AN ABSTRACT OF THE DISSERTATION OF

Shan Xue for the degree of Doctor of Philosophy in Computer Science presented on March 13 2020.

Title: Scheduling and Online Planning in Stochastic Diffusion Networks

Abstract approved: _____

Alan P. Fern

Diffusion processes in networks are common models for many domains, including species colonization, information/idea cascade, disease propagation and fire spreading. In diffusion networks, a diffusion event occurs when a behavior spreads from one node to the other following a probabilistic model, where the behavior could be species, an idea, a virus, fire, etc. In the real world, in addition to observing diffusion processes, people are usually able to control the influence of diffusion by conducting operations on each individual node or node groups. Then the diffusion network control problem is to decide how to perform possible controls in order to maximize or minimize the range of diffusion, especially when there is a limited resource for doing controls.

Diffusion network control problems are challenging for most AI planning techniques. The complexity comes from highly stochastic exogenous events, a large action branching factor (the number of combinations of individual operations), a long time horizon, and the need to reason about numeric resource limits. In this thesis, we explore approaches that offer high-quality policies of controlling diffusion processes in large-scale networks.

We first propose a non-adaptive policy in conservation planning, where the goal is to encourage species spread in a long term. Given a set of control operations of interest, this policy specifies the deadline of taking each operation, so that the resource is used with the most flexibility while keeping the loss of diffusion influence within a desired ratio. This is particularly applicable in cases where a domain expert can develop a set

of control operations that captures their own objectives. Then our approach provides a way of trading off diffusion influence and resource usage.

We further propose a fully adaptive approach for this conservation planning problem by computing a Hindsight Optimization (HOP) solution at every time step. Instead of computing a HOP action in the traditional way which is linear in the number of actions, we take advantage of its separable structure and develop an effective algorithm that scales for exponentially large, factored action spaces. From experiments on both synthetic and real data sets, we show that our algorithm returns near-optimal HOP solutions while scaling to large problems.

Moreover, we extend our implementation of HOP policy to a general framework of online planning for diffusion network control problems. In particular, we give a general and formal representation of diffusion network problems. Our framework proposes a schema of effectively computing multiple lookahead policies, some of which have been successfully applied to various probabilistic planning problems. We evaluate our approach on diffusion network control problems in conservation planning, epidemic control and firefighting. The experimental results demonstrate the behaviors of these lookahead policies and the advantage of each in different domains.

Scheduling and Online Planning in Stochastic Diffusion Networks

by

Shan Xue

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented March 13 2020
Commencement June 2020

Doctor of Philosophy dissertation of Shan Xue presented on March 13 2020.

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Shan Xue, Author

ACKNOWLEDGEMENTS

My long journey at OSU could not be finished without the help and support from many professors, researchers, administrative staffs, graduate students, friends and my family. It is my great pleasure to express my appreciation to people who have given me guidance, support and encouragement over the years.

I am extremely fortunate to have Prof. Alan Fern as my major advisor. I would not have the chance to study in OSU without his support. I am often amazed by his insights, clear thought, and focus on details. He is incredibly patient when inspiring my thinking and learning. In spite of his extremely busy schedule, he provides critical feedback on writing and taught me skills of presentation. I am particularly grateful for his attention to my well-being and consideration of my difficult situations. He has also been extraordinarily supportive of my job search.

Special thanks to all the other professors in my Ph.D. committee: Prof. Tom Dietterich, Prasad Tadepalli, Lan Xue, and Gary Egbert. I really appreciate their time as they always responded quickly to my requests and found time in their busy schedules for my program meetings. For some graduate students, it is often a painful and time-consuming process to have professors agree on a time. But it has always been simple for me. I am also grateful for their valuable feedback on my research during the exams. In particular, Prof. Dietterich shared his knowledge of *fire management* and suggested multiple factors to make my model more realistic; Prof. Tadepalli inspired me the connections and differences between my approaches and other alternatives; Prof. Xue explored the requirements in my probabilistic model, along with possible relaxation and extension; Prof. Egbert suggested a deeper thought of wider applications of my approaches.

This dissertation contains important contributions of my collaborators. My first project on conservation planning was brought in by Daniel Sheldon when he worked as a postdoc of Prof. Dietterich. His knowledge of related work made it easier to get into the problem in a short time and he was patient to go through a proof step by step with me. We continued to collaborate through emails and telecoms after he got a faculty job at University of Massachusetts Amherst. In addition, I also thank Prof. Scott Sanner for introducing Zhenyu Yu and his Influenza problem to us. This is an important domain

in this dissertation for generalizing my approach schema. During Zhenyu's short visit to OSU, we worked on domain details and he also suggested some possibilities of future collaboration.

I would like to express my gratitude to several excellent researchers that I worked with. I enjoyed working with Murugeswari Issakkimuthu when she was visiting OSU. Thank her for working with me on tedious details of generating paths of constraints from trees and setting up encodings of Mixed Integer Programs. We also exchanged experiences of using and debugging CPLEX. During our group meeting, Prof. Alan Fern, Prasad Tadepalli and Roni Khardon closely examined our work and provided guidance on algorithm improvement. Thanks to Forrest Bao, who I met at an AAAI conference. I am highly touched by his enthusiasm for research and appreciate his inspiration. Moreover, it was a wonderful experience to work on a Machine Learning problem in fashion marketing with Ruirui Zhang. I am thankful to have this opportunity to apply AI and ML techniques to solve a real-world problem in an interesting discipline.

I would like to thank all the faculties in the research group of Artificial Intelligence and Machine Learning: Prof. Tom Dietterich, Alan Fern, Prasad Tadepalli, Xiaoli Fern, and Weng-Keen Wong. Every professor taught me a variety of AI and ML approaches. I also appreciate the opportunities to learn from other researchers through regular AI seminars and EECS colloquium. The organizers have done an excellent job. There are very interesting topics and the talks are usually informative and inspiring.

Thanks to all EECS administrative staff. Ferne Simendinger and Nicole Thompson are always available to assist me. They have answers to all the questions I asked, and always act quickly for scheduling events and processing my paperwork. Todd Schechter and Mike Sanders provide perfect IT support, especially the cluster resource and poster printing service.

I am grateful to have many talented peers and lab-mates in KEC. They include but not limited to Jun Yu, Janardhan Doppa, Javad Azimi, Yan Zhang, Aswin Nadamuna, Jervis Pinto, Tao Sun, Xu Hu, Saikat Roy, Xinze Guan, Mandana Hamidi, Liping Liu and Darren Marshall. I am fortunate to know them and everyone is generous to offer advice and share valuable experience with me. Many of them are no longer at OSU now, but continue to provide wise counsel and professional referrals.

I would also like to thank my close friends Lu Qin, Ruirui Zhang, Fan Zhang for their enjoyable company. Though we have been living in different cities, Lu Zhang, Meng Chen

and Dan Su always express their constant concern and encouragement, without which I would not survive the hard times of this program. Thanks to the most brave scuba diving team, for sharing seafood they hunted all year round. Also, a big thanks is given to Dr. Akcali for her professional help with my anxiety and depression.

Finally, my special appreciation goes to my family, for their love, support, understanding and sacrifice. In very hard times, especially when I was not there taking care of them though I should, they kept bad news from me, suffered pain and fought difficulties. They gave me everything but asked for little. The last word of acknowledgement is saved for my grandmother. I dedicate what I have achieved to the memory of her. In my heart, she was, and will be, always by my side.

TABLE OF CONTENTS

Page

1 Introduction 1

1.1 Diffusion Networks . . . 2

1.2 Controls in Diffusion Networks . . . 3

1.3 Non-adaptive Planning for Diffusion Network Control . . . 3

1.4 Fully Adaptive Planning for Diffusion Network Control . . . 4

1.5 Contributions of This Thesis . . . 5

2 Scheduling Conservation Designs for Maximum Flexibility 7

2.1 Introduction . . . 7

2.2 Related Work . . . 10

2.3 Problem Statement . . . 12

2.3.1 Basic Concepts . . . 13

2.3.2 Stochastic Optimization Problem . . . 14

2.3.3 Deterministic Optimization Problem . . . 16

2.4 Approach . . . 20

2.4.1 Set-Weighted Directed Steiner Graph Formulation . . . 20

2.4.2 Primal-Dual Algorithm . . . 25

2.5 Experiments . . . 31

2.5.1 Evaluation of Primal-Dual Algorithm on Real Conservation Map . 31

2.5.2 Evaluation of Primal-Dual on Synthetic Maps . . . 36

2.5.3 Early-Stopping for Trading Off Flexibility and Reward . . . 38

2.6 Summary . . . 38

3 Dynamic Resource Allocation for Conservation Planning 48

3.1 Introduction . . . 48

3.2 Related Work . . . 48

3.3 Problem Statement . . . 50

3.3.1 Population Model . . . 50

3.3.2 Adaptive Planning Problem . . . 50

3.4 Approach . . . 51

3.4.1 Hindsight Optimization for Conservation Planning . . . 51

3.4.2 Dual Decomposition for HOP . . . 53

3.4.3 Dual Decomposition for Baselines . . . 57

TABLE OF CONTENTS (Continued)

Page

3.5 Experiments . . . 59
3.5.1 Performance of Dual Decomposition . . . 59
3.5.2 Adaptive Planning Results on Synthetic Maps . . . 60
3.5.3 Adaptive Planning Results on Real Map . . . 62
3.6 Summary . . . 63

4 A Framework for Online Planning in Diffusion Networks 65
4.1 Introduction . . . 65
4.2 Related Work . . . 66
4.2.1 Epidemic Control . . . 66
4.2.2 Stochastic Firefighting . . . 68
4.3 Problem Statement . . . 70
4.3.1 Basic Notation . . . 70
4.3.2 Stochastic Planning Problem . . . 71
4.3.3 Futures in Diffusion Networks . . . 72
4.4 Approach Schema . . . 73
4.4.1 Lookahead Policies . . . 73
4.4.2 A Framework for Computing Lookahead Policies . . . 75
4.5 Experiments . . . 78
4.5.1 Epidemic Control of Influenza . . . 78
4.5.2 Stochastic Firefighting . . . 83
4.6 Summary . . . 87

5 Lessons Learned and Future Work 88
5.1 Lessons Learned . . . 88
5.2 Future Work . . . 90
5.2.1 Scheduling Conservation Designs for Maximum Flexibility . . . 90
5.2.2 Online Planning in Diffusion Networks . . . 90

Bibliography 91

Appendices 97
A MIP Encoding . . . 98

LIST OF FIGURES

Figure Page

2.1 Example scenario graph ($N = 3$) for problem with parcels $p_1 = \{a, b\}$, $p_2 = \{c\}$. The schedule ($\pi(p_1) = 0, \pi(p_2) = 3$) is also illustrated, using shaded boxes to indicate purchased nodes and heavy line weights to indicate purchased edges. Vertices representing occupied patches under this schedule are colored red. 19

2.2 Description of the reduction from set cover to SW-DSG with a single terminal vertex and a scenario graph. 25

2.3 MIP for the SW-DSG problem and the corresponding dual LP of the MIP's LP-relaxation. The SW-DSG problem is defined by a graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, a root vertex r, a set of terminal vertices T, and a set of edges sets $\mathcal{E} = \{E_s : s = 1, \ldots, M\}$, where each $E_s \subseteq \mathbb{E}$. 40

2.4 Rewards of PD solutions *w.r.t* the number of cascade scenarios. 41

2.5 Rewards of PD schedules *w.r.t.* time horizon H. 41

2.6 Cost curves of PD schedules for horizons 20 to 100. 42

2.7 (Left) Original conservation design used for scheduling shown as green shaded parcels. Free (zero-cost) parcels are also shaded in dark grey and red '+' indicates initially occupied patches. (Right) The top row shows the parcels purchased (shaded green) by our PD schedule over a horizon of 100 years. The middle row shows the population spread over the same horizon for the schedule, where lighter red shading of a patch indicates a smaller probability of being occupied (as measured by 20 simulations). The bottom row shows the population spread of the upfront schedule over a horizon of 100 years. 43

2.8 Cost and objective value of problems on map 768. The horizontal axis varies the amount of budget used to compute the upfront solution that is used as the conservation design given to the scheduling algorithms. 44

2.9 Rewards of primal-dual schedule and upfront schedule on map 1027. . . . 44

2.10 Average cost curves of PD schedules on 10 maps. 45

2.11 Cost curves of PD-ES schedules with pruning. The red line ($\hat{\varepsilon} = 0$) shows the cost curve of non-early-stopping PD schedule. 46

LIST OF FIGURES (Continued)

Figure Page

2.12 Rewards of primal-dual schedules with early-stopping. The number below each data point indicates the percentage of PD-ES reward over PD reward. 47

3.1 Visualization of An Example Future. Suppose there are totally 3 patches: u, v, and w, then each patch repeats at each time step. The edge indicates the colonization from one patch to the other at a certain time step. 51

3.2 (Left) Grid Map 1. (Right) Grid Map 2. For both maps, the number marks the index of patches and each grid represents one parcel. The initially occupied patches are numbered in red and the free parcels are shaded in green. 61

3.3 (Left) Initial state of the real map. (Right) The population distribution after 20 years on the real map. 64

3.4 Purchases and population spread of HNoop. Parcels shaded in green are purchased with a probability of ≥ 0.5. Yellow is used for parcels with purchase probability of < 0.5. Patches are colored by the probability of being occupied (lighter color indicates lower probability). 64

3.5 Purchases and population spread of the HOP policy. The color setting is the same as Figure 3.4. 64

4.1 Grid maps of Firefighting. Each cell represents a land patch. Red cells are initially *burning*. Green and yellow cells are *susceptible*. Yellow ones have more fuel than the green hence yellow cells are more likely to catch fire. 85

LIST OF TABLES

Table Page

2.1 Comparison of Primal-Dual (PD) with MIP and LP 33

3.1 Solution Quality and Run Time(H=20) 60

3.2 Solution Quality and Run Time(H=40) 60

3.3 Rewards on Different Maps 62

4.1 Objective Value and Computation Time 81

4.2 The Number of Infection on A Small Network (Gurobi) 82

4.3 The Number of Infection on A Large Network (DD) 83

4.4 Objective Value and Computation Time 86

4.5 Rewards of MIP 87

4.6 Rewards of MIP 87

LIST OF ALGORITHMS

Algorithm		Page
1	Primal-Dual Algorithm for SW-DSG	27
2	Dual Decomposition Algorithm	56

Chapter 1: Introduction

Many natural processes, such as the idea broadcasting in social networks and disease spreading in population, can be abstracted as a network with a diffusion process. The basic observation is that a behavior of an individual may trigger similar behaviors in its connections. And such viral behavior (information/idea/disease) propagates contagion from nodes to nodes over time following a certain spreading model, which is typically stochastic. Within some time horizon, the behavior can cascade over a significant portion of the network. For example, some species occupying one habitat patch may colonize other patches in a fragmented landscape. If the landscape contains suitable patches, the species would cover some of them after some period of time.

In reality, one would often like to control the diffusion process for optimizing the influence of a cascading behavior (e.g. to conserve some portion of the landscape for maximizing the number of patches covered by the species at the end of some time horizon). Usually, the controls would not only affect the diffusion at current time step, but also have long-term effects. Hence the optimization problem must include many time steps, making it more challenging to solve. Also, the controls do not directly affect the diffusion outcome. A diffusion process includes stochastic events that probabilistically depend on the controls. In other words, there is great uncertainty as to the actual result of the controls. Therefore, to decide the optimal controls, one must reason about a long-term stochastic influence.

Another complexity comes from the size of decision space. An atomic control in networks is often defined with respect to an individual node or a set of nodes. Then the action space is combinatorial due to the combinations of atomic controls. In online planning problems where the task is to find a sequential plan, the long time horizon adds one more dimension to the policy space.

In this dissertation, we present our progress in probabilistic planning with exponential state and action space by addressing several planning problems with diffusion networks. We propose several techniques to handle stochastic optimizations and leveraging large action space. Our solutions are shown to be effective and high-quality in a number of

domains with diffusion network control.

1.1 Diffusion Networks

A network (graph) is a widely used mathematical form to express a set of entities and their relationship, where entities are abstracted as nodes and edges between node pairs represent their connections. As a fundamental process in networks, diffusion is a behavior that cascades from nodes to nodes like epidemics. Some examples include:

Species conservation. A landscape of interest is divided into small land patches, each of which corresponds to a node in the network. Each node has two possible states at each time step: either unoccupied or occupied by the species, and only conserved patches may be occupied. An edge exists between two nodes if the species may colonize from the corresponding land patch to the other. When the species of interest are highly mobile like birds, we have edges between every pair of nodes and arrive at a complete graph.

Information diffusion. Nowadays, online social networks are major media for spreading information at a very large scale. Here information can be news, rumors, opinions, recommendations, etc. The network could be a social network of individuals or organizations. Edges represent a medium for communication or the passage of information.

Disease or virus propagation. For centuries, people have observed infectious diseases spreading through a susceptible population. The connection between people can be interpreted as whom-infects-whom. Computer virus diffusion can be modeled similarly over a computer network. A typical epidemic often spread rapidly by infecting a large number of people within a short period of time.

Fire spreading. Fire also spreads in a network of fragmented flammable surfaces. Fire can jump over small gaps and climb up walls. The diffusion model of a fire heavily depends on the combustible material present, as well as some factors that are independent of the nodes, such as weather.

Note that in practice these networks are usually large scale. Moreover, we are particularly interested in applications with stochastic diffusion processes, i.e. the spreading between two nodes follows a probabilistic model.

1.2 Controls in Diffusion Networks

One important purpose of studying diffusion networks is to make decisions about what actions to take before or during the diffusion process in order to control the diffusion influence. For instance, to prevent an infectious disease breakout, vaccinations are provided to susceptible individuals and treatments are given to infected patients. As another example, a commonly used marketing strategy is to select a group of influential people as seeds and let them introduce a new product to their connections and followers. Usually a type of control operation (e.g. a vaccination) is available to a large number of individual nodes in the network. Therefore, a possible decision is to take a control operation on a subset of nodes, which gives rise to a combinatorial decision space.

Above examples highlight the challenges in planning for diffusion networks, especially when the state space and action space are exponentially large. We aim to pursue a general schema for optimizing the diffusion in these networks by taking control actions during a time horizon. This probabilistic planning problem poses considerable challenges to general off-the-shelf planners. The complicating factors include: 1) highly-stochastic, exogenous dynamics that arise from the large-scale network and the spreading model, 2) the need to reason about spatio-temporal processes, 3) the combinatorial action space at each point of time that comes from all the investment combinations, and 4) the long time horizons of interest that must be considered. The general solution schema we provide suggests approximating the problem via sampling of the future dynamics and then studying the resulting deterministic optimization problem through a well-studied subproblem, or a compact encoding and possible computation speedup using techniques like Lagrangian Relaxation.

