of the occupied sites
A.6	6x6	$p_0 = 0, p_t = 0.7$	Std. Dev. of the occupied sites
A.7	6x6	$p_0 = 0, p_t = 1$	Mean of the occupied sites
A.8	6x6	$p_0 = 0, p_t = 1$	Std. Dev. of the occupied sites sites
A.9	8x8	$p_0 = 0, p_t = 0.7$	Mean of the occupied sites
A.10	8x8	$p_0 = 0, p_t = 0.7$	Std. Dev. of the occupied sites
A.11	8x8	$p_0 = 0, p_t = 1$	Mean of the occupied sites
A.12	8x8	$p_0 = 0, p_t = 1$	Std. Dev. of the occupied sites
A.13	8x50	$p_0 = 0, p_t = 0.7$	Mean of the occupied sites
A.14	8x50	$p_0 = 0, p_t = 0.7$	Std. Dev. of the occupied sites
A.15	8x50	$p_0 = 0, p_t = 1$	Mean of the occupied sites
A.16	8x50	$p_0 = 0, p_t = 1$	Std. Dev. of the occupied sites
A.17	Multi-layer Star (101 nodes)	$p_0 = 0, p_t = 0.7$	Mean of the occupied sites
A.18	Multi-layer Star (101 nodes)	$p_0 = 0, p_t = 0.7$	Std. Dev. of the occupied sites
A.19	Multi-layer Star (101 nodes)	$p_0 = 0, p_t = 1$	Mean of the occupied sites
A.20	Multi-layer Star (101 nodes)	$p_0 = 0, p_t = 1$	Std. Dev. of the occupied sites

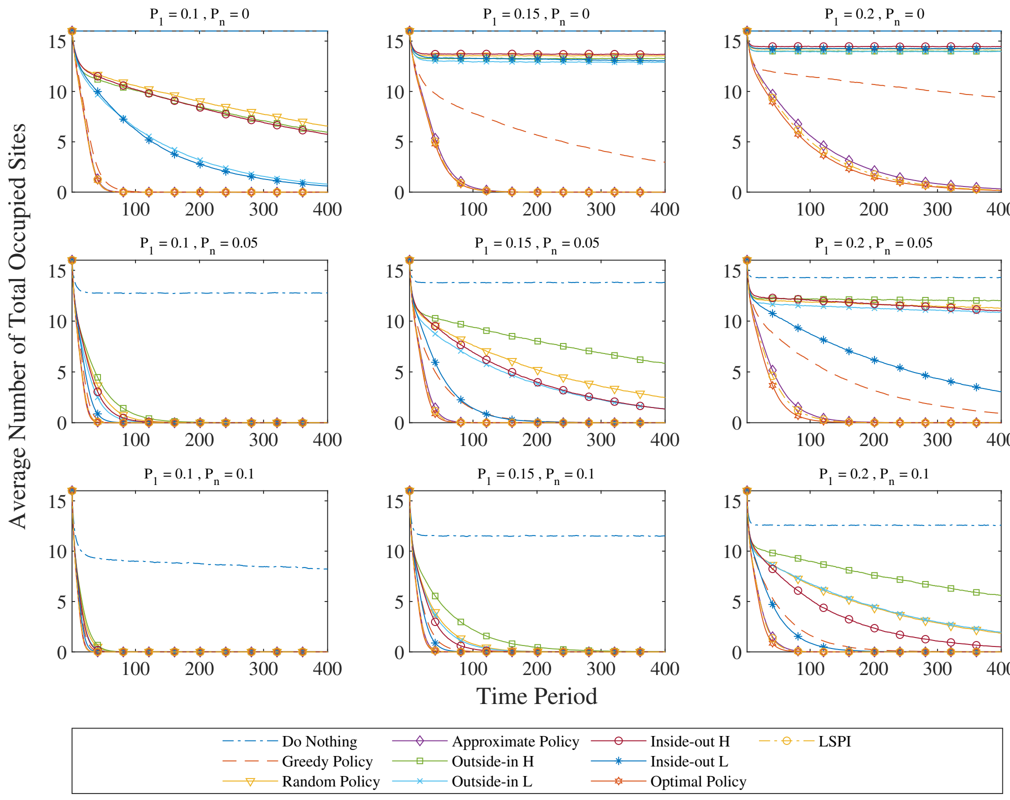


Figure A.1: Average number of occupied sites over ten thousands replications and four hundreds periods for different policies for a 4x4 grid situation. The spontaneous infection parameter is set to zero (p_0=0). Action is assumed to be partially effective ($p_t = 0.7$).

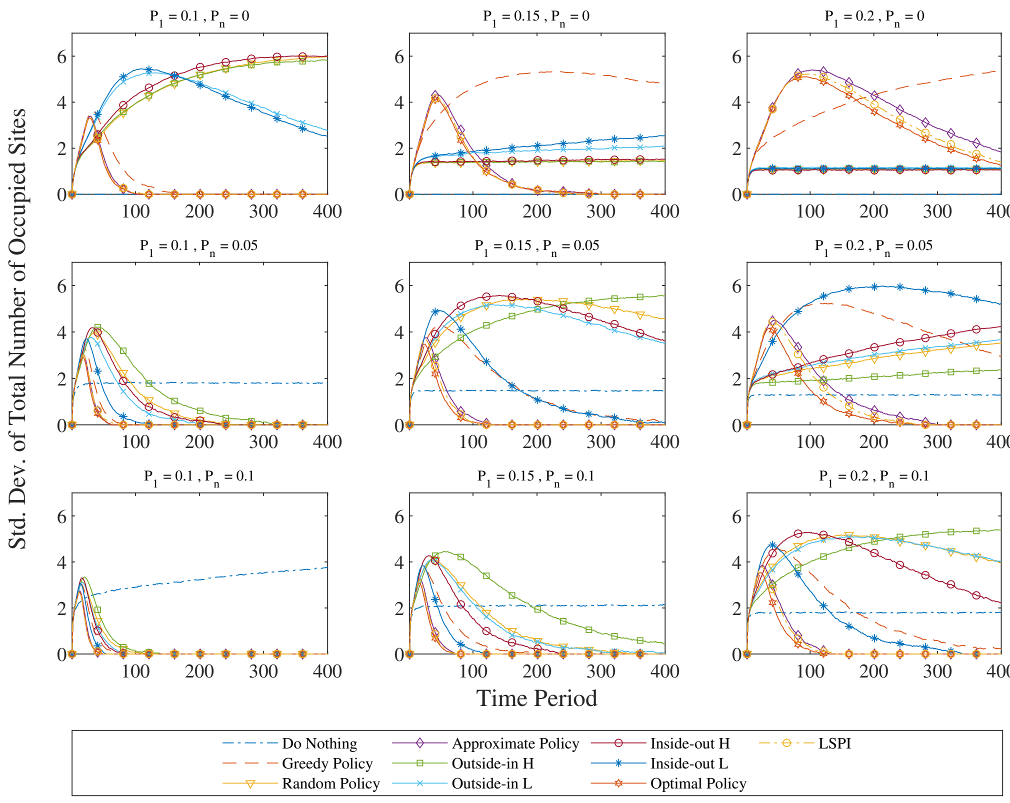


Figure A.2: Standard deviation of the number of occupied sites over ten thousand replications and four hundred periods for different policies for a 4x4 grid situation. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be partially effective ($p_t = 0.7$).

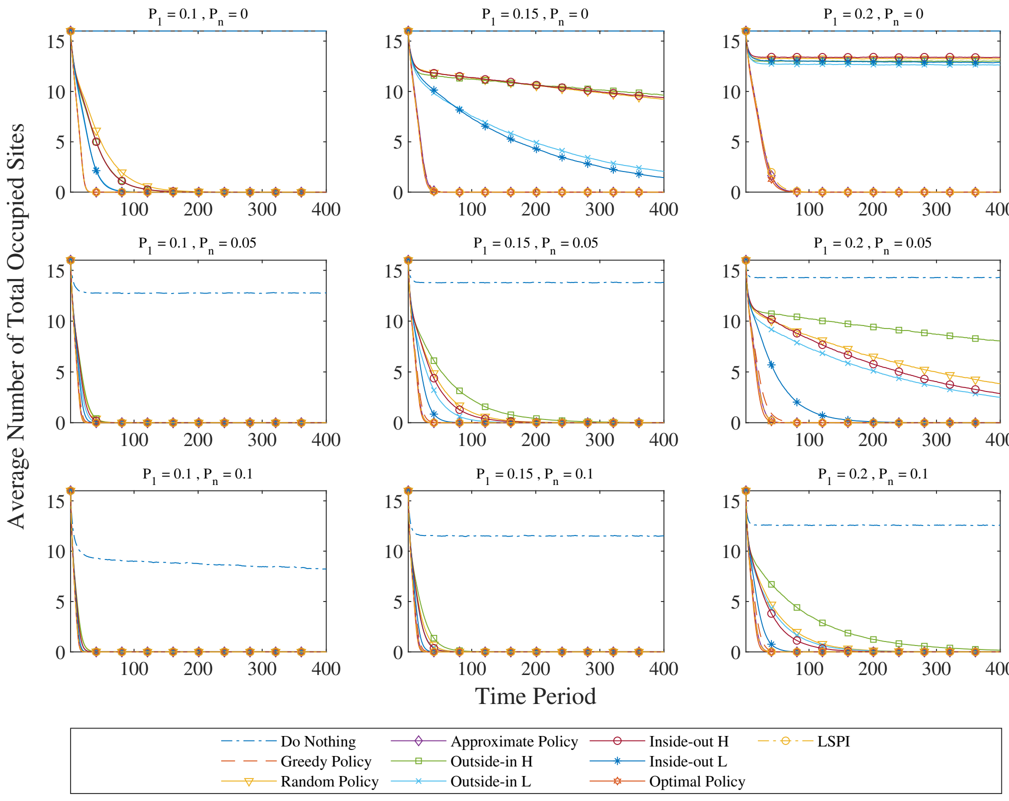


Figure A.3: Average number of occupied sites over ten thousand replications and four hundred periods for different policies for a 4x4 grid situation. The spontaneous infection parameter is set to zero (p_0=0). Action is assumed to be completely effective ($p_t = 1$).

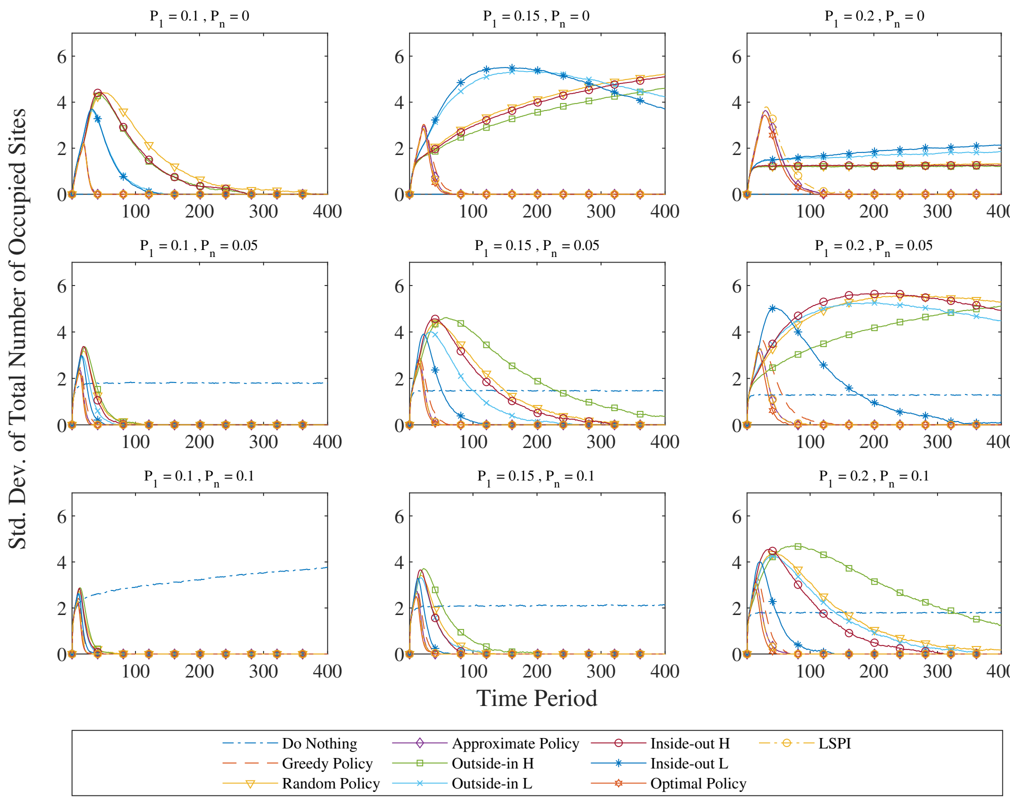


Figure A.4: Standard deviation of the number of occupied sites over ten thousand replications and four hundred periods for different policies for a 4x4 grid situation. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be completely effective ($p_t = 1$).