1.3 Non-adaptive Planning for Diffusion Network Control

With some simplification, a diffusion network control problem is more likely to allow efficient and near optimal approaches. One popular simplified problem is upfront planning, which assumes that all control actions are taken upfront with currently available resources. This version of problem has shown importance in real-world applications. In the viral marketing setting, Kempe et al. [34] address the problem of selecting k of individuals to initiate a diffusion process in order to maximize the eventual influence

of the diffusion. They also show that with several diffusion models, this problem is a submodular maximization problem and admits a greedy, but near optimal approach. Another upfront problem in conservation planning is done by Sheldon et al. [51]. Though their optimization problem is still difficult since it is not submodular, their approach can provide solutions with stochastic optimality guarantee.

1.4 Fully Adaptive Planning for Diffusion Network Control

An ideal approach would be fully adaptive planning, in which the planner faces a task of making sequential decisions in an interaction with the environment. In particular, at every decision step, the planner observes the state of the environment and returns an action that is executed immediately. As a feedback, the environment provides a numeric reward that specifies the immediate utility of taking that action in the current state. Such steps repeat until a terminal state is reached.

Compared to non-adaptive planning, the fully adaptive planning has the advantage of observing and making decisions based on the stochastic outcomes of diffusion. It may be very suboptimal to ignore the actual diffusion outcomes in domains with highly stochastic events. This is analogous to the comparison between open-loop and closed-loop controls. Moreover, in many real problems, resources for controls arrive in increments over time. Then only a small number of controls can be afforded at initial time step. In a long term, it is not reasonable and also suboptimal to only make upfront controls.

One popular framework for adaptive planning is Reinforcement Learning (RL) approaches, in which the planner explores the environment and learns a policy that maps states to actions. Pure RL algorithms require little knowledge about the specific problem as long as a Markov Decision Process (MDP) encoding is given. Typically, the policy is learned and improved through a trial-and-error process, where the agent observes current state, executes an action according to its algorithm, receives only a reward value as feedback, and then repeats. Unfortunately, without additional modeling and knowledge, pure RL algorithms are usually expensive to use in terms of how much experience is needed. Their complexity is even higher in problems with large state and action spaces since the planner aims to compute a policy over the entire reachable state space.

Recently, several techniques are proposed to accelerate the learning of pure RL. For example, in Hierarchical RL, a large task is decomposed into a hierarchy of smaller tasks.

The policy to be learned is then restricted to be consistent with this hierarchy. Ideally, the task hierarchy serves as a pruning technique for eliminating obviously incorrect strategies. However, it may occur that the constraints also prune the optimal solution and the returned policy is suboptimal.

Unlike RL approaches, lookahead policies make decisions through a domain model or simulator of the environment that allows forecasting into the future for a finite horizon. Using future scenarios, lookahead algorithms explicitly consider the space of future actions for optimizing the action values at the current state. There have been a variety of algorithms for building lookahead search trees, including model-based approaches such as RTDP [5] and AO* [7], along with simulation-based approaches such as UCT [36] and its variants, Policy Rollout, etc. Unfortunately, the performance of tree search algorithms largely depends on the action branching factor, which greatly limits the search depth in the tree in domains with large action spaces. While it is helpful to address this problem by pruning bad actions in the search tree (e.g. [45]), it is still a challenge to apply search methods to large factored action spaces.

1.5 Contributions of This Thesis

In Chapter 2, we address a non-adaptive planning problem of diffusion network control inn a conservation planning application. Given a design (set) of control actions, a planner is asked to schedule the time of taking each action in the design so that the budget is used with maximum flexibility while the population loss is within control. With the Sample Average Approximation (SAA), we reduce a special case of this probabilistic scheduling problem to a variant of the Steiner tree problem, which is shown to be computationally hard. We apply primal dual techniques to develop an algorithm that computes both a feasible solution and a lower bound on the quality of the optimal solution. Next, we add early-stopping to our primal dual algorithm in order to trade off between future population and budget flexibility. Experiments on synthetic and real data sets show that our algorithm can effectively return near-optimal solutions and is much more scalable than off-the-shelf optimizers.

In Chapter 3, we propose a fully adaptive approach based on Hindsight Optimization (HOP) for this conservation planning problem. At every decision epoch, we observe the current state and compute a HOP action for immediate execution. Compared to standard

implementations of HOP that enumerate each action, our algorithm shows scalability to the combinatorial action space via the use of a dual decomposition technique. Our experimental results demonstrate that the HOP solution can be computed effectively through our algorithm and HOP can significantly outperform myopic alternatives.

In Chapter 4, we formally set up the probabilistic diffusion network control problem in a general way. To provide a fully adaptive solution, we extend our HOP implementation to a general framework, which can incorporate multiple lookahead policies. Then we apply our approach schema to diffusion network control problems in epidemic control and stochastic firefighting. Evaluation results show the effectiveness of the approach. We also observe different strategies of each lookahead policy in each domain, which highlight their differences.

All together, this thesis explores probabilistic planning techniques for controlling large-scale diffusion networks which contain highly stochastic spreading events. Experimental results show that these techniques return high-quality plans on real-world problems. Hopefully, diffusion network control problems in other disciplines can adopt our approaches easily and successfully.

Chapter 2: Scheduling Conservation Designs for Maximum Flexibility

2.1 Introduction

Reserve site selection is a key problem in conservation planning in which planners select land regions to be designated as nature reserves, either to achieve general conservation goals such as preserving biodiversity, or to achieve specific goals such as supporting the recovery of an endangered species. In general, the problem is extremely complex as it involves reasoning about the interplay between uncertain population spread, uncertain future budgets, and other problem specific factors. In particular, properly assessing population spread involves reasoning about spatial aspects of landscapes such as their sizes, shapes, and connectivity. Further, the decision space is huge, consisting of all possible land investment combinations over time.

Given the above factors, it would be highly desirable for conservation practitioners to enhance their decision making via automated, or semi-automated, planning and scheduling algorithms. Unfortunately, this problem is beyond the scope of existing off-the-shelf stochastic planners and schedulers. This is largely due to the combination of enormous state and action spaces, the highly uncertain, exogenous dynamics, and the need for spatio-temporal reasoning. The main contribution of this paper is to make progress toward handling these complexities by studying a useful subproblem of conservation planning that can be used by practitioners on realistic scenarios. The general schema used to develop our algorithm is more widely applicable (see Section 2.6) and one that has not received significant attention in the AI community. Thus, we hope that this work will also inspire new specialized and general-purpose approaches for complex stochastic planning/scheduling problems.

Recently, Sheldon et al. [51] studied a restricted, but still challenging, version of the conservation planning problem, which we will refer to as *upfront conservation design optimization*. In this problem, the planner is given an upfront budget and a stochastic *metapopulation model* [26] that describes how the species under consideration will spread

throughout a landscape of available habitat. In addition the system is given information about the costs of potential land parcels that are available for purchase and conservation. The objective is to select a set of land parcels to immediately purchase and conserve, subject to the budget constraint, that will maximize the spread of the population within a specified time horizon.

A key simplification present in this problem is that all land parcels are assumed to be purchased upfront with the currently available budget. An advantage of this simplification is that it allows for a reasonably efficient and near optimal solution approach [51]. However, the upfront simplification limits the utility of the approach in a number of ways. First, conservation budgets generally arrive in increments over time, so it is unrealistic to purchase a large set of parcels in advance and restricting to a small set of parcels using the current budget may be very suboptimal in the long run. Moreover, it is often *unnecessary* to purchase parcels that are spatially remote from the current population until the species has spread enough to make them relevant to further population growth. Second, this upfront simplification requires planners to commit in advance to conservation strategies that may take many years to play out, which ignores the potential advantage of observing and responding to the stochastic outcomes of the population spreading process as it unfolds. For example, as the population spread is observed, it may be beneficial to divert money from failed subpopulations to purchase more parcels near thriving populations.

In contrast to upfront planning, an ideal approach would be *fully adaptive planning*, where, at regular decision epochs, the planner would make purchase decisions based on the most recent population and budgetary information. Unfortunately, no currently-available adaptive planning tools can scale to realistic conservation scenarios. This is due to the combination of an enormous state space (possible population and purchase configurations), enormous action space (possible subsets of land parcels to purchase), long horizons (tens to hundreds of years), and high degree of stochasticity in the population spread model.

Given the challenge of arriving at a fully adaptive solution, a main contribution of this paper is to introduce a problem that strikes an important middle-ground between the upfront and fully adaptive approaches. In particular, we consider *conservation design scheduling* for exploring the trade-off between future population and cost, where we are given an initial conservation design (i.e. a set of parcels to purchase) and are asked to

schedule the purchase time of each parcel in a way that (1) achieves a population spread over the time horizon within an arbitrary tolerance of population loss, and (2) maximizes purchase flexibility by delaying the specified purchase time (i.e. purchase deadline) for each parcel as long as possible.

This problem formulation simplifies over the fully adaptive problem in a number of ways. First, the set of parcels to be purchased is provided as input, which removes this degree of freedom from the planning problem. Second, and more significantly, in order to select an action for the current time step, the general case of fully adaptive planning requires computing a policy that dictates what to do at each possible future contingency, or at least the reasonably likely ones. In contrast, the space of possible schedules, our focus here, is much smaller than the space of policies or even partial policies. This allows for a compact encoding of the scheduling problem, which does not appear possible for the problem of computing full, or even, partial policies. This distinction between the fully adaptive and scheduling setting is akin to the distinction between closed-loop and open-loop planning, where generally computing closed-loop plans is considered to be more difficult than open-loop plans for large stochastic problems.

A solution to the above scheduling problem yields a useful tool to conservation planners, who can first develop conservation designs that capture their own complex decision-making objectives, perhaps with optimization software, and then schedule the purchases to obtain the most efficient and cost-effective implementation of that design. The conservation planner then has the flexibility to purchase the parcels at any time before the schedule-specified deadlines, knowing that the population spread will not be hurt too much by such purchase delays.

In addition, our scheduling problem can potentially be used as a component of an adaptive planner. A common and successful approach for many adaptive planning problems is *replanning*, where at each decision epoch a non-adaptive plan is computed from the current state and its first actions are executed. Our work enables a replanning approach that at each decision epoch first computes an upfront design using existing work [51], and then computes a schedule and purchases only the parcels scheduled to be purchased immediately. This purchase strategy would spend the minimum amount of budget at each step while guaranteeing a limited loss in population spread.

In addition to introducing and formalizing the problem of conservation design scheduling, the second contribution of this paper is to develop a principled algorithm for solving

it. The key idea is to apply the *Sample Average Approximation (SAA)* approach [50] in order to arrive at a novel deterministic optimization problem, for which we develop a principled solution with a motivation from its special case. In particular, when the approximated loss tolerance ratio is 0, our deterministic optimization problem is one of network cascade optimization, which we show is equivalent to a novel variant of the directed Steiner tree problem. In the traditional Steiner tree problem, graph edges are associated with costs, and the objective is to compute a Steiner tree with minimum cumulative edge cost. In our variant, the set-weighted directed Steiner graph problem, costs are associated with sets of edges (possibly non-disjoint) rather than individual edges. We show that this problem is computationally hard even under restrictions where the traditional problem admits an efficient solution. We then present an efficient primal-dual algorithm, which is guaranteed to compute both a feasible solution and a bound on the quality of the optimal solution. Then an early-stopping version of the algorithm provides a natural approach to explore the trade-off between future population and budget flexibility.

Our experiments on both real and synthetic data from a Red-cockaded Woodpecker conservation problem show that our primal-dual algorithm produces near optimal results and is much more scalable than standard optimization tools (CPLEX). We also show that the trade-off between population and budget allows for a flexibility of purchasing land parcels.

In what follows, Section 2.2 first presents related work, followed by our problem formulation in Section 2.3. Section 2.4.1 then shows how to reduce our subproblem to the set-weighted directed Steiner graph problem. Section 2.4.2 derives the corresponding primal-dual algorithm and a natural extension for the trade-off problem. Experiments are presented in Section 2.5. Finally we conclude and discuss future work.

2.2 Related Work

Previously, many different algorithms have been proposed to select reserve sites by formulating a numerical measure of reserve quality (together with the possible addition of constraints the reserve must satisfy) and then solving for the optimal set of sites under the proposed model (e.g. see the review article, Williams, Revelle, & Levin, 2005). Although the earliest reserve site selection algorithms largely ignored spatial considera-

tions, many newer models incorporate spatial objectives or constraints directly into the optimization problems. Williams, Revelle, and Levin argue that a primary reason for the importance of spatial attributes is the fact that they capture properties of the landscape that are favorable for the underlying population dynamics, and that an important, but computationally difficult, research direction is to directly optimize with respect to a model for the population dynamics instead of using spatial attributes as a proxy. This is the direction that we are following in this paper by addressing the problem of spatial conservation planning with respect to a specific and widely adopted model of population dynamics.

A recent approach that explicitly reasons about a population dynamics model is the work of Sheldon et al. [51] on the upfront conservation design problem, as described in Section 2.1. In order to cope with the stochasticity of the model, the *Sample Average Approximation* (SAA) approach was employed to transform the stochastic problem into a deterministic combinatorial optimization problem. This problem was then encoded as a Mixed Integer Program (MIP) and solved using state-of-the-art solvers. While that approach was able to solve reasonably large problems via various speedup techniques, the scalability is still limited to a relatively small number of "sample scenarios" used by SAA, which controls the accuracy of the approach. Kumar et al. [37] have addressed this aspect of the approach. Lagrangian relaxation was used to decompose the SAA problem into independent subproblems that could each be solved in a practical time frame, possibly in parallel, by standard optimizers. This was shown to significantly reduce the runtime dependence on the number of SAA samples used.

Unfortunately, directly extending the above approach to compute multi-stage adaptive solutions, where the budget arrives in increments over time, does not seem practical. One attempt at this for two-stage problems was considered by Ahmadizadeh et al. [1]. They explore re-planning using a two-stage non-adaptive problem formulation and find that it can indeed offer advantages over upfront planning. In their setting, the budget split and decision epochs are manually fixed. Unlike that work, our work explicitly separates the decision of which parcels to buy from the decision of when to buy them (the focus of this work), so that we may develop efficient special-purpose algorithms for the latter problem that scale much more easily to bigger problems and more stages.

There are several existing approaches that might be considered for a fully adaptive solution to the conservation problem. For example, the fully adaptive problem can be

encoded as a Markov Decision Process (MDP), but the resulting state and action spaces would be far too big for state-of-the-art solvers. For instance, recent advances in solving large spatio-temporal MDPs [14] require significant restrictions to the solution space, which are not acceptable in our application. As an existing approach for stochastic planning that has been successfully applied by Bent et al. [6], Chang et al. [10], Chong et al. [12] and Yoon et al. [59], Hindsight Optimization samples the future outcomes and optimistically estimates the state value based on the determined futures. However, when the action space is huge, this approach would have computational problems as current algorithms require enumeration of all the candidate actions when approximating the state value. Another approach would be to formulate the adaptive planning problem as a multi-stage stochastic integer program. However, the size of such a problem formulation scales exponentially with the number of stages, and the running time is already very costly for a single stage [51], or for a two-stage problem in a simpler setting that is not fully adaptive [1].

Recently, Golovin et al. [25] proved that a simple greedy planning strategy provides near-optimal solutions in an adaptive conservation setting that at first appears similar to ours. However, in order to provide approximation guarantees, the authors restrict the population dynamics so that no spread occurs between distinct land parcels. While this may be a reasonable assumption for slow-moving species such as certain insects, which were the focus of that work, it ignores critical aspects of the population dynamics of highly-mobile animals such as birds, including the Red-cockaded Woodpecker on which our experiments are based.

2.3 Problem Statement

In this section, we first introduce the basic terminology of conservation design planning and define our main stochastic optimization problem. Next, we describe how the *Sample Average Approximation* (SAA) is used to transform this problem into a deterministic optimization problem, which is the focus of the remainder of the paper.

2.3.1 Basic Concepts

We largely follow the formulation of [51]. Our conservation problems will involve a (large) land region of interest that is divided into *land parcels* that are the smallest land units available for purchase. Each parcel contains some number of distinct *habitat patches*, which are the atomic units in the population dynamics model and can either be occupied or unoccupied by the species of interest. For example, in the Red-cockaded Woodpecker problem considered in our experiments, habitat patches correspond to particular trees that have been prepared by humans (or existing birds) to facilitate nesting. Each parcel p has a cost $c(p)$, which denotes the cost of purchasing the land and restoring or conserving all of its habitat patches so that they are suitable for the species to occupy.

A *conservation design* is a set of parcels that are intended to be purchased and conserved. Given a conservation design D, a *purchase schedule* π for D is a mapping from parcels in D to purchase times in $\{0, 1, \ldots, H, \infty\}$, where H is the time horizon of interest and purchasing a parcel at time $t = \infty$ means this parcel is not going to be purchased. Thus the scheduler may choose not to purchase some parcels even though they are part of the design so as to realize the best tradeoff between budget flexibility and population spread. Although the species population dynamics have a yearly time step in our model (described below), the allowed purchase times (i.e. decision epochs) may be less frequent depending on the specific problem. An *upfront schedule* is one that assigns all parcels to purchase time $t = 0$.

It is worth noting that the purchase times specified by a schedule are best viewed as purchase deadlines. That is, we interpret the schedule as constraining the purchases to occur before or at the specified times. This view is justified by the fact that in the setup below, purchasing a parcel at an earlier time than specified will never result in worse population spread.

2.3.1.1 Population Dynamics Model

We use the same stochastic dynamics model as [51], which is an instance of a popular *metapopulation model* from the ecology literature [26]. A patch a has two possible states at each time step, either unoccupied or occupied, and only conserved patches may be occupied. The population dynamics consists of two types of stochastic events.

Colonization events occur when a population from patch a colonizes an unoccupied patch b, which happens with probability p_{ab}. Extinction events occur when a patch a that is occupied at time t becomes unoccupied at time $t+1$, which happens with probability $1-p_{aa}$. All events are independent. The details of the probabilities used in our experiments are given in Section 2.5.

The single-step colonization probability p_{ab} in our experiments typically decays with the distance between patches a and b, which encodes spatio-temporal dynamics in which populations slowly spread from a source population when new habitat is made available. Thus, in long-term planning for population spread, it is often unnecessary to purchase parcels that are distant from a source population at time $t=0$, since the probability of the population spreading to such distant patches in the near future is negligible. By delaying such purchases until they become relevant to the design (i.e. the population has spread nearby), a conservation organization can use limited funds much more flexibly. However, it is non-trivial to decide how much to delay purchases so as not to harm the spread, since this decision depends very much on the spatio-temporal details of the population spread model. It is this decision that the optimization problem defined below is designed to make.

2.3.2 Stochastic Optimization Problem

Our problem statement will rely on two important concepts: 1) the *reward* of a schedule, and 2) the *flexibility* of a schedule. We first define these two concepts and then formulate the optimization problem in terms of them.

The reward of schedule π, denoted by $R(\pi)$, is a random variable that encodes the amount of population spread at time H, which is simply a count of the number of occupied patches at time H. It is easy to show in our model that the upfront schedule always achieves at least as much reward as any other schedule and thus maximizes the expected reward. Thus, we define the maximum expected reward as $R^* = E[R(\pi_{\text{upfront}})]$. Our optimization goal is to find a schedule π that almost achieves this optimal expected reward, —i.e. $E[R(\pi)] \geq (1-\varepsilon)R^*$ where ε is a positive real number and indicates the percentage tolerance of reward loss —but has maximum "purchase flexibility". We know that the upfront schedule achieves $(1-\varepsilon)R^*$, however, it requires commitment to all expenditures at the first time step and is thus the least flexible. Indeed, we now

formalize the notion of flexibility in terms of expenditures over time.

Given any schedule π we can define its corresponding *cost curve* C_π to be a function from purchase times to accumulated cost, so that $C_\pi(t)$ is equal to the total cost of parcels purchased under π from time 0 up to and including time t. This curve is non-decreasing and provides a view of a schedule's spending profile over the time horizon. In particular, if the profile of cost curve C_{π_1} is never above that of C_{π_2}, i.e. the total expenditures of π_1 never exceed those of π_2, then we can say that π_1 offers more flexibility in terms of budget management compared to π_2 and should be preferred if all else is equal.

Now we define a surrogate cost function over schedules as

$$\text{cost}_f(\pi) = \sum_p c(p) \cdot f(\pi(p))$$

which is parameterized by a function f from times in $\{0, \ldots, H, \infty\}$ to real numbers. We require $f(\infty) = 0$ for any f so that any parcel that is not purchased within the time horizon would not contribute to the surrogate cost. We can see that this surrogate cost function is simply a weighted sum of the parcel costs, where the weight is determined by f based on the parcel's purchase time. Although our algorithm can work with any real-valued function, we assume henceforth that f is strictly decreasing. There are two reasons for this. First, discounting future costs makes sense due to economic factors such as inflation. Second, intuitively, since f decreases with purchase times, minimizing with respect to cost_f would favor schedules that delay purchasing. In particular, if policy π_1 has a cost profile that is never greater than that of π_2, then π_1 will be assigned a lower surrogate cost when f is strictly decreasing.

If all parcels have positive costs, then the upfront schedule is the unique element that maximizes the surrogate cost, and the schedule that defers all purchases until time H gives the unique minimum as f is a strictly decreasing function. However, if we restrict to schedules that achieve at least reward $(1 - \varepsilon)R^*$, the latter schedule will be excluded and there may no longer be a unique minimum.

We can now specify the problem of *conservation design scheduling*, which is to find a schedule π^* from the set of all possible schedules such that:

$$\pi^* \in \arg\min_\pi \text{cost}_f(\pi) \text{ s.t. } E[R(\pi)] \geq (1 - \varepsilon)R^* \tag{2.1}$$

That is, out of all schedules that achieve reward $(1-\varepsilon)R^*$ we want to return one that is minimal in terms of its surrogate cost (i.e. it has maximal flexibility). Thus, ε controls the trade-off between flexibility and reward. In particular, using larger ε increases the set of feasible schedules and allows the potential for returning a more flexible schedule by sacrificing some reward.

Note that by varying the choice of f it may be possible to generate different solutions to Equation 2.1, each of which is *minimal* in the sense that no other feasible policy has a strictly lower cost curve. In our experiments, we use a simple discounted f given by $f(t) = \beta^t$ for a discount factor $\beta \in (0, 1)$.

In practice, it is likely that a conservation manager will not have a particular value of ε in mind at design time. Rather, ε is best viewed as a parameter that will be varied in order to observe the different flexibility-reward trade-offs that are possible. The final selection of a schedule would then be based on an assessment of those possibilities.

Finally, it is worth noting that for $\varepsilon = 0$ (no reward approximation), the upfront solution will be the only feasible solution under typical population spread models. Thus, using $\varepsilon > 0$ is necessary for achieving any additional flexibility. This is because requiring a policy to achieve expected reward *exactly* R^* (i.e., $\varepsilon = 0$) requires it to make purchases to accommodate very unlikely outcomes of the population spread model that contribute a tiny, but positive, amount to the expected reward. For example, consider an outcome where the population jumps from its initial location to a very distant location in the first year, and then undergoes no further spread. This has vanishingly small, but positive, probability. The upfront schedule will support this population spread, since the distant location is purchased at the first step. Thus, any schedule that does not purchase the distant parcel in the first step will suffer a tiny loss in reward compared to the upfront schedule, so it does not achieve $\varepsilon = 0$. However, such a purchase will tend to be useless for the vast majority of the probability mass.

2.3.3 Deterministic Optimization Problem

The above optimization problem is stochastic in the sense that its constraint is defined in terms of an expectation over a complicated population spread distribution. This greatly complicates the direct solution of this problem. As in prior work on stochastic optimization of upfront schedules [51], we address this complication by converting the

stochastic problem into an approximately equivalent deterministic optimization problem. This is done via the very common *Sample Average Approximation (SAA)* approach (see Shapiro, 2003 for a survey of some results). The key idea is to approximate a stochastic optimization problem using a collection of samples from the probability distribution, which are used to approximate expectations or probabilities via averages over samples.

In our problem formulation, each sample corresponds to a so-called *cascade scenario*, which is a particular realization of the population spread process over the time horizon. The main idea behind our application of SAA is to generate a set of such *cascade scenarios* from the probabilistic population spread model, and to approximate the expected reward of schedules as the average reward over the scenarios. The scenarios are combined into a single *scenario graph*, which is illustrated in Figure 2.1 and explained in detail in the remainder of this section.

More concretely, a cascade scenario is a layered graph, where layers correspond to time steps, with a vertex $v_{a,t}$ for each patch a and each time step t. For each pair of patches (a, b) and time step t, a coin is flipped with probability p_{ab} to determine if the directed edge $(v_{a,t}, v_{b,t+1})$ is present or not. If this edge is present and patch a is occupied at time t (through previous colonizations or non-extinctions), then patch b will be colonized and become occupied at time $t + 1$, as long as it is conserved. That is, the presence of edge $(v_{a,t}, v_{b,t+1})$ is interpreted as meaning that if a is occupied at time t and b is conserved at or before time $t+1$, then b will be occupied at time $t+1$ in the particular scenario. In this way, a cascade scenario graph encodes occupancy as reachability. In particular, assuming (for now) that all patches are conserved, then patch b is occupied at time t exactly when $v_{b,t}$ is reachable from a vertex $v_{a,0}$ corresponding to an initially occupied patch a.

To approximate the probabilistic spread model, we sample a set of N i.i.d. cascade scenarios $\{C_1, \ldots, C_N\}$, where we will denote the vertices in C_n by $\{v^n_{a,t}\}$. These scenarios are combined into a single *scenario graph*, which has an additional root vertex r with directed edges $(r, v^n_{a,0})$ to each vertex representing an initially occupied patch a. Figure 2.1 shows an example scenario graph that has three scenarios over a range of five time steps involving three patches a, b, and c, and two parcels, one containing a and b and the other containing just c. In this example, a is the only initially occupied patch and hence is connected at time step zero to the root node r across all three scenarios. Assuming that all parcels are conserved (i.e. purchased upfront), if a vertex is connected to the root

node r, then the corresponding patch is considered to be occupied at the corresponding time in the particular scenario. This is because we defined r so that it can only be connected to other vertices through initially occupied patches.

Scenario graphs will be used in our work to estimate the reward of schedules as follows. Given a scenario graph, a schedule π is said to purchase node $v^n_{a,t}$ and all of its incoming edges if patch a is purchased no later than time t, that is, $\pi(p) \leq t$ where a belongs to parcel p. Thus, purchasing a parcel p at time t can be viewed as purchasing all vertices in the scenario graph, along with their incoming edges that involve patches in p that occur at layer t or later. This reflects the fact that once a patch is purchased and conserved, it is considered to be conserved and hence eligible for occupancy for the remainder of the time horizon. In Figure 2.1, an example schedule is shown that purchases parcel p_1 (containing a and b) at time 0, and parcel p_2 (containing c) at time 3. The vertices that are purchased by this schedule are shown in the shaded region and their (purchased) incoming edges are shown in bold.

We can now define under what conditions a vertex in a scenario graph is considered to be occupied given a schedule. Vertex $v^n_{a,t}$ becomes occupied under π if there is a path through *purchased* edges from r to $v^n_{a,t}$. We define the variable $X^n_\pi(a,t)$ to be equal to 1 if $v^n_{a,t}$ is occupied under π and 0 otherwise. In Figure 2.1, we have shaded in red the set of vertices that are occupied under the example policy. As an example, note that in scenario 3, the vertex $v^3_{c,3}$ is not occupied since there is no path from r to it through purchased edges. This is despite the fact that there is a path in the graph from r, since that path involves some unpurchased edges. Note that the upfront schedule would purchase this node, since all vertices and edges would be considered to be purchased under that schedule.

The average reward of a schedule π relative to a scenario graph built from scenarios $\{C_1, \ldots, C_N\}$ is denoted as follows.

$$\hat{R}(\pi) = \frac{1}{N} \sum_{n=1}^{N} \sum_{a} X^n_\pi(a, H)$$

This is just the average across scenarios of the number of occupied patches at time H. In Figure 2.1 the average reward for the example schedule would be 2. A key property of scenario graphs is that as $N \to \infty$ we have that $\hat{R}(\pi)$ converges to $E[R(\pi)]$ for any fixed

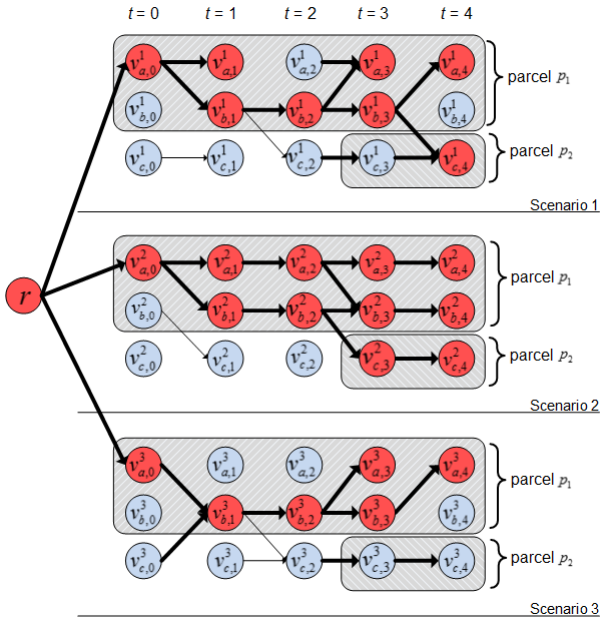

Figure 2.1: Example scenario graph ($N = 3$) for problem with parcels $p_1 = \{a, b\}$, $p_2 = \{c\}$. The schedule ($\pi(p_1) = 0, \pi(p_2) = 3$) is also illustrated, using shaded boxes to indicate purchased nodes and heavy line weights to indicate purchased edges. Vertices representing occupied patches under this schedule are colored red.

π. This implies that the set of schedules $\{\pi : \hat{R}(\pi) \geq (1-\varepsilon)\hat{R}(\pi_{\text{upfront}})\}$ converges to the set $\{\pi : R(\pi) \geq (1-\varepsilon)R^*\}$ as N grows, which is the set of policies that we wish to optimize flexibility over. Further, for any policy π, one can use standard probability concentration bounds (e.g. Chernoff bounds) to show that the event $|R(\pi)-\hat{R}(\pi)| \geq \epsilon$ has a probability mass that decreases exponentially fast as N grows. This suggests that only a relatively small number of scenarios are required to reliably obtain a tight approximation to the true expected reward of a policy. In practice, however, it is important to empirically validate that the approximation errors are reasonable for the number of scenarios used in the approximation.

The above motivates a deterministic SAA formulation of our original stochastic optimization problem (2.1) where flexibility is optimized subject to a constraint based on

the empirical reward $\hat{R}$. That is, our deterministic problem is to solve:

$$\pi^* \in \arg\min_{\pi} \text{cost}_f(\pi) \text{ s.t. } \hat{R}(\pi) \geq (1-\hat{\varepsilon})\hat{R}^* \tag{2.2}$$

where $\hat{R}^* = \hat{R}(\pi_{\text{upfront}})$.

2.4 Approach

Recall that for the stochastic optimization problem (2.1) setting $\varepsilon = 0$ resulted in an optimization problem that would typically have only the upfront schedule as a feasible solution. Rather, here, for the approximate SAA formulation, there will typically be non-upfront solutions that are feasible, even when using $\hat{\varepsilon} = 0$. This is because even for large (but practical) values of N, the set of scenarios used for the approximation will not tend to include highly unlikely scenarios, which need to be accounted for in the stochastic solution when using $\varepsilon = 0$. This observation motivates our solution approach for (2.2). In particular, in Section 2.4.1, we first consider the problem when $\hat{\varepsilon} = 0$, which turns out to be a new variant of the classic Steiner tree problem. We then derive an incremental primal-dual algorithm for the problem (Section 2.4.2) that can be used to approximately solve the $\hat{\varepsilon} = 0$ case or the $\hat{\varepsilon} > 0$ case through early stopping. Our experiments will show that this approach is able to provide significant flexibility with little loss in reward, with the flexibility-reward trade-off being controlled by $\hat{\varepsilon} > 0$.

2.4.1 Set-Weighted Directed Steiner Graph Formulation

As motivated above, here we focus on optimization problem (2.2) for the case when $\hat{\varepsilon} = 0$. That is, we must optimize flexibility subject to the constraint that we obtain the optimal empirical reward as measured by $\hat{R}$. In this section, we show how to formulate this problem as a novel variant of the Steiner tree problem.

2.4.1.1 Set-Weighted Directed Steiner Graph

From (2.2), we arrive at our final optimization problem for $\hat{\varepsilon} = 0$:

$$\pi^* \in \arg\min_{\pi} \text{cost}_f(\pi) \text{ s.t. } \hat{R}(\pi) = \hat{R}^*. \tag{2.3}$$

We can view this problem as a type of Steiner tree problem on the scenario graph. In particular, we say that any vertex at time $t = H$ is a *terminal vertex* if it is reachable from the root r, which is the set of nodes with $X^n_{\pi_{\text{upfront}}}(a, H) = 1$ and hence contribute to the upfront reward $\hat{R}^*$. The only way for π to satisfy the constraint $\hat{R}(\pi) = \hat{R}^*$ is to purchase a set of edges in the scenario graph that connect all of those target nodes to r. Thus, the constraint in Equation 2.3 corresponds to purchasing edges such that r has a path to each terminal, as in the Steiner tree problem.

As an example, consider again the scenario graph in Figure 2.1. The terminal nodes in this example are all nodes at layer $t = 4$ except for $v^1_{b,4}$ and $v^3_{b,4}$, which are the only two nodes that are not connected to r by a directed path. A schedule that satisfies the constraint $\hat{R}(\pi) = \hat{R}^*$ must purchase edges such that all of these terminals are reachable from r. Note that for the example schedule of Figure 2.1, the set of purchased edges does not satisfy this constraint since there is no path of purchased edges to the terminal vertex $v^3_{c,4}$.

While our problem is very similar to the traditional Steiner tree problem, there is a significant difference. In the traditional problem, each edge is associated with a distinct weight and can be purchased individually, with the goal of connecting all terminals using a set of edges of minimum total weight (which always forms a tree). Rather, our situation is more complicated because we purchase parcels, which correspond to subsets of edges in the scenario graph. In particular, purchasing a parcel p at time t, which incurs cost $c(p)f(t)$ in Equation (2.3), corresponds to purchasing an edge set $E_{p,t}$ with cost $c(p)f(t)$ that contains all the edges $(u, v^n_{a,t'})$ that come from any vertex u and arrive at any vertex $v^n_{a,t'}$ with $a \in p$, $t' \geq t$ and $n \in \{1, \ldots, N\}$. Note that under this cost model, the total cost of edge sets purchased by a schedule π exactly equals our surrogate objective $\text{cost}_f(\pi)$.

From the above we see that our problem is an instance of a problem that we will call the *Set-weighted Directed Steiner Graph (SW-DSG)* problem, a novel variant of the Steiner tree problem, the goal of which is to select a set of vertices with minimal total cost in order to connect all the terminal vertices to the root. For the remainder of the

paper we will discuss this problem in its general form to simplify notation. The input for SW-DSG is a directed graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ with a single root vertex r, a set of terminal vertices $T \subseteq \mathbb{V}$, a set of M edge sets $\mathcal{E} = \{E_1, \ldots, E_M\}$ where each $E_s \subseteq \mathbb{E}$, and a non-negative cost c_s for each E_s. In particular, our conservation problem has edge sets $\mathcal{E} = \{E_{p,t}\}$ with $E_{p,t} = \{(u, v^n_{a,t'}) : (u, v^n_{a,t'}) \in \mathbb{E}, a \in p, t' \geq t, n \in \{1, \ldots, N\}\}$ and cost $c_{p,t} = c(p)f(t)$. A subset of $\mathcal{E}$ forms a Steiner graph if the union of the edges connect r to all vertices in T. The desired output is a minimum cost subset of $\mathcal{E}$ that forms a Steiner graph. Note that the optimal Steiner graph need not be a tree in SW-DSG, unlike in the traditional Steiner tree problem.

It is clear that SW-DSG is more general than the original deterministic optimization problem since the latter has a specific edge set structure. For instance, $E_{p,t_1} \subseteq E_{p,t_2}$ if $t_1 > t_2$. However, this structure does not make it an easier problem than SW-DSG. In the following sections, we prove that both problems are NP-complete and our primal-dual algorithm is the same for either setting. The special structure does not lead to any algorithmic advantages in deriving a primal-dual algorithm. Therefore, we mainly discuss the problem in the form of SW-DSG to simplify notation.

While the SW-DSG problem was motivated by our particular conservation application, it is relevant to other problems that have Steiner style objectives, but where the "edge resources" are best considered as groups. For example, Steiner trees are often used for the design of communication networks where edges correspond to existing or potentially new communication links. For situations where those links must be purchased as coherent sets (e.g. the communication infrastructure of different companies/organizations), our SW-DSG problem would be the appropriate formulation.

2.4.1.2 Computational Complexity

To our knowledge, the SW-DSG generalization of the Steiner tree problem has not been previously studied and hence we now consider its computational complexity.

The SW-DSG problem is a generalization of the traditional directed Steiner tree (DST) problem, which is known to be NP-complete [32]. Further, under standard complexity assumptions, DST is hard to approximate by a factor better than $\log(|T|)$ [11]. Note that these results hold even for acyclic directed graphs. There are a number of effective heuristic algorithms for DST [18], with many of the most successful relying on

shortest path computations as a subroutine. While shortest paths can be computed in edge weighted graphs efficiently, this turns out to not be the case for our set-weighted problem. In particular, note that the shortest path problem is a special case of DST (or SW-DSG) where there is a single terminal vertex. This problem turns out to be NP-Hard for SW-DSG, even when restricted to acyclic graphs and the special edge set structure shown in our original deterministic optimization problem, which is the case for the scenario graphs from our conservation problem.