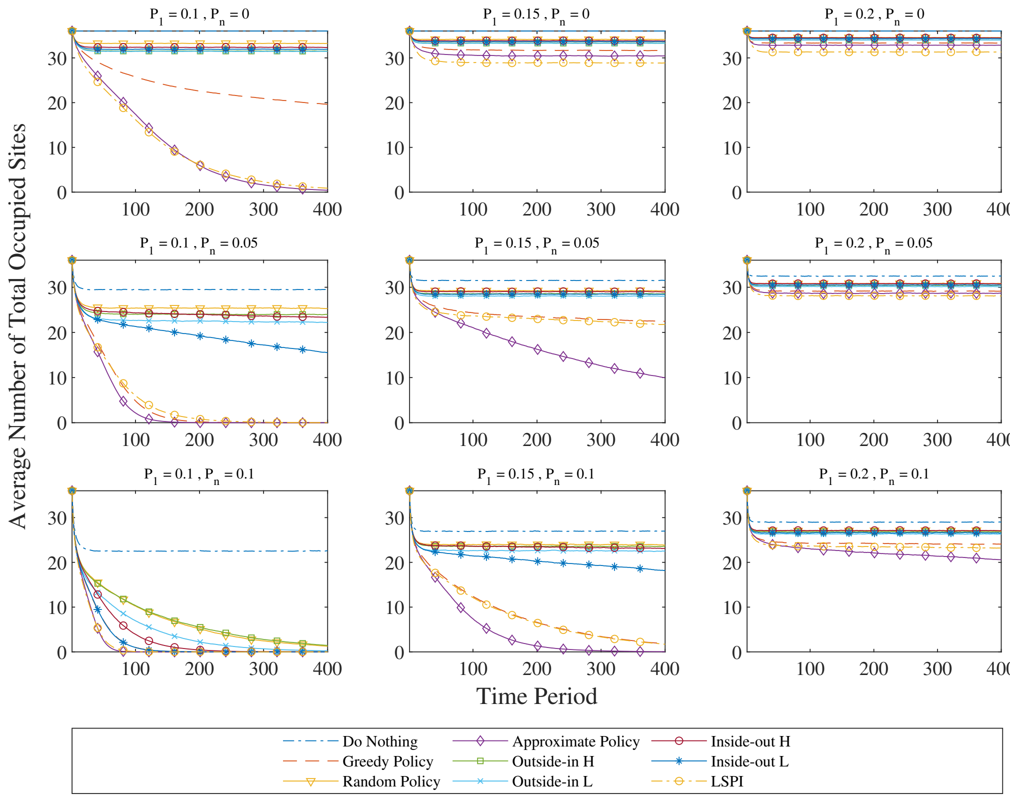


Figure A.5: Average number of occupied sites over ten thousand replications and four hundred periods for different policies for a 6x6 grid situation. The spontaneous infection parameter is set to zero (p_0=0). Action is assumed to be partially effective ($p_t = 0.7$).

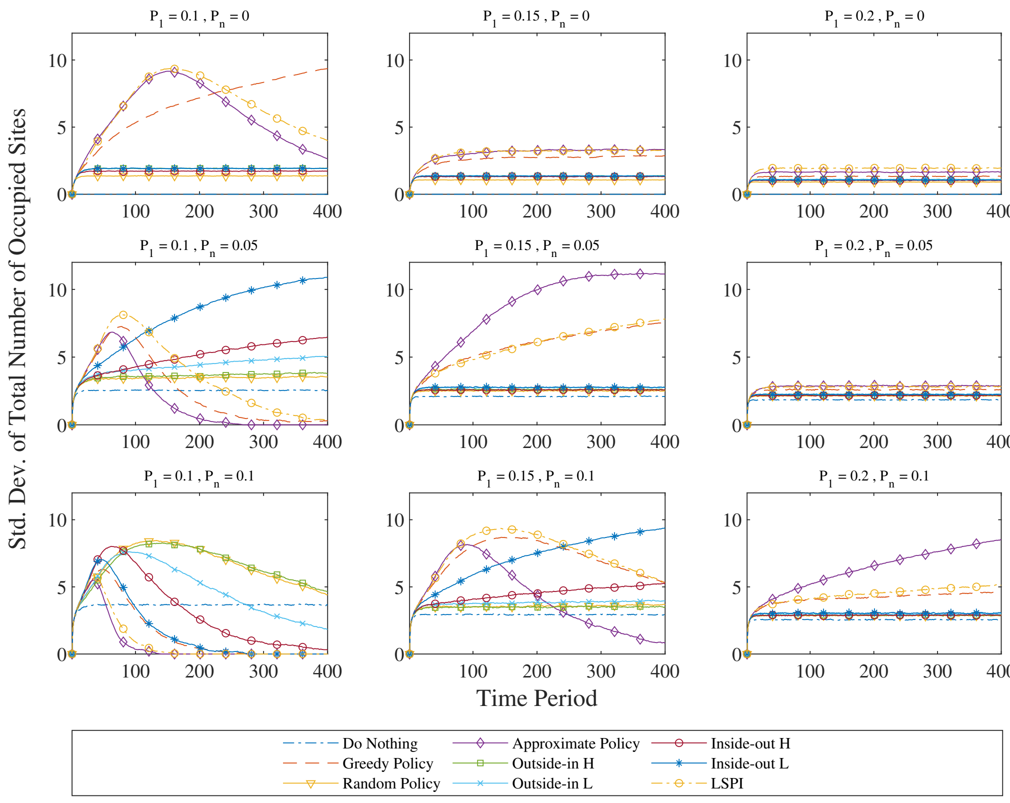


Figure A.6: Standard deviation of the number of occupied sites over ten thousand replications and four hundred periods for different policies for a 6x6 grid situation. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be partially effective ($p_t = 0.7$).

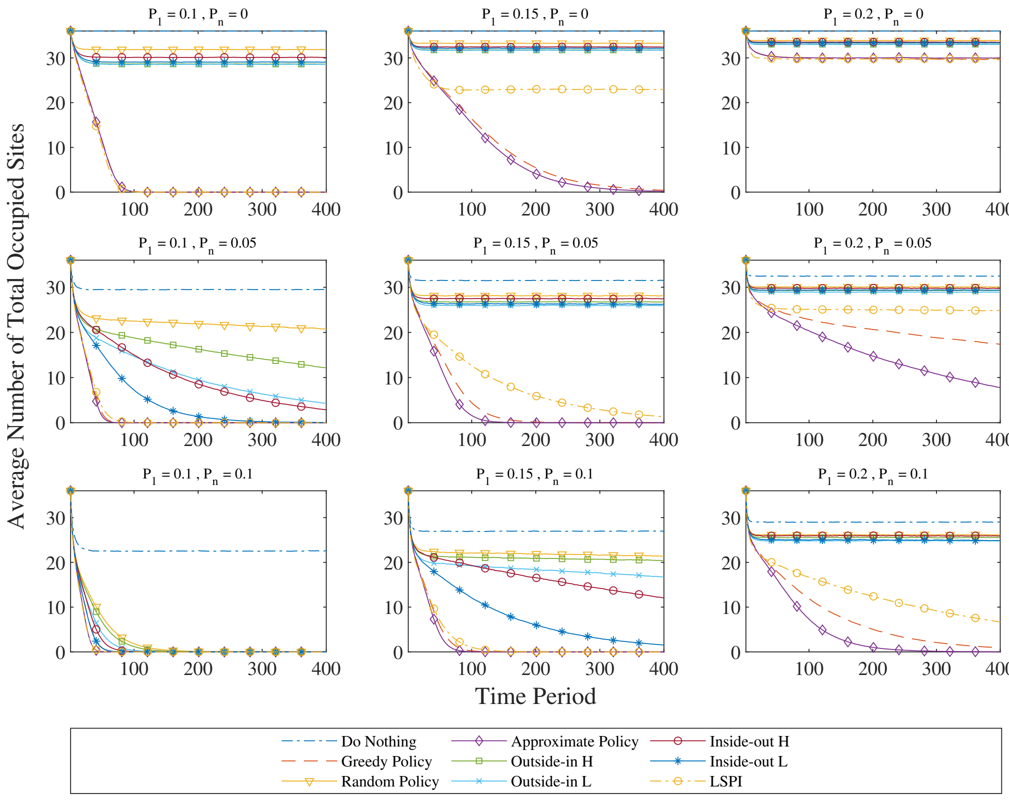


Figure A.7: Average number of occupied sites over ten thousand replications and four hundred periods for different policies for a 6x6 grid situation. The spontaneous infection parameter is set to zero (p_0=0). Action is assumed to be completely effective ($p_t = 1$).