Theorem 1. *The SW-DSG problem is NP-hard even when restricted to acyclic graphs with a single terminal vertex and the edge set structure in scenario graph.*

Proof. We prove the hardness by reducing the weighted set cover problem to the subclass of SW-DSG problems restricted to a scenario graph with one scenario and exactly one terminal. Note that here we consider the decision version of the SW-DSG problem, which asks if there is a feasible Steiner graph whose cost is less than a specified threshold C^*. An instance of the weighted set cover problem specifies a ground set of elements $S = \{e_1, \ldots, e_n\}$, a set $\mathcal{S} = \{S_1, \ldots, S_m\}$ of m subsets $S_j \subseteq S$, a cost C_j for each subset, and a cost bound C^*. The problem asks whether there is a collection $\mathcal{S}' \subseteq \mathcal{S}$ with total cost no more than C^* such that $\bigcup_{S_j \in \mathcal{S}'} = S$.

Given a set cover instance, we first describe how to construct a scenario graph as illustrated in Figure 2.2 and later describe the corresponding SW-DSG instance. The graph contains $2n$ layers, which alternate between *set layers* and *element layers* starting with a set layer (n layers of each, hence $2n$ layers). Each layer has m vertices labeled $S_1, \ldots, S_m$ to represent the sets in $\mathcal{S}$ and n vertices labeled $e_1, \ldots, e_n$ to represent the elements in S. In addition we include a root vertex r. Each vertex can also be seen as a parcel with a single patch. The edges in the graph only go from one layer to the immediate next layer as follows. The root vertex has an edge going to each S_j in the first set layer. For the ith element layer (i.e. layer $2i$ in the graph), we include an edge from a vertex with label S_j in the previous layer to a vertex with label e_i in the current layer whenever $e_i \in S_j$. Finally, vertex e_i at the ith element layer has an edge from it to each S_j in the next layer.

The corresponding SW-DSG (conservation problem) instance on this graph is specified as follows. The root node is r and the single terminal vertex is e_n at the final element layer. The edge sets are specified as follows, which is the same as the setting

in a scenario graph. There are edge sets $E_{j,t}$ for each S_j at time t. In particular, $E_{j,t}$ contains every incoming edge that is from any vertex and to any vertex labeled as S_j at layers $t' \geq t$. We let the strictly decreasing $f(t)$ be sufficiently close to 1 for all t's. The cost of each $E_{j,t}$ is equal to $C_j \times f(t)$ where C_j is the cost of S_j in the original set cover problem. In other words, the cost of $E_{j,t}$ is almost C_j. Similarly, there are edge sets $E_{i,t}$ for each e_i at time t. We set their costs as 0. The cost threshold for the SW-DSG problem is equal to the threshold C^* of the set cover problem.

To see that this reduction is correct, consider the case where the resulting SW-DSG instance has a feasible solution. The solution provides a path from r to e_n through purchased edge sets that have total cost at most C^*.

Since the edge set $E_{i,t}$ for each e_i has zero cost, the edge set cost is the result of purchasing edge sets $E_{j,t}$. By the construction the path must go through a sequence of alternating element nodes and set nodes. In particular, the path must traverse each element node e_i for $i = 1, \ldots, n$. The only way for this to happen is to purchase for each of those e_i at least one edge leading to e_i from one of the immediately preceding S_j at layer $t \leq 2i$, which is only possible when $e_i \in S_j$. This can only happen by purchasing the corresponding edge set $E_{j,t}$, corresponding to S_j, which has a cost of (almost) C_j. From this we see that the collection of S_j corresponding to purchased edge sets must cover all of the elements and that their total cost is no more than C^*. Thus, the collection of sets is a solution to the set cover problem.

Conversely consider an instance of the set cover problem with a feasible solution. It is easy to verify that a feasible solution to the corresponding SW-DSG problem is to purchase edge sets $E_{j,1}$ corresponding to any S_j in the set cover solution. Combining the above we see that there is a feasible solution to the SW-DSG instance if and only if there is a feasible solution to the set cover instance.

□

The above result proves that the shortest (or least cost) path problem is also NP-hard for SW-DSG, i.e. the problem of finding a least cost path when edges are purchased as sets. Thus, it is difficult to extend prior shortest-path-based heuristics for the Steiner tree problem to our problem. Given that SW-DSG is in NP, it is NP-complete. This motivates our derivation of an efficient heuristic solution approach in the next section, which computes both a feasible solution along with a bound on the cost of the optimal

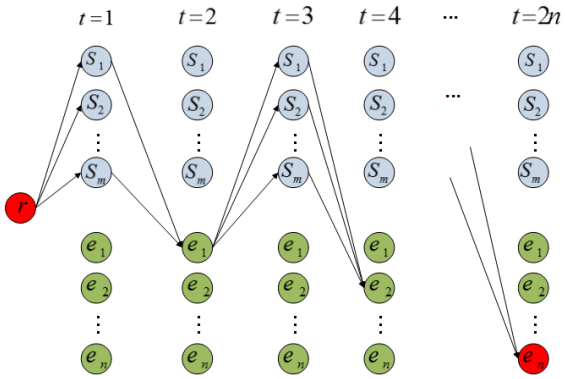

Figure 2.2: Description of the reduction from set cover to SW-DSG with a single terminal vertex and a scenario graph.

solution. Importantly this bound provides a sense of how good the computed solution is compared to the optimal.

2.4.2 Primal-Dual Algorithm

A potential solution approach to the SW-DSG problem is to encode it as a Mixed Integer Program (MIP), which is straightforward, and then to use an off-the-shelf MIP optimizer. While this approach produced non-trivial results for the upfront conservation problem [51], as our experiments will demonstrate, it does not scale well for our problem. A related approach could be to consider a rounding procedure for the MIP's LP-relaxation. While solving the LP-relaxation is easier than solving the MIP, our experiments show that the scalability of LP solvers is also poor for the problem sizes of interest to us. Instead, we exploit the MIP encoding in another way, by following the primal-dual schema [55] to derive a scalable algorithm that performs near optimally in our experiments. Our work can be considered as a non-trivial generalization of previous work [57], where the primal-dual schema was applied to DST. Moreover, an early-stopping version of our primal-dual algorithm provides a way to trade-off the schedule flexibility and reward ($\hat{\varepsilon} > 0$). Note that the primal-dual algorithms for SW-DSG and our original deterministic conservation problem only differ in the notations. In other words, the edge set structure in conservation problem does not offer further improvements for the algorithm. Thus in

the following, we only present the approach for SW-DSG.

2.4.2.1 Primal-Dual Algorithm for SW-DSG

To apply the primal-dual schema, we start by giving a primal MIP for the SW-DSG problem along with the dual of its LP-relaxation in Figure 2.3. The primal MIP includes a binary variable $y(E_s)$ for each edge set in $\mathcal{E}$, which indicates whether E_s was purchased ($y(E_s) = 1$) or not ($y(E_s) = 0$). The objective of the primal is then simply the sum of these variables weighted by the costs of the corresponding edge sets. The Steiner graph constraint, requiring that all terminals be connected to the root node by purchased edges, is encoded using a standard network-flow encoding (lines 2.2–2.4) involving flow variables $x^k_{i,j}$. The flow variable $x^k_{i,j}$ encodes the flow on edge (i, j) destined for terminal k. The flow balance constraints (line 2.2) guarantee that one unit of flow is carried on a path from the root node r to terminal k.

The LP-relaxation of the primal simply replaces the integer constraints on the $y(E_s)$ variables with a positivity constraint. The dual of this relaxed problem (lines 2.6–2.9) includes dual variables u^k_i and $w^k_{i,j}$ corresponding to the primal flow constraints. Note that the constraint that one unit of flow leaves the root is implied by the other flow constraints. By omitting this constraint, one could simplify the dual by eliminating the u^k_r variables (or, equivalently, set $u^k_r = 0$ for all $k \in T$).

Given the primal and dual formulations of our problem, we can now apply the primal-dual schema for designing optimization algorithms. In particular, our primal-dual algorithm is iterative where each iteration increases the value of the dual objective and purchases a single edge set E_s, which corresponds to setting the primal variable $y(E_s) = 1$. The iteration stops when the purchased edges form a Steiner graph (i.e. the primal becomes feasible). The value of the dual objective at the end of the iteration serves as a lower bound on the optimal primal objective, which provides a worst-case indication of how far from optimal the returned solution is. At a high level, this algorithm is a simple greedy heuristic that continuously purchases the most beneficial edge set in order to build paths for an unconnected terminal. The primal-dual schema provides a principled way of incrementally computing this heuristic and at the same time computing a lower bound.

Algorithm 1 gives pseudo-code for the algorithm. The main data structure is an

Algorithm 1 Primal-Dual Algorithm for SW-DSG.

{**Inputs:** Graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, edge sets $\mathcal{E} = \{E_1, \ldots, E_M\}$, costs $\{c_1, \cdots, c_M\}$, terminals $T \subseteq \mathbb{V}$}
Initialize:
$u_i^k = 0$, for each $k \in T, i \in \mathbb{V}$; $w_{i,j}^k = 0$, for each $(i,j) \in \mathbb{E}$, $k \in T$
$\mathbb{G}' = (\mathbb{V}, \mathbb{A})$ with $\mathbb{A} = \emptyset$
lowerBound $= 0$, solution $= \emptyset$
while $\mathbb{G}'$ is not a Steiner graph **do**
  Let k be random vertex in T not connected to r in $\mathbb{G}'$
  $S = \{s \mid E_s \cap \text{Cut}(k) \neq \emptyset, s \notin \text{solution}\}$
  $s^* = \arg\min_{s \in S} \Delta(s,k)$
  where $\Delta(s,k) = \left(c_s - \sum_{k' \in T,(m,n) \in E_s} w_{m,n}^{k'}\right) / |E_s \cap \text{Cut}(k)|$
  $u_j^k = u_j^k + \Delta(s^*,k)$, for each $j \in C(k)$
  $w_{i,j}^k = w_{i,j}^k + \Delta(s^*,k)$, for each $(i,j) \in \text{Cut}(k)$
  $\mathbb{A} = \mathbb{A} \cup E_{s^*}$
  lowerBound = lowerBound $+ \Delta(s^*,k)$
  solution = solution $\cup \{s^*\}$
end while
Pruning: solution = solution $- \{s \mid \exists s' \in \text{solution}, E_s \subset E_{s'}\}$

auxiliary graph $\mathbb{G}' = (\mathbb{V}, \mathbb{A})$ with the same vertices as the input graph $\mathbb{G}$. The auxiliary graph edge set $\mathbb{A}$ is initially empty and then each iteration adds the newly purchased edges $E_s \in \mathcal{E}$. The algorithm terminates when the edges in $\mathbb{A}$ form a Steiner graph. The edge sets used to construct this graph are then returned as the solution, following a pruning step that removes obviously redundant edge sets, which the algorithm can sometimes include during the iteration process.

In order to describe the algorithm in detail, we first introduce some terminology. Given a current auxiliary graph $\mathbb{A}$, we let $C(k)$ denote the set of all vertices that have directed paths to terminal node k via only edges in $\mathbb{A}$. Note that we consider k to be included in $C(k)$. Also, we define the *cut set* of a terminal node k, denoted by $\text{Cut}(k)$, to be the set of all edges (i,j) such that $j \in C(k)$ and $i \notin C(k)$. Intuitively, if k is not already reachable from the root, we know that at least one edge in $\text{Cut}(k)$ must be added to $\mathbb{A}$ in order to arrive at a Steiner graph.

The algorithm first initializes all dual variables to zeros and the auxiliary graph $\mathbb{A}$ to include all vertices and no edges. Each iteration then proceeds by first randomly selecting

a terminal vertex k that is not connected to r in the auxiliary graph. At an intuitive level, the algorithm will then select an edge set E_s that contains a cutset edge of k according to a heuristic $\Delta(s,k)$ that is derived by applying the primal-dual schema. More concretely, the aim of each iteration is to raise the dual objective value by increasing the value of u_k^k while maintaining feasibility. Increasing u_k^k by itself will violate constraints of type (8) in the dual and lines 5 through 8 of the algorithm maintain feasibility by selecting an edge set E_{s^*} among those that intersects the cut set of k and then raising all variables corresponding to vertices in $C(k)$ and edges in $\mathrm{Cut}(k)$ by a value $\Delta(s^*,k)$ (including u_k^k). This is done in a way that causes the dual constraint of type (7) corresponding to edge set E_{s^*} to become tight. Since this constraint corresponds to primal variable $y(E_{s^*})$, the algorithm effectively sets $y(E_{s^*})=1$, indicating a purchase, by adding the edges in E_{s^*} to $\mathbb{A}$. The dual objective value at termination is the sum across iterations of $\Delta(s^*,k)$ and is returned as the lower bound.

The key property of our algorithm is that each iteration increases the dual objective, while also maintaining feasibility of the dual. This guarantees that at each iteration the dual objective value corresponds to a true lower bound on the optimal value of the primal.

Theorem 2. *Each iteration of the primal-dual algorithm produces a feasible dual solution with increased objective.*

Proof. As the base case, the initialization assigns all dual variables to be zeros, which is a feasible solution. Now suppose that iteration $q-1$ starts with a feasible solution $\{u_i^l, w_{i,j}^l\}$, which satisfies the dual constraints of type (7) and (8). Now if the algorithm terminates, we get a feasible solution. Otherwise let k be the terminal vertex selected. For all variables $\{u_i^l, w_{i,j}^l\}$ with $l \neq k$ the values are not changed, so (8) is satisfied. For the remaining variables with $l=k$, there are three cases. *Case 1:* For $j \notin C(k)$, the variables u_j^k and $w_{i,j}^k$ are unchanged, so they cannot contribute to a violation of (7) or (8). *Case 2:* For any edge (i,j) with both $j,i \in C(k)$, we increase both u_j^k and u_i^k by $\Delta(s^*,k)$ and continue to satisfy the corresponding constraint of (8). *Case 3:* For any cut set edge $(i,j) \in \mathrm{Cut}(k)$, we increase u_j^k and $w_{i,j}^k$ by $\Delta(s^*,k)$ so that (8) remains satisfied. Since the $w_{i,j}^k$ for edges in the cut set are increased, we must ensure that constraints of type (7) do not become violated. The choice of $\Delta(s^*,k)$ made by the algorithm can be verified to never violate any of those constraints and makes at least one of them tight. □

After the main portion of the algorithm terminates, a pruning step is conducted to remove any edge set that is a subset of some other edge set in the solution, which decreases the total cost while maintaining feasibility. In particular, in the context of our conservation scheduling problem, the pruning step ensures that each parcel is purchased no more than once in the final solution. There are other more aggressive and computationally expensive pruning techniques that could also be used. For example, one could consider removing each one of the selected edge sets from the final solution and then test for feasibility. If the solution is still feasible, then the edge set can be eliminated. We did not find this more aggressive style of pruning to be necessary in our experiments.

Implementation and Running Time. Note that our pseudo-code stores and updates values for the u_j^k and $w_{i,j}^k$ dual variables. Then a naive implementation of the above algorithm would result in $O(|\mathbb{E}||T|)$ runtime for initialization as well as the computation per iteration, where $|\mathbb{E}|$ is the number of edges in the graph and $|T|$ is the number of terminals. This could be too much for SW-DSG problems with a large network such as the one in our conservation application. However, the algorithm is described in this way only for presentation purposes. It turns out that for the purposes of running the algorithm, it can be implemented significantly more efficiently. In particular, we only need to store and update the sum of corresponding $w_{i,j}^k$ values for each edge set (i.e. the sum term that appears inside of the definition of $\Delta(s,k)$ on line 6), and maintain the current objective value (stored as lowerBound in the pseudo-code), which is updated on line 10. Therefore, before the iterations only those $M+1$ variables need to be initialized, where M is the number of edge sets and is much smaller than the size of the network. In our implementation the dominant computation per iteration is the computation of the cut set for the selected terminal k. We find the cut set by a backward traversal from terminal k toward the root. The time for this computation is acceptable when terminals are only connected to relatively small parts of the overall graph. This is the case in our conservation problem, where terminals are only connected to nodes in the same cascade and among those only ones that are spatially close enough to be reached. In other applications where terminals are possibly connected to a large portion of the graph, it may be preferable to incrementally maintain a cut set for every terminal at each iteration to reduce the computation. Then the memory needed is $O(C|T|)$ where C is the maximum size of a cut set and presumably $C \ll |\mathbb{E}|$. After getting the cut set, the algorithm takes $O(MC)$ time to identify the best edge set and update the solution.

2.4.2.2 Early-Stopping for Fractional Connection

In our primal-dual algorithm, the computation continues until all of the terminals in the scenario graph are connected by paths from the root. In the context of our conservation problem this corresponds to having no reward approximation loss ($\hat{\varepsilon} = 0$). Here we modify the above primal-dual algorithm to allow for reward approximation loss where $\hat{\varepsilon} > 0$. This case corresponds to only modifying the SW-DSG feasibility constraint to only require a fraction $1 - \hat{\varepsilon}$ of the terminals to be connected, leading to a natural way of exploring the trade-off between reward and flexibility.

Given the incremental, greedy nature of our primal-dual algorithm, which adds one edge set each iteration, a natural choice for this fractional connection problem is to stop the algorithm whenever at least a fraction $1 - \hat{\varepsilon}$ of the terminals are connected.

While this basic early-stopping approach will lead to some improvement in the cost of the returned solution, compared to $\hat{\varepsilon} = 0$, the savings are often quite minimal. This is due to the fact that the primal-dual algorithm grows paths from the terminal nodes to the root and is unaware of the early-stopping condition. As a result, some of the paths that were being grown are never actually connected to the root at the point that the algorithm is stopped. These unconnected paths can be considered to be a waste of resources with respect to meeting the fractional connection constraint. Thus, to make this early-stopping algorithm viable, it is necessary to perform pruning on the early-stopping result. Our algorithm for fractional coverage then has two stages: 1) Generation, where early-stopping is used to generate an initial solution that meets the fractional connection constraint, and 2) Pruning, where the solution produced in stage 1 is pruned while maintaining the fractional connection constraint.

For the pruning stage we use a simple but effective greedy strategy. The idea is to iterate through the purchased parcels in the schedule returned by the early-stopping stage and to delay the purchase of each parcel as long as possible while ensuring that the number of connected terminal nodes is almost always within the required fractional connection tolerance.

We have found that for our conservation application, where the SW-DSG problem corresponds to a set of cascades, it is beneficial to prune using an independently generated and larger set of scenarios than those used to create the initial solution. This is analogous to using validation data to tune algorithm parameters in Machine Learning prediction

problems, and it is beneficial for the same reasons. We found that pruning based on the original set of scenarios was often overly aggressive and hurt empirical performance due to over-fitting of the SAA scenarios. Since we can easily generate independent scenarios to estimate the true expected reward of the pruned policies, it is better to prune based on that criterion instead. Also, since the computational complexity of evaluating the reward of pruned policies is low compared with the SAA optimization, we can afford to use a larger set of scenarios. In particular, in our experiments we formed the initial schedules based on a set of 10 cascade scenarios and conducted the pruning step with respect to 40 cascade scenarios.

This approach for pruning can also be viewed as directly enforcing a threshold on the (independently estimated) expected reward from the original stochastic problem (Equation 2.1) instead of enforcing a threshold on the objective value of the SAA problem (Equation 2.2). Since we can't calculate the correct threshold value (the RHS of Equation 2.1) without knowing the true optimum R^* of the stochastic problem, we use the SAA optimum $\hat{R}^*$ in its place. The SAA optimum is a stochastic upper bound to R^*, so this is generally a conservative approach for enforcing Equation 2.1.