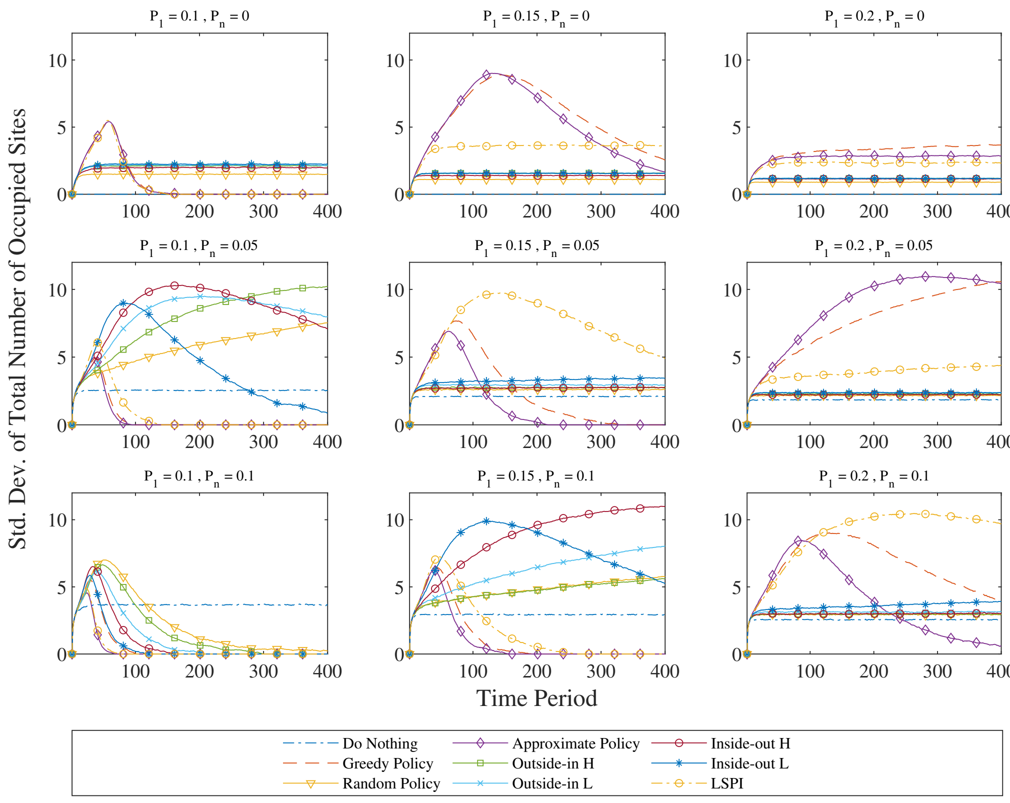


Figure A.8: Standard deviation of the number of occupied sites over ten thousand replications and four hundred periods for different policies for a 6x6 grid situation. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be completely effective ($p_t = 1$).

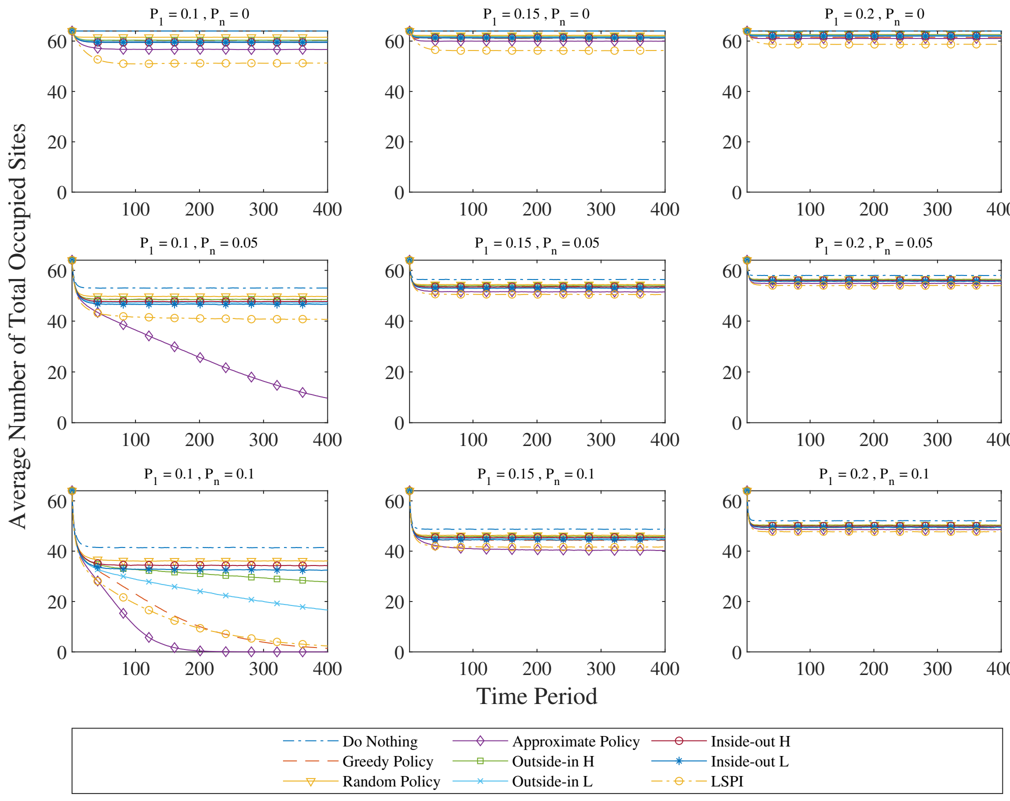


Figure A.9: Average number of occupied sites over ten thousand replications and four hundred periods for different policies for a 8x8 grid situation. The spontaneous infection parameter is set to zero (p_0=0). Action is assumed to be partially effective ($p_t = 0.7$).

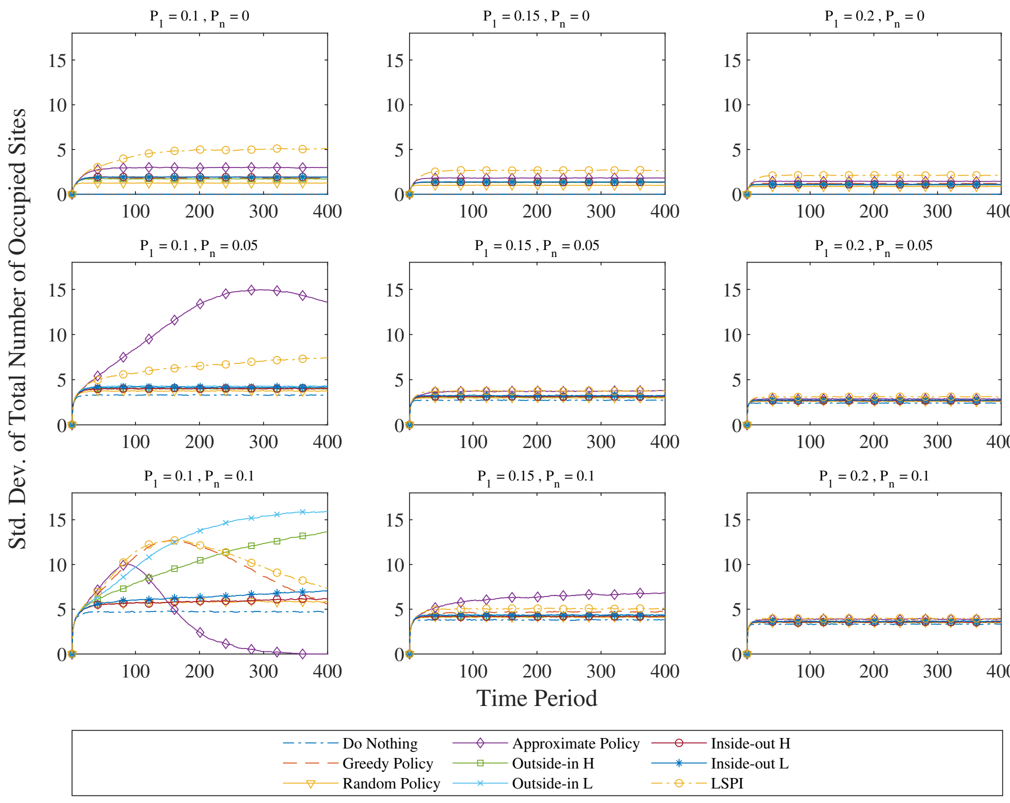


Figure A.10: Standard deviation of the number of occupied sites over ten thousand replications and four hundred periods for different policies for a 8x8 grid situation. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be partially effective ($p_t = 0.7$).