2.5 Experiments

In this section, we first evaluate our primal-dual algorithm by applying it to a real, full-scale conservation problem. Next, to verify the robustness of our approach to other problems, we present results using synthetic conservation data from a problem generator used in several recent studies. We focus the first two parts of the experimentation on the case of $\hat{\varepsilon} = 0$, which we will see provides substantial gains in flexibility. In order to explore the trade-off between flexibility and reward (population spread), at the end of this section, we evaluate the early-stopping approach for $\hat{\varepsilon} > 0$.

2.5.1 Evaluation of Primal-Dual Algorithm on Real Conservation Map

The real map we use is the same dataset as in prior work by Sheldon et al. [51] on computing upfront conservation designs. The data is derived from a conservation problem involving the Red-cockaded Woodpecker (RCW) in a large land region of the south-

eastern United States that was of interest to The Conservation Fund. The region was divided into 443 non-overlapping parcels (each with an area of at least 125 acres) and 2500 patches serving as potential habitat sites. Parcel costs were based on land prices and some land parcels were already conserved and thus had cost zero. We use the same population spread model as Sheldon et al. [51], which was based on individual-based models of the RCW. Since our approach requires a conservation design as input, we use the design computed by Sheldon et al. [51] using a total budget constraint of $320M. The map of the area is shown in the left cell of Figure 2.7, with parcels making up the design shaded green and free parcels (with cost 0) shaded grey; red '+' marks indicate initially occupied patches. Our method also requires specifying a strictly decreasing function for defining the surrogate cost function, for which we use $f(t) = \beta^t$ with $\beta = 0.96$. We found that the results are not very sensitive to the value of β.

2.5.1.1 Comparing to Optimal Solutions

Here we compare the solutions of our primal-dual algorithm to optimal solutions found using the CPLEX solver applied to a MIP encoding of the SW-DSG problem. The MIP encodings become very large as the horizon and number of scenarios increase. In particular, there are $443 \cdot H + 2500 \cdot H \cdot N$ variables and the number of constraints grows with the number of edges, in the cascade network, which becomes impractical as N and H grow. Since the optimal solver can't scale to large versions of the problem, we consider problems involving cascade networks with just 2 scenarios and horizons ranging from just 15 to 40 years. We also use CPLEX to compute solutions to the LP-relaxation of the MIP. The objective value returned for the LP provides an alternative approach to computing a lower bound on the optimal solution and thus is interesting to compare to our lower bound in terms of tightness and runtime. Since our primal-dual algorithm is stochastic due to the random selection of terminal nodes, we report averages over 20 runs, noting that the standard deviations are negligible.

The first two data columns of Table 2.1 show the (surrogate) cost of the solutions found by CPLEX solving MIP and our algorithm (PD) for increasing horizons, where larger horizons correspond to larger problems. When a method fails to return a solution due to memory constraints no value is shown in the table. We see that for horizons where MIP is able to yield solutions by CPLEX, our algorithm produces solutions that have

Table 2.1: Comparison of Primal-Dual (PD) with MIP and LP

	Cost (M$)		Lower Bound		Run Time (s)		
	MIP	PD	LP	PD	MIP	LP	PD
H = 15	126.8	126.2[2]	122.2	84.9	5.5	6.1[3]	0.9
H = 20	123.6	125.7	117.7	71.9	8.2	7.6	2.5
H = 25	117.6	121.4	104.7	61.5	28	10	9.0
H = 30	130.4	134.0	117.3	56.9	5126	15	11
H = 35		131.3	109.9	64.1		18	25
H = 40		127.5		59.7			45

very similar costs (here lower cost is better). We also see that the MIP solver runs out of memory and is unable to return solutions for the 2 largest problem instances, which are already scaled down versions of the problem (small number of cascades and horizon).

The next two columns of Table 2.1 provide results for the lower bound computed by CPLEX solving the LP and by the PD algorithm. We see that the lower bound produced by the LP is significantly tighter than the bound produced by our algorithm. However, the LP cannot be solved by CPLEX for the largest problem, while our approach still yields a lower bound. Overall, though our lower bound is not as good as the LP (when it can be computed), it is generally within a factor of two of the optimal solution, which provides a non-trivial assurance about the quality of the returned solution for very large problems.

The final three columns of Table 2.1 present the time used by the approaches for each problem, where blank cells indicate that the method ran out of memory. Our algorithm is significantly faster than the MIP approach, which fails for the two largest problems, and is comparable with the LP approach, which only provides a lower bound and fails for the largest problem. This latter result indicates that a solution based on LP-rounding would face difficulty, since even solving the LP for these large problems (40 time steps with 2500 patches each) is computationally demanding. An advantage of the primal-dual

[2]Here the PD cost is less than the "optimal" MIP cost returned by CPLEX. After investigating, we found that CPLEX correctly evaluates the solution it returns, but thinks that the solution is optimal when it is not. It appears that this issue is due to the small error tolerance allowed by the CPLEX solver.

[3]Here MIP takes less time than LP. We think this is possibly because CPLEX uses different algorithms for LP and MIP. Especially, MIP is solved by a branch-and-bound algorithm that uses modern features like cutting planes and heuristics, making CPLEX a powerful MIP solver.

algorithm is that it avoids encoding the LP and works directly with the graph.

2.5.1.2 Number of Cascades in the Scenario Graph

According to SAA, the optimal solution over a finite set of cascade scenarios will converge to true optimum with more scenarios. Previously only two cascades are used due to the poor scalability of CPLEX. Now we study the number of cascade scenarios we should use to ensure a good solution. Recall that as N increases, the original stochastic problem is approximated more accurately. Yet a larger N corresponds to more computation. More importantly, here with $\epsilon = 0$, a larger N leaves the schedule less space for flexibility. In the extreme case of $N \to \infty$, the only possible schedule is the upfront schedule that has minimal flexibility. To find a good value of N in practice, we study the primal-dual solutions with different number of cascade scenarios by validating the reward $R(\pi)$ that the each primal-dual schedule can achieve. Given that the population spread is stochastic, we compute the reward $\hat{R}(\pi)$ by running 20 simulations of the stochastic population spread model. Each simulation provides a reward value (number of occupied patches at the horizon) and we average the results. We do this for the schedules produced by our primal-dual algorithm and for the upfront schedule. Recall that the intention is to nearly match the reward of the upfront schedule. Figure 2.4 presents the results when the time horizon is $H = 20$. We observe that the primal-dual schedule achieves more and more reward when N increases, and the reward is converging towards the expected reward of the upfront schedule. We also see that 10 cascades is quite close to get the best performance and the rate of improvement is slowing down. Thus, the remainder of our experiments use 10 cascades for the SAA.

2.5.1.3 Quality of Conservation Schedules

We now evaluate our algorithm on problems of more realistic sizes. Here, we consider problems based on 10 cascades and horizons ranging from 20 to 100 years, which are well beyond the range approachable for the MIP and LP. The solution times for our algorithm ranged from 15 seconds for $H = 20$ to 29 minutes for $H = 100$.

First, we evaluate the average accumulated reward of the schedule returned by our method for each horizon in Figure 2.5. The average reward of the upfront schedule ranged

from 332 for $H = 20$ to 615 at $H = 100$ and for all time horizons the primal-dual solution attained average reward at least 95.3% of optimal, with negligible error bars about the averages. The small gap indicates that for 10 scenarios the SAA approximation is quite good—the gap could be further reduced by increasing the number of scenarios.

Of course, we must also consider the cost curves corresponding to the schedules, since that is what affords the flexibility criterion of our problem. Figure 2.6 presents the cost curves for our schedules. Note that as defined in Section 2.3.2, a cost curve shows the (non-discounted) accumulated cost of a schedule over time. Then the cost curve for a schedule produced for horizon H will only increase until time H and then remain flat, reflecting that no purchases are made after that time. We see that for all horizons the cost curves show a fairly gradual increase in cost expenditures over time, indicating that the schedules are indeed providing a significant amount of flexibility regarding purchase times, particularly when compared to the upfront schedule, the cost curve of which is the flat black line in Figure 2.6 since all the parcels are purchased at time 0. In experiments not shown, we found that the cost curves vary by a small amount for different values of β, but the same general trend is present. Interestingly, in all curves there is a sudden jump in cost at around 20 years. To understand this in Figure 2.7 we show both the parcel purchases made by our schedule and the population spread on the map over the 100 year horizon. We see that at t = 20 the sharp increase in cost is due to the purchase of some relatively expensive and vast parcels in the southern part of the design. Looking at the population spread dynamics, it is apparent that those parcels are a critical gateway for ensuring reliable spread to the southwestern part of the design in later years. Delaying the purchase any longer significantly increases the probability that such spread does not occur, which our approach discovers.

Another interesting observation can be seen by comparing the expected population spread under the PD computed schedule and the expected population spread of the upfront schedule (Figure 2.7). The most striking difference between the spreads is seen at time steps $t = 20, 60, 80$ in the northeastern part of the map. For the upfront schedule the entire northeastern part is occupied in large part, while for the PD schedule there is a "hole" in the northeastern part near where the initial bird populations are located. Note, however, that this hole is finally occupied by the horizon of the problem ($t = 100$). At that time, the spread in the upfront and PD schedules are visually very similar, which agrees with the fact that their measured rewards were also similar. The reason for the

difference in population spread is that the PD schedule delays the purchase of some of the northeastern parts of the map near the initial bird population until about 20 years before the time horizon ends. From the population spread process of both schedules, we found that these parcels are very closely connected and hence can become occupied in a fairly short time if a bird population is nearby. This is apparent for the upfront schedule, where those areas are already occupied by $t = 20$. Thus, the purchase of such parcels can be delayed as long as there is enough time left for the population to spread over these landscapes. Therefore, the purchasing can be delayed not only for parcels far away from current population, but also for parcels that can be covered quickly and reliably. Note that such flexibility is mainly due to our definition of the reward function, which only takes the population at time H into account. If we count the population at every time step, presumably a good schedule would purchase the "hole" area very soon for more population.

2.5.2 Evaluation of Primal-Dual on Synthetic Maps

To evaluate the primal-dual algorithm more thoroughly, we randomly selected 10 synthetic maps generated and used in prior work [1]. All the maps consist of the same region of 146 non-overlapping parcels and 411 patches, with different configurations of parcel costs and the initial population. For each map, we considered problems involving different conservation designs, where each design corresponded to the upfront solution when the budget is limited to a factor b of the total parcel cost of the map, where b ranged from 0.1 to 0.5. In this section, we present a very similar analysis as in Section 2.5.1 and show consistent results, indicating that our primal-dual algorithm is stable across different problems.

2.5.2.1 Comparing to Optimal Solutions

We first compare the upper and lower bounds returned by our primal-dual algorithm to the optimal objective values of MIP and LP as computed using CPLEX. Since CPLEX still has scalability issues when solving the larger synthetic problems, we restrict the comparison to problems with 2 and 4 scenarios and a horizon of 20 years.

Results on all of the 10 maps are very similar. As an example, Figure 2.8 shows the

surrogate cost (PD-UB) and dual objective value (PD-LB) of the primal-dual solution on map 768, together with the optimal surrogate cost (CPLEX-MIP) and the lower bound computed for the LP by CPLEX. We see that compared to optimal, our algorithm can still produce solutions with similar costs, especially when the problem is easy (b is smaller). Also, we again see that the lower bound computed by CPLEX is better than the PD lower bound. However, the PD lower bound is still within a factor of 2.

2.5.2.2 Quality of Conservation Schedules

We now consider larger problems based on 10 cascade scenarios and horizon $H = 40$, for which the MIP and LP are not practically solvable.

We first compare the average accumulated reward of the schedule returned by our primal-dual algorithm and the upfront schedule. Figure 2.9 shows results for one of our maps, indicating that the rewards achieved by our primal-dual schedules are always very close to those of the upfront schedules, as desired. The results for other maps are very similar.

We also study the cost curves of the schedules in order to illustrate their flexibility compared to upfront schedules. Figure 2.10 presents the average cost curves for our schedules across all the 10 maps. It is noted that when the budget is very limited, the purchase is not delayed very much. For example, when $b = 0.1$, most parcels are purchased before $t = 15$, long ahead of the time horizon. On further analysis this can be explained by the fact that for many of the maps and such small budgets, the sets of affordable parcels are fairly spread out and loosely connected. This means that the population requires more time in order to reliably spread across such sets. Thus, the parcels must be purchased quite early in the horizon to support that spread. Rather for larger budgets, the sets of parcels that must be purchased are spread out but also more tightly coupled, which allows for easier, more reliable population spread. Thus, it is possible to delay purchases to a much larger extent as seen by the cost curves for the larger budgets. This shows that our algorithm is able to afford considerable flexibility when the initial conservation design supports reasonably reliable population spread.

2.5.3 Early-Stopping for Trading Off Flexibility and Reward

We now consider the early-stopping variant of our primal-dual algorithm, referred to as PD-ES, for producing schedules that trade off flexibility for reward using $\hat{\varepsilon} > 0$. The dataset we use here is the real conservation map. Figure 2.11 illustrates the cost curves of the early-stopping schedules with $\hat{\varepsilon} = 0.0, 0.05, 0.10, 0.15, 0.20$ and $H = 20, 40, 60, 80$, which demonstrates the possible budget saving over time if a corresponding fraction of reward loss is allowed. Figure 2.12 shows the average simulated rewards of these schedules.

First we notice that the average reward achieved by the early-stopping schedules is almost always within the specified error tolerance, which shows that the pruning step is generalizing effectively. We also see that the cost curves of the early-stopping schedules show significant improvement for even small values of $\hat{\varepsilon}$ compared to no early-stopping ($\hat{\varepsilon} = 0$). For example, when $H = 60$ and $\hat{\varepsilon} = 0.20$, there is almost no cost during the first several years, and for several decades the cost is approximately half of the cost required for $\hat{\varepsilon} = 0$. These results show that our approach is able to provide a set of schedules that spans a spectrum of trade-offs, which can then be considered by conservation managers.

2.6 Summary

In this work, we addressed the problem of scheduling purchases of parcels in a conservation design. We formulated the problem as a network cascade optimization problem which was reduced to a novel variant of the classic directed Steiner tree problem. We showed that this problem is computationally hard and then developed a primal-dual algorithm for the problem. Our experiments showed that this algorithm produces close to optimal results and is much more scalable than a state-of-the-art MIP solver. We also showed that an early-stopping variant of the algorithm is able to explore the possible trade-offs between flexibility and reward, which is an important consideration in practice.

The scheduling problem considered in this work poses considerable challenges to generic off-the-shelf schedulers and planners. The complicating factors include: 1) highly-stochastic, exogenous dynamics that arise from the population spread model, 2) the need to reason about spatio-temporal processes, 3) the long horizons that must be considered,

and 4) the combinatorial space of potential investment options at each point in time. The general solution schema we pursued in this work is likely to be applicable to other problems that pose similar challenges to existing techniques. In particular, this general schema suggests approximating the problem via the SAA and then studying the resulting deterministic optimization problem. Often the resulting deterministic problem will correspond to an existing well-studied problem, for which state-of-the-art approximation algorithms can be used. In other cases, such as in this work, the resulting problem will be related to an existing well-studied problem and a solution can be designed by extending existing solution frameworks. We expect that this generic SAA schema will be particularly useful for problems involving stochastic spread of populations or information across networks, since the deterministic problems will typical map to graph-theoretic problems, for which there is a vast literature.

$$\textbf{(Primal)} \quad \min \sum_{s=1}^{M} y(E_s) \times c_s, \quad \text{subject to:} \tag{2.1}$$

$$\sum_{(i,h)\in\mathbb{E}} x^k_{i,h} - \sum_{(j,i)\in\mathbb{E}} x^k_{j,i} = \begin{cases} 1, & \text{if } i = r \\ -1, & \text{if } i = k \\ 0, & \text{if } i \neq r,k \end{cases}, k \in T, i \in \mathbb{V} \tag{2.2}$$

$$x^k_{i,j} \leq \sum_{s:(i,j)\in E_s} y(E_s),\ k \in T, (i,j) \in \mathbb{E} \tag{2.3}$$

$$x^k_{i,j} \geq 0,\ (i,j) \in \mathbb{E}, k \in T \tag{2.4}$$

$$y(E_s) \in \{0,1\} \tag{2.5}$$

$$\textbf{(Dual)} \quad \max \sum_{k\in T} (u^k_k - u^k_r), \quad \text{subject to:} \tag{2.6}$$

$$\sum_{k,(i,j)\in E_s} w^k_{i,j} \leq c_s, s \in \{1,\ldots,M\} \tag{2.7}$$

$$u^k_j - u^k_i - w^k_{i,j} \leq 0, k \in T, (i,j) \in \mathbb{E} \tag{2.8}$$

$$w^k_{i,j} \geq 0 \tag{2.9}$$

Figure 2.3: MIP for the SW-DSG problem and the corresponding dual LP of the MIP's LP-relaxation. The SW-DSG problem is defined by a graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, a root vertex r, a set of terminal vertices T, and a set of edges sets $\mathcal{E} = \{E_s : s = 1,\ldots,M\}$, where each $E_s \subseteq \mathbb{E}$.

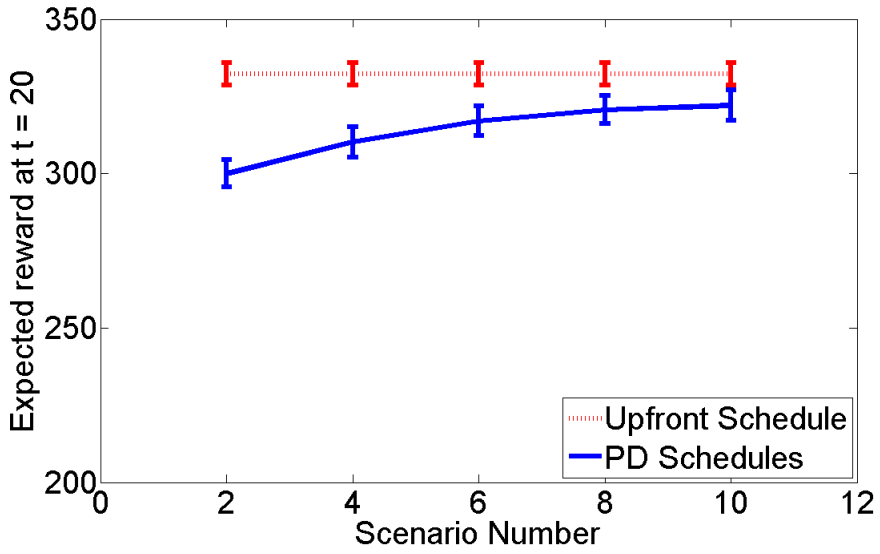

Figure 2.4: Rewards of PD solutions *w.r.t* the number of cascade scenarios.