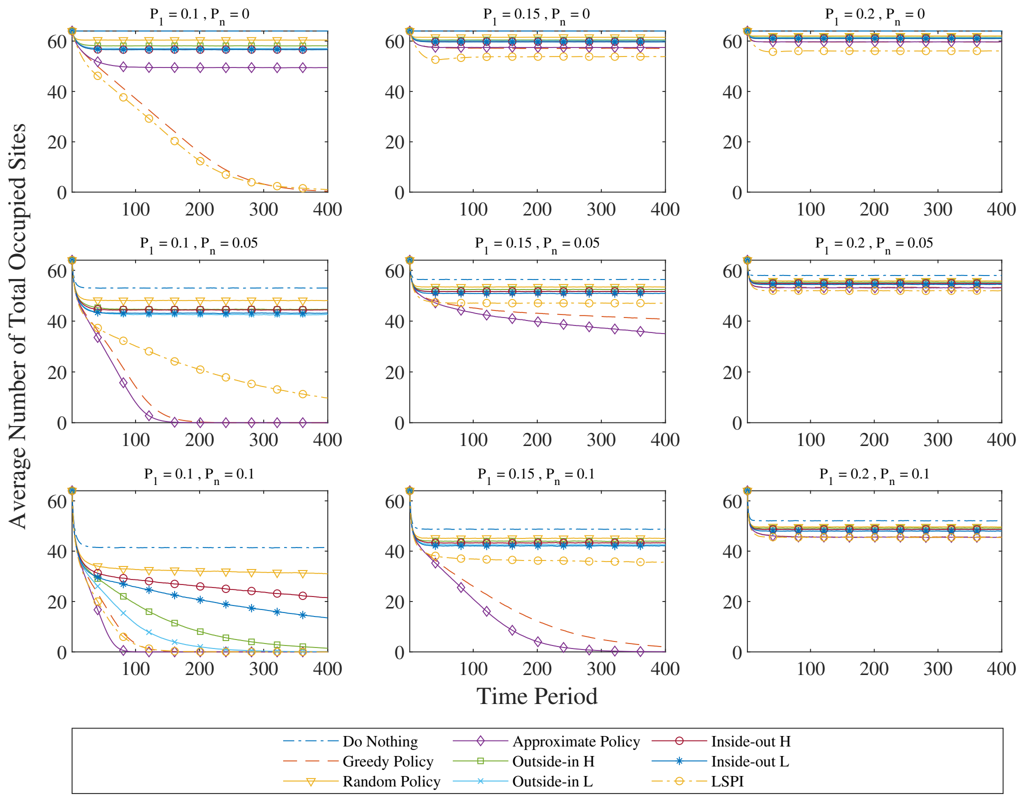


Figure A.11: Average number of occupied sites over ten thousand replications and four hundred periods for different policies for a 8x8 grid situation. The spontaneous infection parameter is set to zero (p_0=0). Action is assumed to be completely effective ($p_t = 1$).

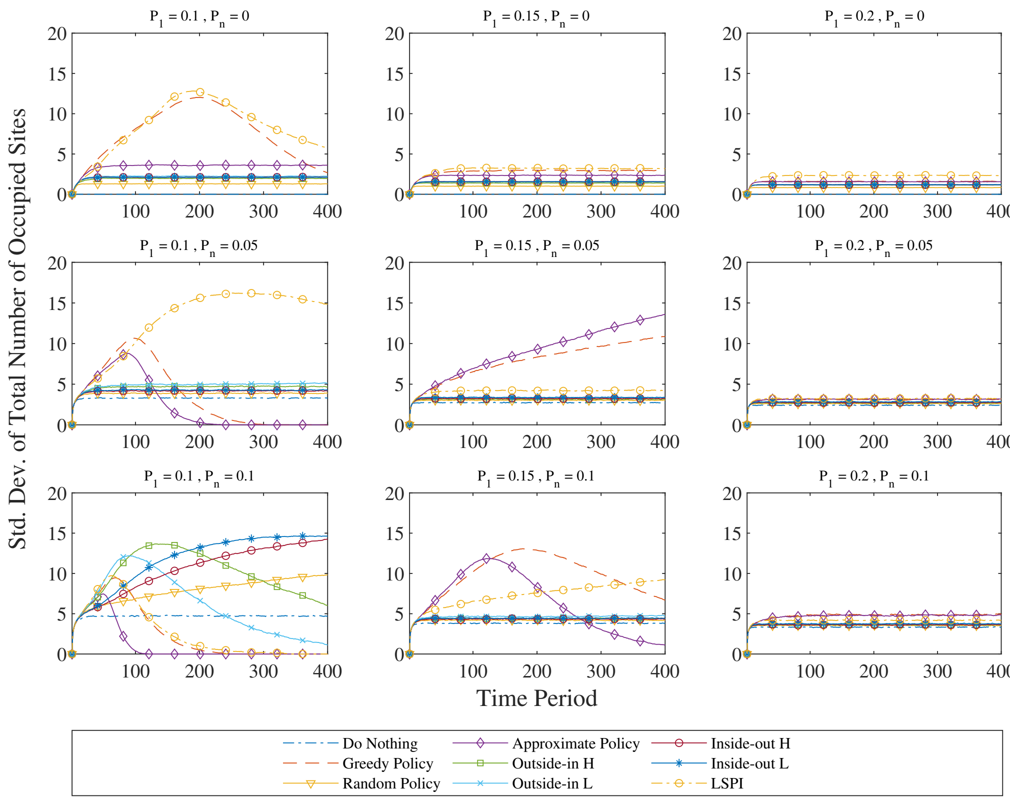


Figure A.12: Standard deviation of the number of occupied sites over ten thousand replications and four hundred periods for different policies for a 8x8 grid situation. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be completely effective ($p_t = 1$).