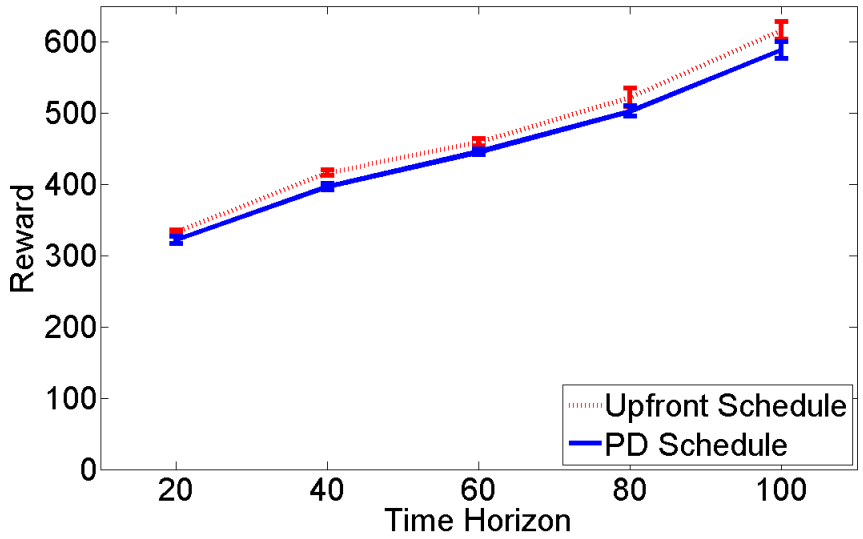

Figure 2.5: Rewards of PD schedules *w.r.t.* time horizon H.

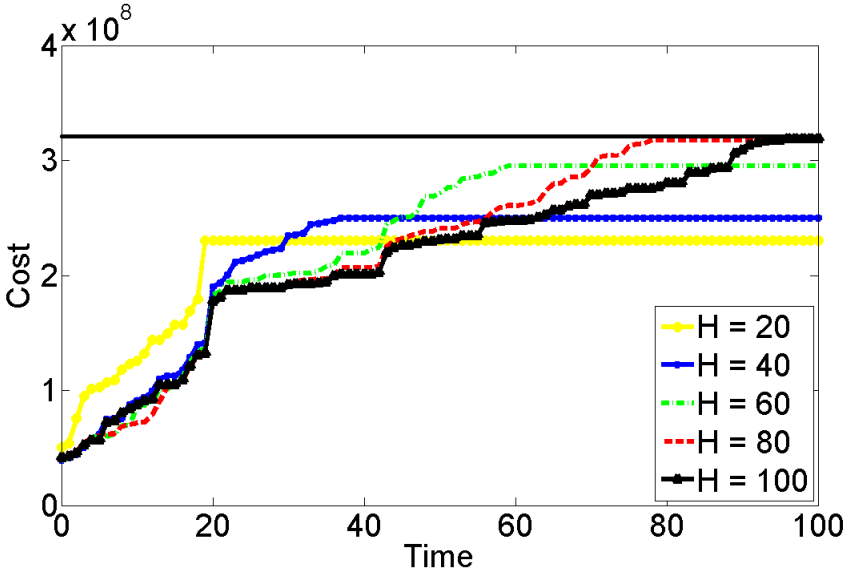

Figure 2.6: Cost curves of PD schedules for horizons 20 to 100.

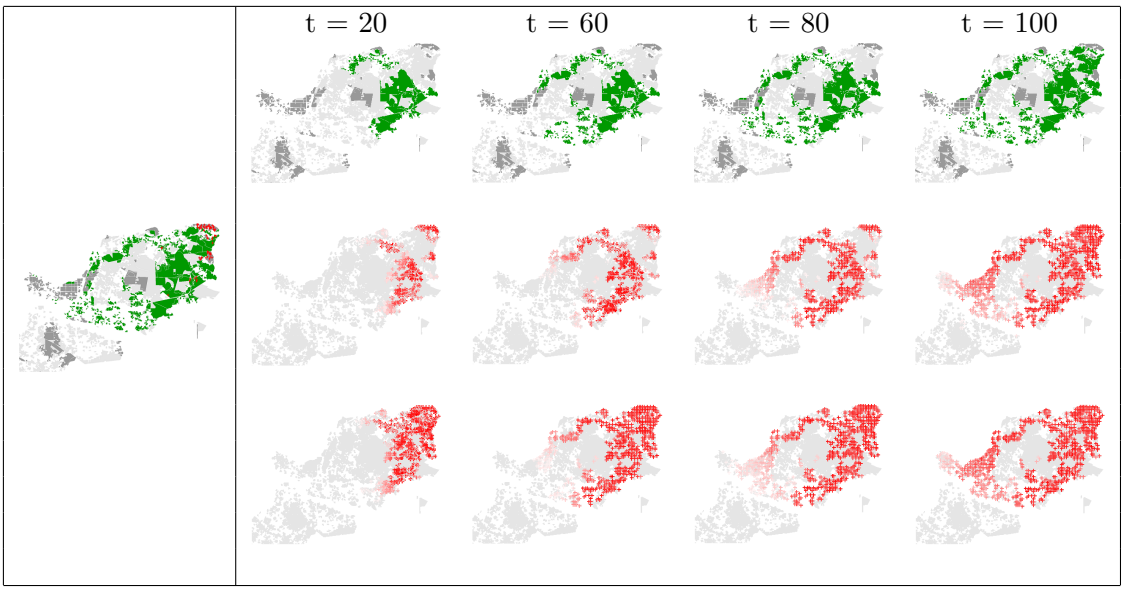

Figure 2.7: (Left) Original conservation design used for scheduling shown as green shaded parcels. Free (zero-cost) parcels are also shaded in dark grey and red '+' indicates initially occupied patches. (Right) The top row shows the parcels purchased (shaded green) by our PD schedule over a horizon of 100 years. The middle row shows the population spread over the same horizon for the schedule, where lighter red shading of a patch indicates a smaller probability of being occupied (as measured by 20 simulations). The bottom row shows the population spread of the upfront schedule over a horizon of 100 years.

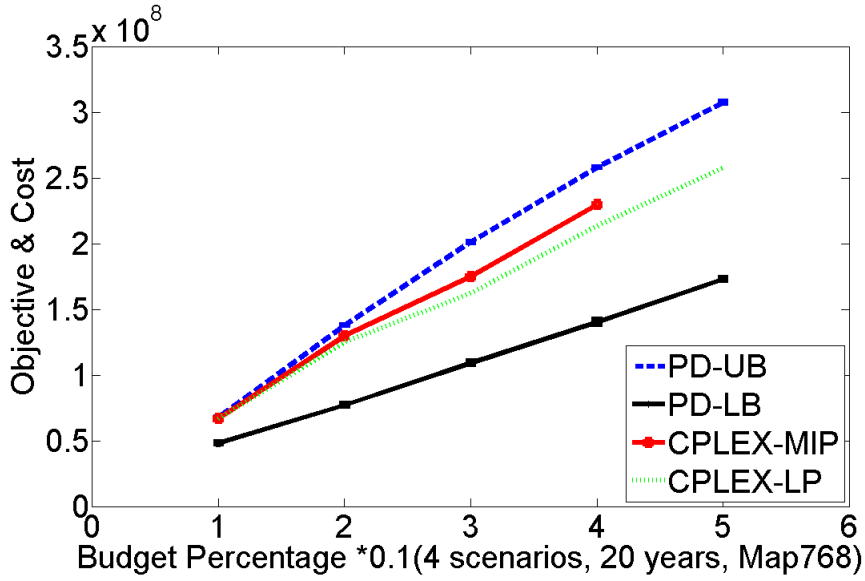

Figure 2.8: Cost and objective value of problems on map 768. The horizontal axis varies the amount of budget used to compute the upfront solution that is used as the conservation design given to the scheduling algorithms.

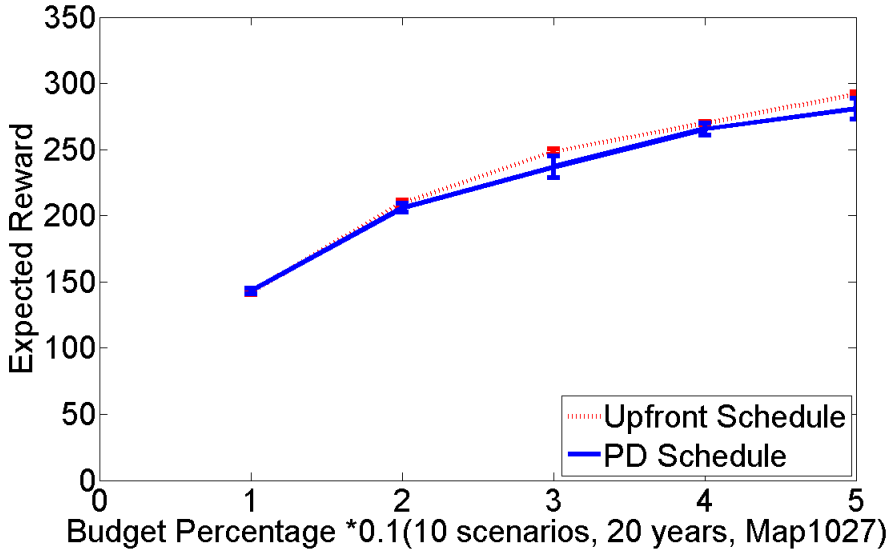

Figure 2.9: Rewards of primal-dual schedule and upfront schedule on map 1027.

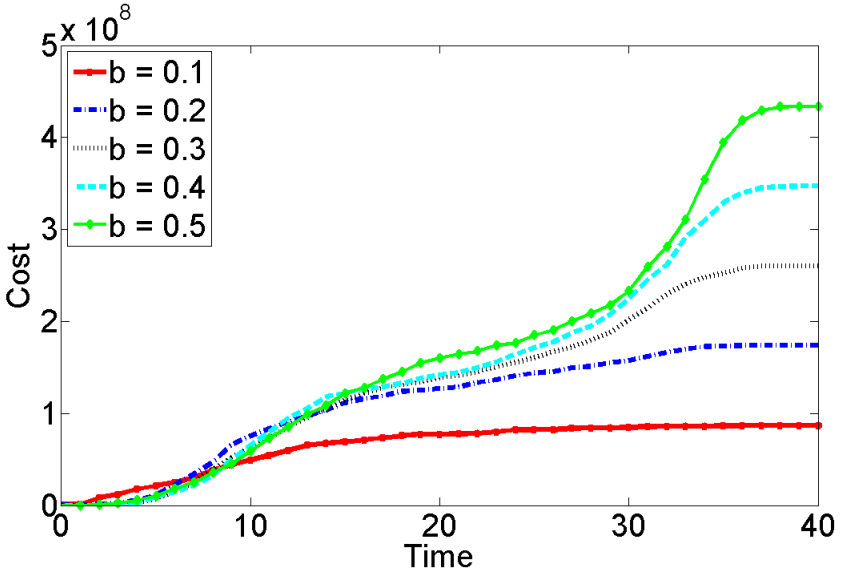

Figure 2.10: Average cost curves of PD schedules on 10 maps.

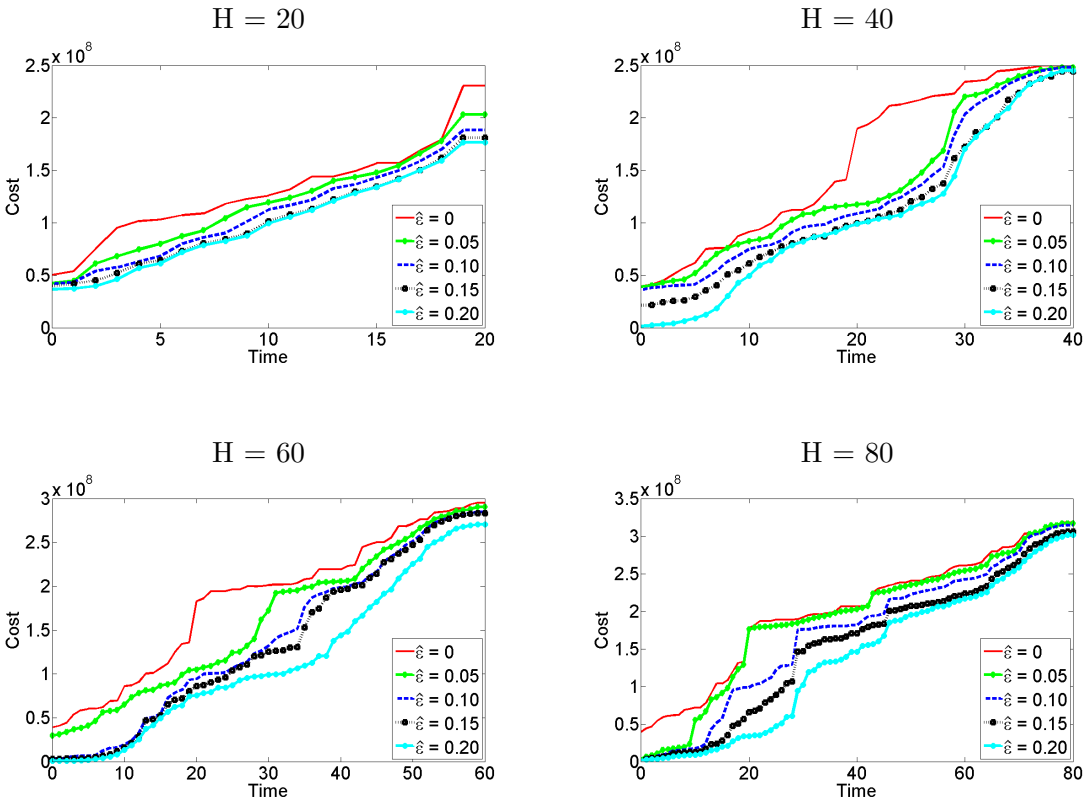

Figure 2.11: Cost curves of PD-ES schedules with pruning. The red line ($\hat{\varepsilon} = 0$) shows the cost curve of non-early-stopping PD schedule.

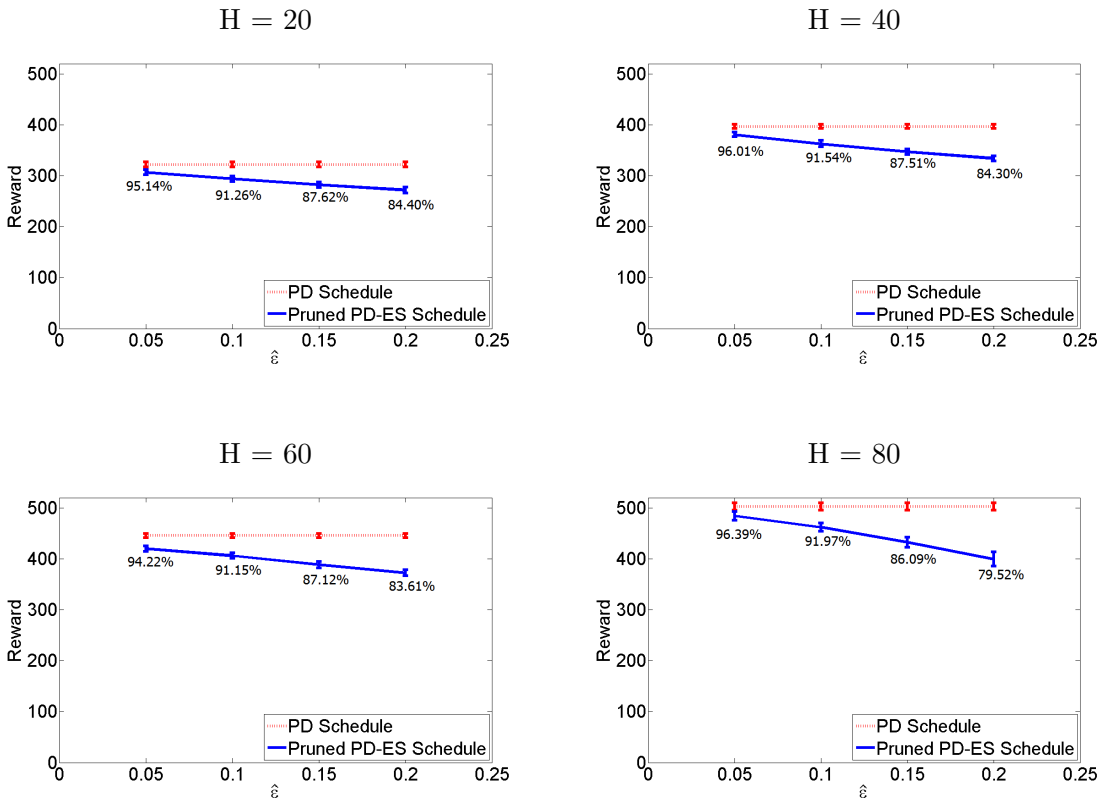

Figure 2.12: Rewards of primal-dual schedules with early-stopping. The number below each data point indicates the percentage of PD-ES reward over PD reward.

Chapter 3: Dynamic Resource Allocation for Conservation Planning

3.1 Introduction

Now we address the adaptive conservation planning, where the goal is to maximize the population growth of a species by purchasing and reserving land parcels to best support population spread. This optimization problem is challenging as it requires reasoning about the stochastic population spread on a large network, uncertain future budgets, and a combinatorial action space (the set of possible investment combinations).

Our work fills the research gap in conservation planning with an adaptive approach for highly spreading species. We take the future population dynamics, the future budget, and the future actions into account. Our approach is based on hindsight optimization (HOP), which computes an upper bound on action values and selects the action that has the maximum value. Unfortunately, standard implementations of HOP scale linearly in the number of actions available at any time, which is prohibitive for our exponential action space. Thus, our main contribution is to develop an efficient algorithm for computing HOP policies for such exponentially large, factored action spaces. We accomplish this by representing the HOP policy via a large Mixed Integer Program (MIP) and then applying the Dual Decomposition schema to make its solution more practical. Our experiments show that HOP can significantly outperform more myopic alternatives while also showing scalability to large problems.

3.2 Related Work

Generally, our conservation planning problem can be seen as an adaptive probabilistic planning problem. Then we may consider existing planning approaches. Encoded as a Markov Decision Process (MDP), this conservation planning problem is still challenging for state-of-the-art planers as the corresponding state space and action space are very large.

For instance, tree search is a particularly popular approach for probabilistic planning,

which estimates action values for the current state by constructing a lookahead search tree. Many tree search algorithms have shown strong empirical performance in very challenging domains. Examples of tree search planners include heuristic search methods like AO* [42], as well as simulation-based methods like UCT [23] and its variants. However, in practice, the performance of these approaches strongly depends on the action branching factor. In domains with a considerable action space, the feasible search tree depth is greatly limited. While it is helpful to address this problem by pruning bad actions in the tree (e.g. [45]), it is still a challenge to apply search methods to large factored action spaces.

Symbolic dynamic programming (SDP) algorithms also attempt to save computations by using symbolic representations of policies and value functions. Though SDP has been developed for factored action space ([47]), its scalability to large problems is still limited since it aims to compute policies over the entire reachable state space.

Another possible online planning approach is Hindsight Optimization (HOP), which has been successfully applied to a variety of difficult stochastic planning problems, e.g. [10], [12], [58], [60], [31]. At every decision step, it optimistically estimates the value of each action by simulating future samples assuming that the optimal policy is always followed after this action is taken. The action with the best value is then selected as the next action. Unfortunately, the standard implementation of HOP enumerates all the possible actions at any time, which would have a computational problem with our huge, combinatorial action space.

In an adaptive conservation planning problem for slow-moving species, [25] proposes a simple greedy policy, which is near-optimal since their problem is submodular. This policy selects the best action that can achieve the best future goal while assuming that no actions would be taken after current time step. We name this policy as HNoop as it assumes only Noop actions are allowed in the future. HNoop is a strong policy, yet it is easy to prove that for problems without submodularity, HNoop may perform arbitrarily worse.

3.3 Problem Statement

3.3.1 Population Model

The population model here is the same as the one in Chapter 1. In addition, there is a stochastic budget process, where new funds arrive each year according to some distribution. Parcels can only be purchased when enough funds are available and hence must be purchased incrementally. Since the population can only spread to patches in conserved parcels, the population diffusion is strongly influenced by parcel purchases.

3.3.2 Adaptive Planning Problem

We consider a finite-horizon setting, where the goal is to optimize the total population across patches at the specified horizon H. Decision epochs occur every T_r years and at each a decision is made about the set of parcels to purchase, limited by the current budget. Thus, our planning problem is to produce a policy π that is given the current problem state at a decision epoch and outputs a set of parcels to purchase with cost no more than the current budget $B(t)$. Here the problem state is composed of the time-to-horizon, the species occupancy at each patch, the current budget, and knowledge of previous purchases. We follow a model-based approach, where π can be computed using provided stochastic models of the population dynamics and budget.

It will be useful to define the notion of a future F, which is a random variable encoding the random outcomes that dictate the future population spread and budget over the horizon H. In our setting, a realization f of F deterministically dictates whether a population can spread between two patches at any particular time and the budget at each future year. Such a future can be visualized as a deterministic network as in Figure 3.1. In the network, all the patches repeat at every time step and an edge between patch u at time t and v at time $t+1$ indicates that if the species is at u at time t then it will spread to v at time $t+1$.

For a fixed realization f, we effectively get a deterministic problem, for which any policy can be evaluated against. We say a policy π is feasible for a future f *iff* at any time step the total cost of parcel purchases of π is within the budget limit of f. For a future f, a feasible policy π, and initial state s_0 we denote the total population/reward achieved at the horizon H by $R(s_0, f, \pi)$. The optimal solution to our problem can

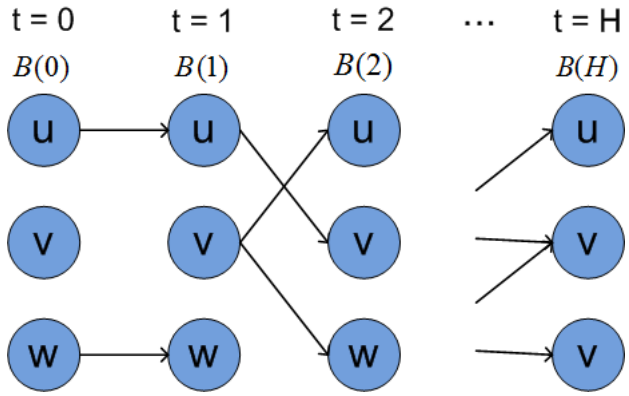

Figure 3.1: Visualization of An Example Future. Suppose there are totally 3 patches: u, v, and w, then each patch repeats at each time step. The edge indicates the colonization from one patch to the other at a certain time step.

then be expressed as finding a policy π^* that maximizes the expected reward, i.e. $\pi^* = \arg\max_\pi \mathbf{E}[R(s_0, F, \pi)]$.

An exact solution for π^* is beyond the capabilities of existing planners. To address this, prior work [25] used a simple policy based on myopic, greedy action selection. The main contribution was to give approximation bounds for the policy under strong assumption of no diffusion across parcels. Rather, here we develop a non-myopic action selection heuristic, hindsight optimization, which has shown success in various other areas of planning. While hindsight optimization also requires strong assumption to provide performance guarantees, our experiments demonstrate that its non-myopic nature can lead to significant improvements over [25].

3.4 Approach

3.4.1 Hindsight Optimization for Conservation Planning

Hindsight Optimization. The main idea behind hindsight optimization (HOP) is to drive action selection by computing upper bounds on the values of states. Given a state s, the hindsight value of the state $V_{\text{hs}}(s)$ is an optimistic estimate of the true value obtained

by interchanging expectation and maximization, i.e. $V_{\text{hs}}(s) = \mathbf{E}\left[\max_\pi R(s, F, \pi)\right]$. This clearly gives an upper bound on the value since it allows for inconsistent policies to optimize each future.

The key computational advantage of working with V_{hs} rather than directly with the value function V is that it can be approximated by solving a set of deterministic planning problems for individual futures. That is, given a set of sampled futures $\{f_1, f_2, \ldots, f_K\}$ we estimate the hindsight value as:

$$\hat{V}_{\text{hs}}(s) = \sum_k \left[\max_\pi R(s, f_k, \pi)\right]$$

where the internal maximization problem for each f_k can often be solved with existing solvers. The hindsight Q-function $Q_{\text{hs}}(s, a)$ is accordingly defined as the expected hindsight value achieved for states reached by taking action a in state s and is also an upper bound on the true Q-value of a state-action pair. The HOP policy for a state s is then defined as $\arg\max_a Q_{\text{hs}}(s, a)$. Under certain assumptions performance guarantees can be made for the HOP policy ([40, 59]). However, in general, no guarantees can be made and examples of arbitrarily poor performance compared to optimal can be constructed. Fortunately such examples are often not reflective of real problems and the HOP policy is often an effective way to select action non-myopically.

The traditional way to compute the HOP policy is to estimate $Q_{\text{hs}}(s, a)$ by sampling a set of states resulting from taking a in s, computing the hindsight value for each state, and averaging. Unfortunately, this traditional approach scales linearly with the size of the action space and hence is not feasible for the combinatorial action space in our conservation problem and many others with factored actions.

HOP for Large Action Spaces. The traditional computation of the HOP policy estimates Q-values for each action for the purpose of maximizing over them. However, this is not strictly necessary if we can directly compute the optimizing HOP action at a state without explicitly estimating each Q-value. Thus, the main idea behind our approach is to encode the problem of computing the optimizing HOP action (Q-values) as a mixed integer program (MIP) and then apply decomposition techniques to solve it, which avoids the explicit enumeration over actions.

Our MIP for HOP is defined relative to a set of sampled futures $\{f_1, f_2, \ldots, f_K\}$ and a current state s. Similar to the formulation of [51], for each future f_k, we can define a

MIP, denoted as MIP_k, which encodes the problem of finding a policy that maximizes the reward of f_k. That is, MIP_k solves the problem $\max_{\pi_k} R(s, f_k, \pi_k)$, where π_k is represented as a binary vector that specifies for each parcel and time, whether to buy the parcel at that time in future f_k. We will denote by π_k^1 the parcel purchases specified for the first time step in MIP_k. We also let OBJ_k and CON_k denote the objective and constraints of MIP_k respectively and let V_k be all variables in MIP_k excluding π_k. The detail of MIP_k is similar to that in [51] and not crucial to our main contribution. Since we have separate decision variables for each future, the maximization and summation in the Q-value function can be interchanged, which results in a MIP that encodes the HOP policy:

HOP Policy MIP:

$$\min_{\{\pi_k, V_k\}} -\frac{1}{K}\sum_k \text{OBJ}_k, \text{ s.t.}$$

$$\pi_1^1 = \pi_2^1 = \ldots = \pi_K^1 \text{ and } \bigcup_k \text{CON}_k$$

That is, we can compute the HOP policy by maximizing the sum (minimizing the negative sum) of objectives across the futures, subject to the constraint that the solutions for each future agree on the first action. This MIP can then in concept be given to any MIP solver and then the returned value of π_1^1 can be returned as the HOP action.

While in our experience, it is generally feasible to use existing MIP solvers to solve the individual MIP_k problems for realistic scenarios, when the number of futures increases, solving the combined MIP can become prohibitive. This is an important limitation since the variance of the HOP policy reduces with more futures. Fortunately, this issue can be largely overcome via the use of dual decomposition techniques as the HOP policy MIP reveals a separable structure.

3.4.2 Dual Decomposition for HOP

In the HOP policy MIP, the individual MIP_k problems are only coupled via the policy constraints on the first action. If the constraints are removed then the combined MIP can be solved by solving each MIP_k independently. This structure motivates the applica-

tion of Lagrangian dual decomposition, which we formulate below and follows a similar structure as prior work on upfront conservation planning by [37].

We start by rewriting the coupling constraint $\pi_1^1 = \pi_2^1 = \ldots = \pi_K^1$ of MIP_k as the set of constraints $\{\pi_k^1 = \mathbf{d} \ : \ k = 1, \ldots, K\}$ where $\mathbf{d}$ is a new vector of binary variables that represents the HOP policy action at the first time step. We let $\pi_{k,i}^1$ and d_i denote component i of π_k^1 and d, indicating whether i was purchased or not at time step 1. We can now relax these coupling constraints to get the Lagrangian of the HOP MIP by introducing Lagrangian multipliers $\lambda_{k,i}$ for each constraint.

$$L(\{V_k, \pi_k\}, \mathbf{d}, \boldsymbol{\lambda}) = -\frac{1}{K}\sum_{k=1}^{K} \text{OBJ}_k + \sum_{k,i} \lambda_{k,i}(\pi_{k,i}^1 - d_i)$$
$$\text{s.t.} \ \bigcup_k \text{CON}_k$$

The dual is then given by

$$\begin{aligned} q(\boldsymbol{\lambda}) &= \min_{\{\mathbf{V_k}, \boldsymbol{\pi_k}\}, \mathbf{d}} L(\{V_k, \pi_k\}, \mathbf{d}, \boldsymbol{\lambda}) \\ &= \min_{\{\mathbf{V_k}, \boldsymbol{\pi_k}\}, \mathbf{d}} -\frac{1}{K}\sum_{k=1}^{K} \text{OBJ}_k + \sum_{k,i} \lambda_{k,i}(\pi_{k,i}^1 - d_i) \\ &= \min_{\{\mathbf{V_k}, \boldsymbol{\pi_k}\}, \mathbf{d}} \sum_{k=1}^{K} -\frac{1}{K}\text{OBJ}_k + \sum_{k,i} \lambda_{k,i}\pi_{k,i}^1 \\ &\quad - \sum_i d_i \sum_k \lambda_{k,i}, \qquad \text{s.t.} \ \bigcup_k \text{CON}_k \end{aligned}$$

Intuitively, the relaxed constraints in the dual act as a penalty for violating the consistency requirement that all policies across futures agree on the first action. Since the dual minimizes over $\mathbf{d}$, in order to ensure that $q(\boldsymbol{\lambda}) > -\infty$ we require the constraint $\sum_{k=1}^{K} \lambda_{k,i} = 0, \forall i$. To simplify notation, we denote the space of Langrange multipliers that satisfy this constraint as:

$$\Lambda = \{\{\lambda_{k,i}\} \mid \sum_{k=1}^{K} \lambda_{k,i} = 0, \forall i\}$$

Under this constraint the last term in the dual vanishes and we finally get the dual which consists of independent subproblems for any fixed $\boldsymbol{\lambda}$:

$$q(\boldsymbol{\lambda}) = \min_{\{\mathbf{V_k}, \pi_\mathbf{k}\}} -\frac{1}{K} \sum_{k=1}^{K} \text{OBJ}_k + \sum_{k,i} \lambda_{k,i} \pi^1_{k,i}$$
$$\text{s.t.} \bigcup_k \text{CON}_k \text{ and } \{\lambda_{k,i}\} \in \Lambda$$

One important characteristic of the dual is that $q(\boldsymbol{\lambda})$ for any feasible $\boldsymbol{\lambda}$ is a lower bound on the optimal primal MIP objective value, which motivates attempting to make the bound as tight as possible by maximizing $q(\boldsymbol{\lambda})$ over $\boldsymbol{\lambda}$. Since $q(\boldsymbol{\lambda})$ is not continuous and the dual includes constraints over $\boldsymbol{\lambda}$ we use projected subgradient descent for this purpose, iterating as follows:

$$\boldsymbol{\lambda}_k^{(j+1)} = [\boldsymbol{\lambda}_k^{(j)} + \alpha_{j+1} g_k(\boldsymbol{\lambda}_k^{(j)})]_\Lambda \tag{3.1}$$

where j is the iteration number, $g_k(\cdot)$ is a subgradient of $q(\boldsymbol{\lambda})$ with respect to λ_k, α_j is the step size, and $[z]_\Lambda$ is the projection of z onto constraint space Λ.

For our objective, one subgradient of $q(\boldsymbol{\lambda})$ with respect to λ_k is $\bar{\pi}^1_k$ such that

$$\bar{\pi}_k = \arg\min_{\mathbf{V_k}, \pi_\mathbf{k}} -\frac{1}{K} \text{OBJ}_k + \sum_i \lambda_{k,i} \pi^1_{k,i},$$

which can be found by solving a minimization problem involving a single future f_k and hence is much more tractable than the full MIP. Note that this minimization problem is simply the original objective of MIP_k with an added term involving the current Lagrange multiplier values that can be viewed as assigning a penalty or reward for purchasing particular parcels at the first time step. Finally, given the subgradient, the projection onto Λ (with Euclidean norm) is well known, requiring only that we subtract from each

component of the subgradient the average component value. Letting $\bar{\pi}_k^{1,(j)}$ denote the subgradient at iteration j we get the following:

$$\boldsymbol{\lambda}_k^{(j+1)} = \boldsymbol{\lambda}_k^{(j)} + \alpha_{j+1}\left[\bar{\pi}_k^{1,(j)} - \frac{\sum_{k'=1}^{K} \bar{\pi}_{k'}^{1,(j)}}{K}\right] \tag{3.2}$$

This shows that the gradient steps for dual optimization can be computed by optimizing independent sub-problems for each future (i.e. solving for each $\bar{\pi}_k$), which avoids solving a single MIP involving all futures. Putting everything together, the complete dual optimization algorithm is given in Algorithm 2.

Algorithm 2 Dual Decomposition Algorithm

1: **Given:** initial vector $\boldsymbol{\lambda} \in \Lambda$
2: **while** convergence is not reached **do**
3: *Optimize subproblems independently:*
solve each subproblem and get $\{\bar{\pi}_k^{1,(j)}\}$
4: *Compute average value of* $\bar{\pi}_k^{1,(j)}$ *over* K *subproblems:*
$\hat{\boldsymbol{d}}^{(j)} = \frac{\sum_{k'=1}^{K} \bar{\pi}_{k'}^{1,(j)}}{K}$
5: *Update* $\boldsymbol{\lambda}$*:*
$\boldsymbol{\lambda}_k^{(j+1)} = \boldsymbol{\lambda}_k^{(j)} + \alpha_{j+1}[\bar{\pi}_k^{1,(j)} - \hat{\boldsymbol{d}}^{(j)}]$
6: **end while**

At a high level, the final algorithm involves optimizing the dual via iterations. Each iteration involves solving multiple modified MIP$_i$ problems, which are different from the originals in that the costs of certain purchases at the first time step are modified in order to encourage the subproblems to agree on the first actions. Specifically costs are increased for parcels that are not currently purchased by most futures and decreased for parcels that are purchased by many futures. The iteration ends when either the subproblems all agree on the first action, in which case we get the optimal HOP action, or the maximum number of iterations is reached, in which case we extract a solution as described below.

Feasible Solution Extraction. The above algorithm optimizes the dual and does not explicitly provide a primal solution, which in our case is the action of the HOP pol-

icy. After optimizing the dual it is often the case that the solutions to the independent subproblems (i.e. the π_k^1) are consistent and hence represent a feasible primal solution. In these cases, any of the π_k^1 are optimal primal solutions, and we output the resulting action as the HOP action. However, in general there can be a duality gap, and we are not guaranteed that optimizing the dual will produce feasible primal solutions. Thus, as is typical in Lagrangian relaxation techniques, we must define a strategy for heuristically selecting a feasible primal solution guided by the information obtained during the optimization.

Our approach is a heuristic based on the consistency requirement. As in the algorithm, we let $\hat{d}_i$ denote the average value of $\pi_{k,i}^1$ over different futures, which is 1 if parcel i is purchased in all futures and 0 if it is not purchases in any futures, and otherwise $\hat{d}_i \in (0,1)$ indicating the percentage of futures in which parcel i was purchased. To extract a HOP action $\mathbf{d}$ where d_i indicates whether to purchase i, we first set $d_i = 0$ whenever $\hat{d}_i = 0$, since purchasing i was not preferred in any future. Next, we cycle through each patch i with $\hat{d}_i = 1$ and purchase the patch (set $d_i = 1$). If there is remaining budget after processing all parcels with $\hat{d}_i = 1$, we sort all remaining parcels with $\hat{d}_i \in (0,1)$ in descending order. Then for each parcel, if purchasing it does not violate the budget constraint we set $d_i = 1$ and otherwise set $d_i = 0$.

Step Size Control. Correctly controlling the step size α_i can have a large impact on efficiency, since each iteration involves solving K MIP problems (one per future). We follow the same, relatively standard, step size control as [37], where the step size is computed according to the gap between the feasible primal solution quality and the dual solution quality. In particular, after extracting a feasible solution for the primal, let APX_j be the sum of rewards on every future and $DUAL_j$ be the dual objective value, we set

$$\alpha_j = \frac{APX_j - DUAL_j}{\sum_{k,t,i}(\bar{\pi}_{k,i}^{t,(j)})^2}. \tag{3.3}$$

3.4.3 Dual Decomposition for Baselines

There are multiple alternative heuristics that can be formulated within the same dual decomposition framework described above for HOP. These heuristics differ in terms of

the horizon over which they consider future population spread and whether or not they consider the possibility of selecting actions in the future when selecting actions at the current moment. Below we describe two baselines within this framework that are included in our experiments.

GreedyZero Policy. The GreedyZero policy is our most near-sighted baseline as it selects the action that looks best assuming the population growth and available budget after the next decision epoch is zero. In particular, the futures for GreedyZero are simulated for only T_r years instead of until time H. Correspondingly, its MIP is exactly the same as the HOP MIP except that the horizon H is always replaced by T_r, and no purchases are allowed after the first year.

HNoop Policy. The HNoop policy is unlike GreedyZero as it considers the population spread until the real horizon H. However, unlike HOP and similar to GreedyZero, it does not consider the possibility of selecting actions after the first time step. Thus, it evaluates purchasing actions according to how much long-term population spread they will facilitate assuming that the Noop action is taken thereafter. The computation of HNoop is similar to our approach for HOP, except that we simply remove all "action variables" from each MIP_k after the first time step, which prevents the consideration of future actions. This myopic policy will often work well, when the consideration of future actions is unimportant. However, when this is not the case, we might expect HOP to have an advantage. For example, in some conservation situations it is important to consider building longer-term paths for population spread in order to encourage spread to a particularly good habitat. Such paths will often not result from purely myopic reasoning.