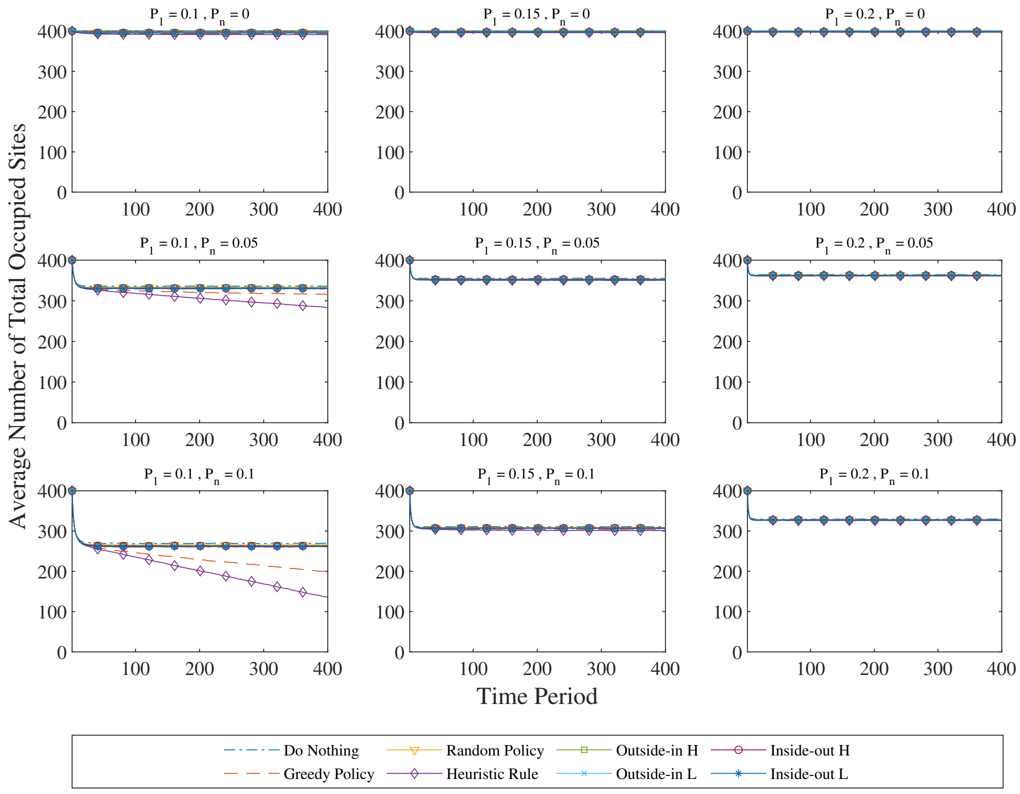


Figure A.13: Average number of occupied sites over a thousand replications and four hundred periods for different policies for a 8x50 grid situation. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be partially effective ($p_t = 0.7$).

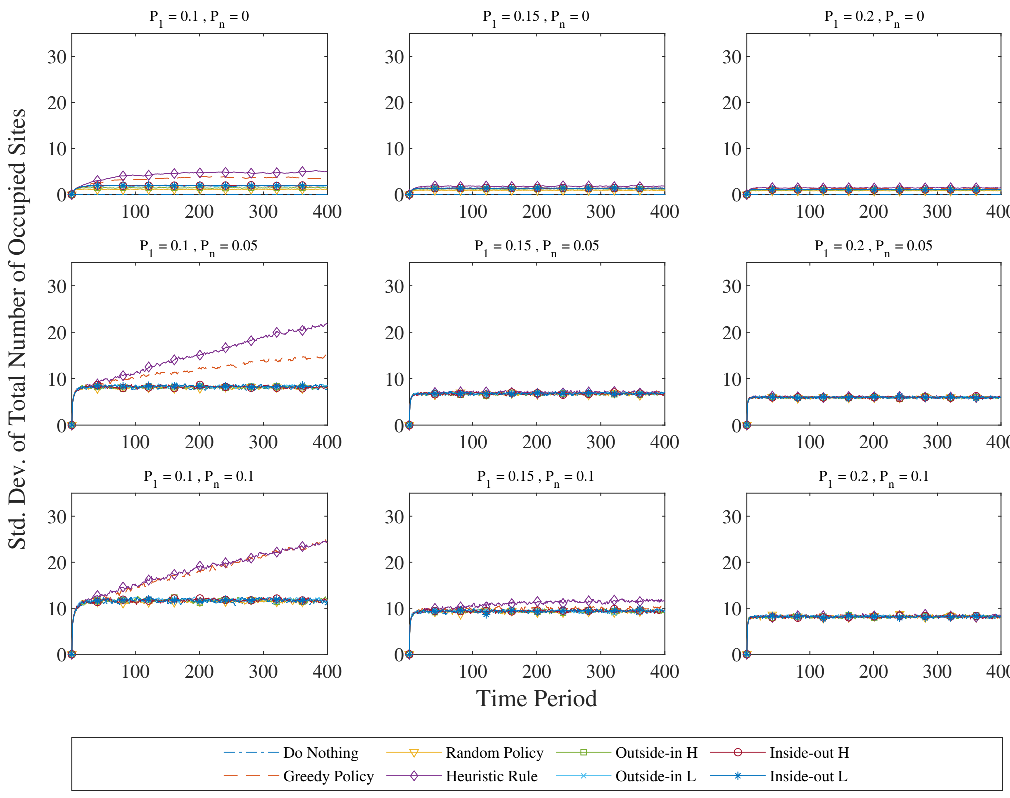


Figure A.14: Standard deviation of the number of occupied sites over a thousand replications and four hundred periods for different policies for a 8x50 grid situation. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be partially effective ($p_t = 0.7$).

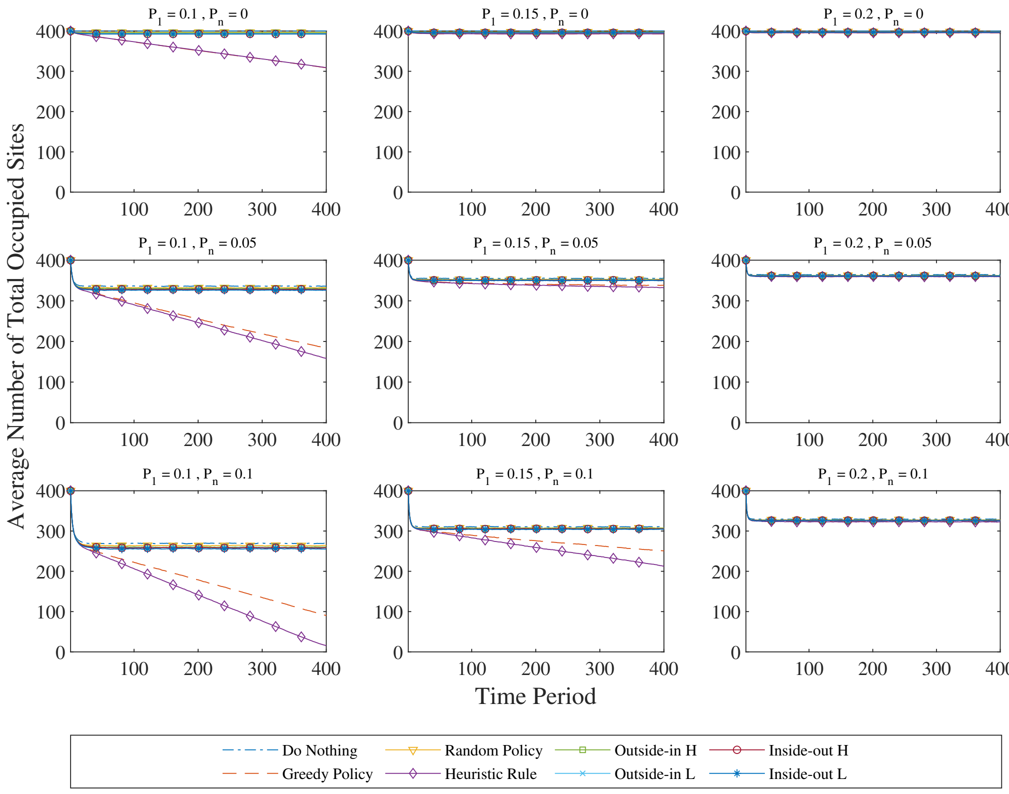


Figure A.15: Average number of occupied sites over a thousand replications and four hundred periods for different policies for a 8x50 grid situation. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be completely effective ($p_t = 1$).

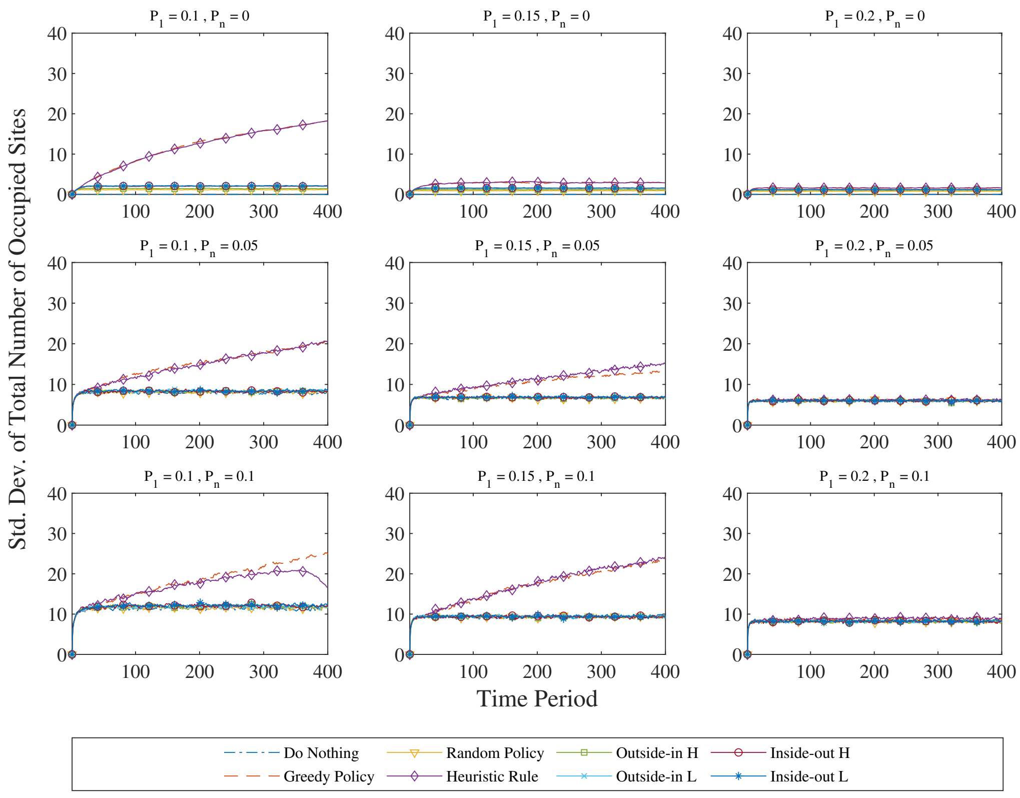


Figure A.16: Standard deviation of the number of occupied sites over a thousand replications and four hundred periods for different policies for a 8x50 grid situation. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be completely effective ($p_t = 1$).

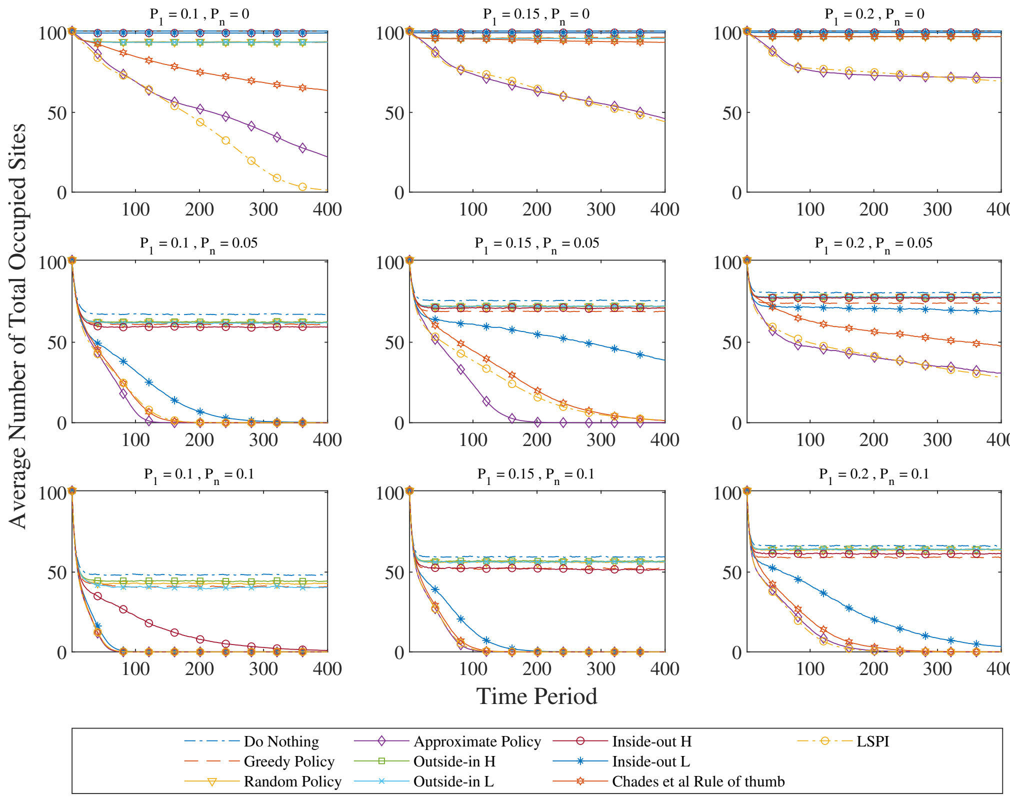


Figure A.17: Average number of occupied sites for four hundred periods for different policies for a multi-layer star network with 101 nodes. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be partially effective ($p_t = 0.7$).

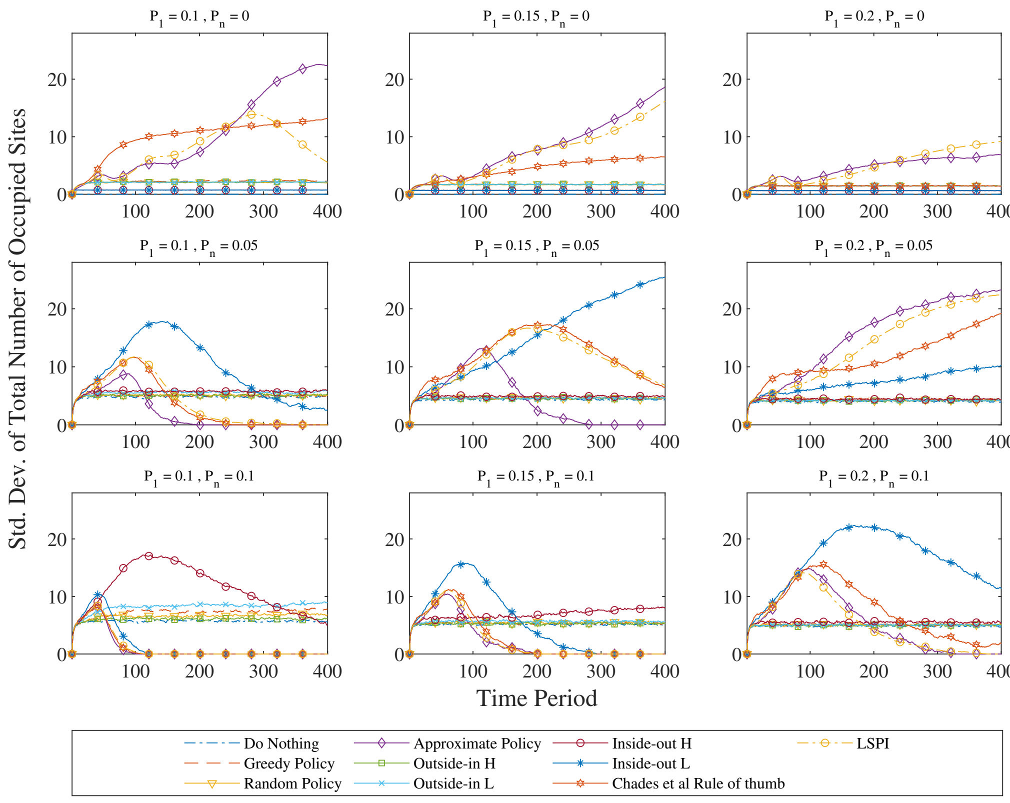


Figure A.18: Standard deviation of the number of occupied sites over a thousand replications and four hundred periods for a multi-layer star network with 101 nodes. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be partially effective ($p_t = 0.7$).

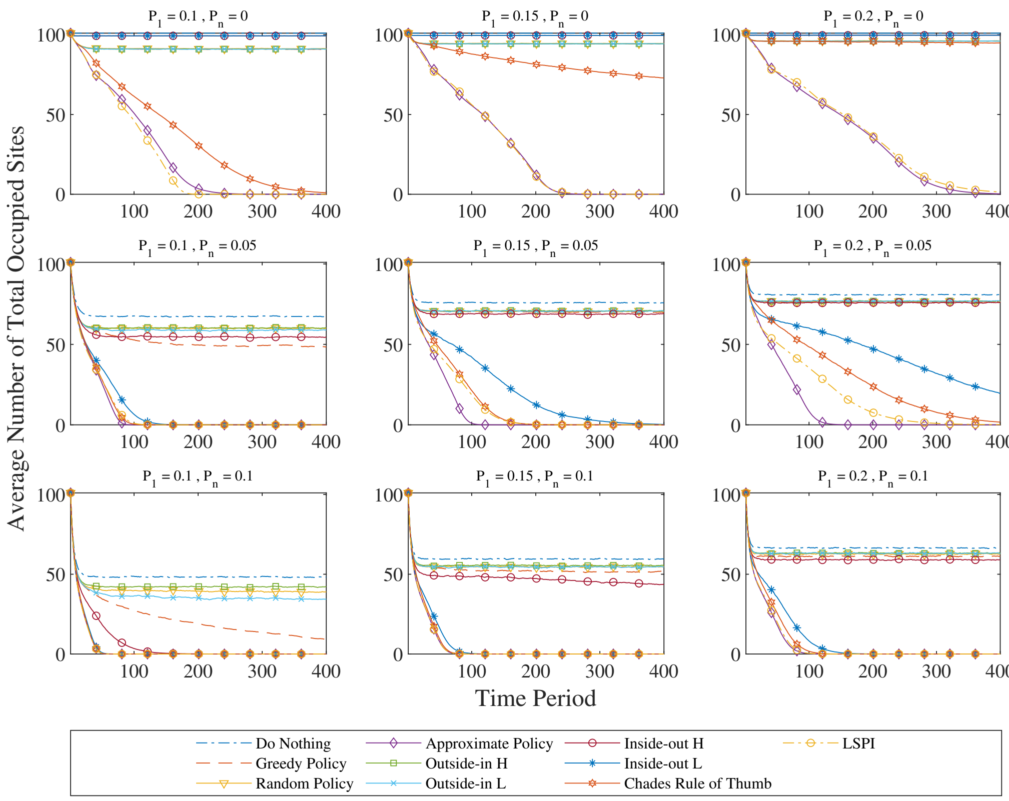


Figure A.19: Average number of occupied sites over a thousand replications and four hundred periods for different policies for a multi-layer star network with 101 nodes. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be completely effective ($p_t = 1$).

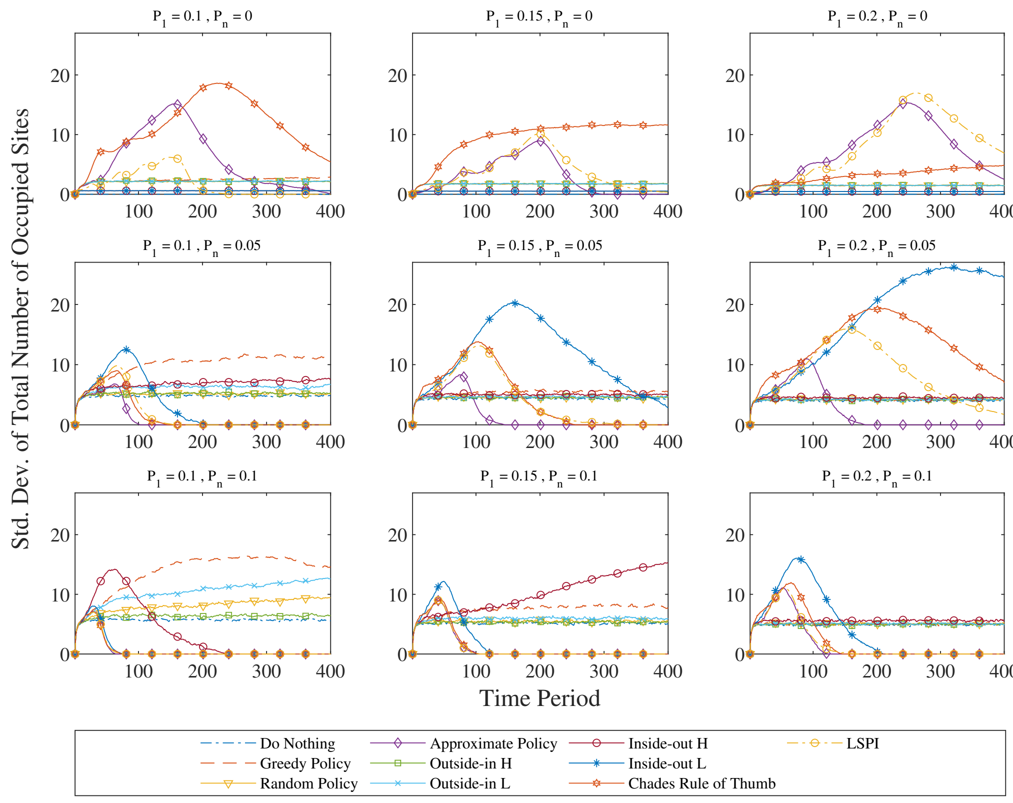


Figure A.20: Standard deviation of the number of occupied sites over a thousand replications and four hundred periods for a multi-layer star network with 101 nodes. The spontaneous infection parameter is set to zero ($p_0 = 0$). Action is assumed to be completely effective ($p_t = 1$).