Interestingly, the conceptual definition of the HNoop policy corresponds exactly to the myopic policy proposed in the only prior work on adaptive conservation planning in [25]. However, their computation of HNoop was carried out via a greedy algorithm that considered greedily adding parcels into the purchased set one at a time until the budget was exhausted. Given certain submodularity assumptions that greedy algorithm came with approximation guarantees. Our framework provides an alternative approach to computing HNoop that is not purely greedy, instead relying on decomposition for efficiency.

3.5 Experiments

Our evaluation uses a real dataset of the Red-cockaded Woodpecker (RCW) recovery project from [51] along with some hand-designed synthetic maps. The various problems differ in terms of the spatial layout of available parcels, the initial population of birds, and the set of parcels that are already reserved (i.e. free parcels). The real RCW map is from a large land region of the southeastern United States that was of interest to The Conservation Fund. The region was divided into 443 non-overlapping parcels (each with area at least 125 acres) and 2500 patches serving as potential habitat sites. Parcel costs were based on estimated land prices.

Throughout the experiments, we use a reliable MIP solver, IBM CPLEX, to directly solve MIPs if needed. Our experiments are performed on a single core machine with a memory limit of 6GB. We do not set time limit for the computation.

3.5.1 Performance of Dual Decomposition

Here we compare the use of Dual Decomposition (DD) for computing the HOP policy compared to directly applying CPLEX to the full HOP policy MIP, which includes all futures into a single MIP. This provides an indication of the optimality of DD, when CPLEX can solve the problems, and also the efficiency/scalability of the approach. We use the large RCW map and run DD until either the step size $\alpha \leq 0.001$ or a maximum of 50 iterations is reached.

Table 3.1 shows the HOP value computed by CPLEX and DD as well as the timing results for time horizon $H = 20$. When a method fails to return a solution, no value is shown in the table. From the table, we see that the objective value of DD is very close or equal to that of CPLEX, meaning that DD is providing an extremely close approximation to the HOP policy. When the number of future scenarios is small, the MIP instances seen by CPLEX are relatively small and can be solved quickly by CPLEX. As the number of futures increases, CPLEX takes a much longer time to find solutions compared to DD or even fails to solve the problem within a reasonable amount of time and memory. To make the problem more challenging, we increase the time horizon to $H = 40$ where the problem size is much larger. Table 3.2 shows the results. We see similar results, but they are more pronounced since the problems seen by CPLEX are now significantly larger.

Table 3.1: Solution Quality and Run Time(H=20)

K	CPLEX-OBJ	DD-OBJ	CPLEX-Time(s)	DD-Time(s)
5	-393.0	-392.4	26.8	112.0
10	-389.6	-388.0	41.9	100.3
15	-354.5	-353.2	71.2	79.0
20	-382.6	-381.4	102.5	210.8
25	-383.2	-381.2	368.7	159.8
30	-378.6	-375.0	650.8	187.1
35	-380.7	-377.9	430.1	205.6
40	-394.2	-392.8	1201.3	247.6
45		-391.3		280.3

Table 3.2: Solution Quality and Run Time(H=40)

K	CPLEX-OBJ	DD-OBJ	CPLEX-Time(s)	DD-Time(s)
5	-502.4	-502.4	61.8	94.4
10	-480.8	-480.4	196.9	181.4
15	-505.7	-501.7	1039.7	294.2
20	-504.6	-504.6	1417.2	358.2
25	-500.8	-499.7	4091.1	487.0
30		-502.3		568.5

The expensive computation and failures of CPLEX indicate that the full MIP approach is not practical to solve real-world problems.

It is important to note that DD can be easily parallelized to achieve significant speedup in terms of the number of processors. In particular, if we use one processor per future, the wall clock time per iteration would be equal to the maximum time required to solve an individual future. If those times are nearly uniform then this would result in nearly linear speedup.

3.5.2 Adaptive Planning Results on Synthetic Maps

We now evaluate the quality of the HOP policies and compare it with the GreedyZero and HNoop policies. The results reported for each algorithm are averaged over 10 runs to account for randomness of the environment and sample futures. We note that we have attempted to apply other planning formalisms to this problem, such as Monte-Carlo Tree

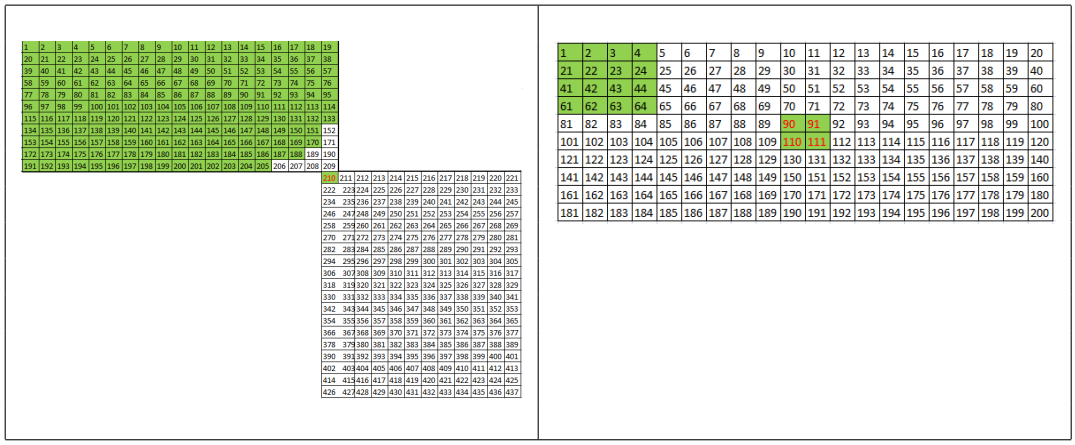

Figure 3.2: (Left) Grid Map 1. (Right) Grid Map 2. For both maps, the number marks the index of patches and each grid represents one parcel. The initially occupied patches are numbered in red and the free parcels are shaded in green.

Search, but without success due to the extreme action and stochastic branching factors of our problems.

We created two simple grid maps (Figure 3.2) to illustrate the advantage of the non-myopic HOP policy over the more myopic baselines. In the maps, each rectangle represents one parcel and each patch is marked using a number (its index). Note that most parcels contain only one patch while some contain two patches. The cost of each parcel is 1 and the annual budget is also 1. In other words, only one parcel can be purchased each year. In the first grid map, most parcels contain only one patch, but there are many parcels with two patches in the south of the initial population. Generally speaking, there are two possible directions of purchasing: to either the Northwest or the Southeast. Presumably, purchasing the Northwest part would lead the population to the most promising free area as long as the time horizon is large enough for the population to spread there, while the Southeast region provides more instant benefit as each year two patches would be available instead of only one. It is obvious that the optimal policy would follow the first strategy in order to maximize the long-term reward. The results shown in Table 3.3 illustrate that HOP recognizes the potential benefit of the free parcels and purchases parcels towards the correct direction, while GreedyZero and HNoop are easily distracted by the Southeast purchasing due to their myopia.

Grid map 2 is similar to grid map 1, but the purchasing is not limited to only two

directions so a planner would have more choices of purchasing, which also means more distractions from the far-away free area. Again, HOP recognizes the potential benefit of the free parcels and purchases parcels towards that direction, while GreedyZero and HNoop expand the reserve around the initial population uniformly. This leads HOP to achieve a higher reward as shown in Table 3.3.

Table 3.3: Rewards on Different Maps

Map	HOP	HNoop	GreedyZero
Grid map 1	85.2	35.0	70.2
Grid map 2	32	25.1	23.2
Real map	248.75	220.6	198.8

3.5.3 Adaptive Planning Results on Real Map

The real map (Figure 3.3) shows, via red + marks, where the initial bird population is, and free parcels are shaded in dark gray. Parcels shaded in pink are expensive yet affordable ones. The right map gives the natural population spread that would result if all parcels were conserved. We see that the free area in the Northeast corner is promising for optimal reward, therefore the optimal solution prefers to build a path from the initial population to it as long as there is enough budget and time for diffusion. However, many parcels on such a path are comparatively more expensive, adding more distractions for myopic decision makers.

We present the reward data for the three policies in Table 3.3, showing that HOP gains more reward than others. To further check their strategies, we plot the purchased parcels and corresponding population spread of the HOP and HNoop policies in Figures 3.4 and 3.5. While not shown, the GreedyZero policy gradually purchases parcels around the initial population. Compared to GreedyZero, HNoop finds the Southwest part more beneficial for longer-term population spread, so more parcels are bought in that direction. However, HNoop fails to recognize the most promising free area in the Northeast, which requires consideration of future actions. HOP recognizes the Northeast area and purchases expensive parcels to build a path to the free parcels. In addition, HOP also buys parcels around the initial population if the reward gain is justified.

3.6 Summary

We presented an action selection approach for adaptive conservation planning where it is necessary to dynamically reason about an extremely large set of resource management decisions and how they will impact the stochastic population spread. Our algorithm, based on hindsight optimization (HOP), is the first non-myopic approach for this problem and was shown to be effective compared to natural baselines. The main technical contribution was to show how to compute HOP policies for huge branching factor action spaces, for which prior HOP algorithms were inapplicable.

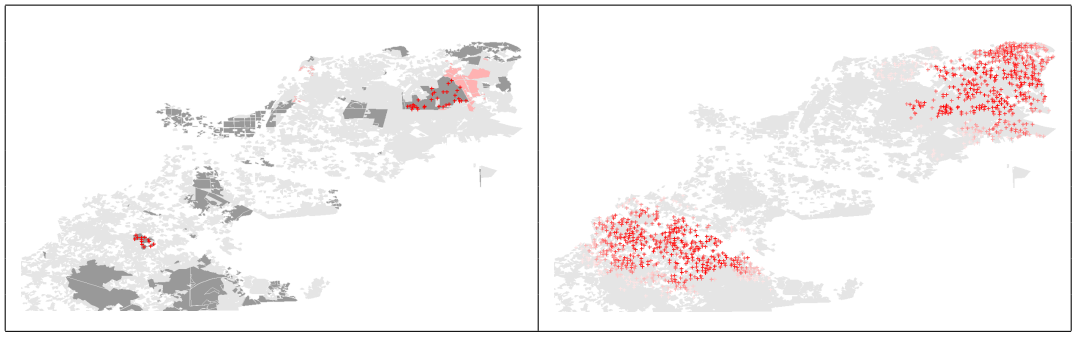

Figure 3.3: (Left) Initial state of the real map. (Right) The population distribution after 20 years on the real map.

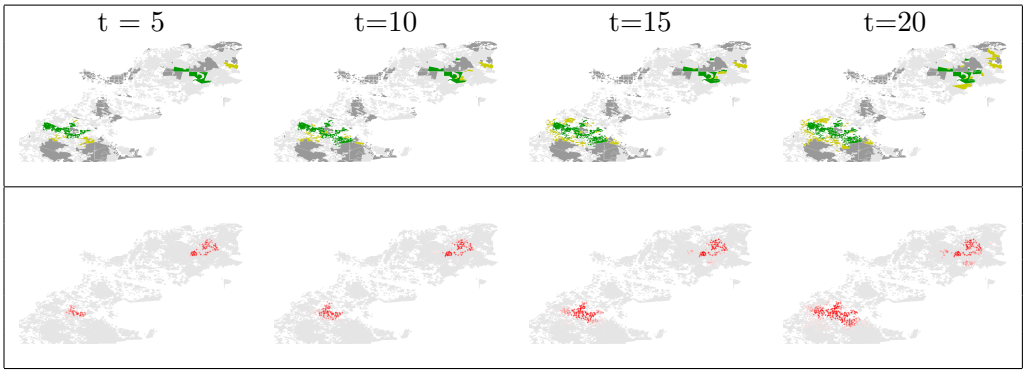

Figure 3.4: Purchases and population spread of HNoop. Parcels shaded in green are purchased with a probability of ≥ 0.5. Yellow is used for parcels with purchase probability of < 0.5. Patches are colored by the probability of being occupied (lighter color indicates lower probability).

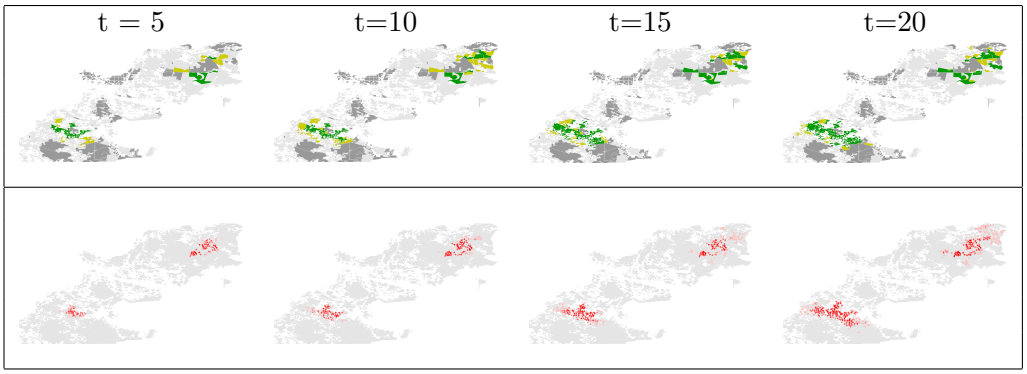

Figure 3.5: Purchases and population spread of the HOP policy. The color setting is the same as Figure 3.4.

Chapter 4: A Framework for Online Planning in Diffusion Networks

4.1 Introduction

In this chapter, we address the online planning problem of diffusion network control in a general way. At every time step, a planner takes the most recent state information into consideration and provides a control action for the current step that specifies the operations to take right away on certain sets of nodes in the network. The ultimate goal is to optimize some measure of the long-term realization of the diffusion process, i.e. to minimize the number of infected individuals in an epidemic breakout.

This dynamic optimization problem is challenging for existing planners due to several factors. First, the diffusion process poses highly stochastic, exogenous dynamics that arise from the large-scale network and the probabilistic spreading model. Second, the action branching factor is usually large due to the possible combinations of node sets at each time step. In addition, the long time horizon of interest adds one more dimension to the policy space.

The first contribution of this chapter is to provide a general formulation of the online planning problem in diffusion networks. We explore multiple realistic examples of diffusion network control. Even though they share similar diffusion processes in networks, each application has domain specific settings of node states, diffusion models and control models. The formulation we propose is generic enough to incorporate a variety of problems with diffusion network control.

Another contribution of this chapter is to explore an effective method to solve the problems with diffusion network control. We describe how to adapt our HOP algorithm to a solution schema for handling the complexities in this group of problems. From a higher level, we can view Hindsight [12], HNoop [25] and some other variants as a group of lookahead policies, which make decisions by looking into the future diffusion process and reasoning about the effects of each possible action. But they differ in terms of the horizon over which they consider about future spread and whether or not they consider the possibility of selecting actions in the future. Traditionally, each lookahead policy has

been implemented in very different ways. Unfortunately, a direct application of these approaches is not practical mainly because they treat each action as atomic and exhibit serious scalability issues for our combinatorial action space. Therefore, our work can also be seen as an alternative way to compute these lookahead policies more effectively.

The evaluation of our approach involves probabilistic planning in two new domains: epidemic control of influenza and dynamic firefighting. In the past decades, both diffusion applications have been well studied, but most work on diffusion network control has been limited to a static version of the planning problem in which all the actions are taken before the diffusion begins. This limits the solution quality in practice since it ignores the advantage of observing and responding to the stochastic outcomes of diffusion at every decision epoch. Thus, our approach fills a research gap in these domains by providing high-quality solutions for dynamic diffusion network control.

In addition, although we present our approach for diffusion networks, the possible application is not limited to this setting. At a high level, our approach is a framework of effective implementation of some lookahead policies, which can be applied to any probabilistic planning problem with a domain-specific tool for simulating future samples.

In what follows, Section 4.2 presents prior work related to epidemic control and Firefighting, followed by our problem formulation in Section 4.3. In Section 4.4 we present our approach schema that incorporates multiple lookahead policies. Descriptions of domains we study and the corresponding experiments are given in Section 4.5. Finally we conclude our work and suggest future research possibilities.

4.2 Related Work

4.2.1 Epidemic Control

Epidemic control has been a very important issue for protecting public health and the problem of diseases propagation has attracted huge interest by researchers in various areas.

The earliest study of epidemic propagation focus on epidemiological models of various diseases such as chickenpox, HIV/AIDS and influenza ([4], [17]). Hethcote [29] gives a general transfer diagram to model the possible status change on an individual. There are five compartments labeled as M, S, E, I and R in their diagram. M denotes people

who are new infants and have temporary passive immunity to the disease by receiving antibodies from the mother. After the maternal antibodies disappear, the infant becomes susceptible (S). Any other people who do not have antibodies are also susceptible individuals, *i.e.* they can get infected. A disease transmission occurs when there is adequate contact between an infected (I) individual and a susceptible (S) one. When this occurs, the susceptible enters the exposed (E) stage with a latent period, when they are infected but not yet infectious to others. After the latent period ends, an exposed (E) individual becomes infective (I) and is capable of transmitting the disease. When an individual gains permanant immunity, the status is recovered (R). Note that depending on the characteristics of a particular disease, its model may miss some of the five compartments. Here we are applying our approach to influenza control where infants are not crucial, so we omitted M and use SEIR model in our work.

Based on these models, one research direction is to estimate the growth of the infected population as a whole. The goal is to find an epidemic threshold of a network so that the disease would naturally die out. In particular, in the disease spreading network, each edge is associated with β which is the rate of infection, and each infected node is associated with δ which is the virus death rate. An epidemic threshold τ is the value that a disease breakout dies out quickly if $\beta/\delta < \tau$. With different models of the network, the epidemic threshold is proposed in multiple works. Bailey[4], McKendrick[39] and Anderson et al.[48] give the epidemic threshold for homogeneous graphs, where each individual has equal connection with others. In many real-world networks, a power-law structure is followed, in which there exist a few nodes with high connectivity but the majority of the nodes have low connectivity. Pastor et al. ([43], [44]) and Eguiluz et al. [19] derive the epidemic threshold for such power-law networks and show that it is more accurate for realistic epidemic networks. Newman et al. [41] address the disease spread problem in small-world networks with high clustering ([56]). Chakrabarti et al. [9] further develops a nonlinear dynamic system to more accurately model an arbitrary network and proposes the epidemic threshold to ensure an exponential disease die-out rate.

However, studies on epidemic thresholds do not model specific individuals and do not include options to treat individuals. Researchers also study how to use immunization to control propagation. Several papers show that targeted vaccinations can effectively reduce epidemics, though they usually have certain assumptions about the networks. For

instance, Briesemeister et al.[8] studied a random immunization on power-law graphs. An acquaintance immunization policy was proposed in [13] , which is proved to be much better than a random policy. The acquaintance immunization policy tends to immunize nodes that are connected with more neighbors (like a hub in the network). Following this idea, Madar et al. [38] and Dezso et al. [16] give a hub-based policy for vaccinations in scale-free networks. Later some studies model vaccinations using graph theory, especially graph cut (e.g. [3], [20], [28]). To minimize the number of infected individuals, they use vaccinations to create a barrier or cut to block the diseases. Goldenberg et al. [24] further extends the problem to a probabilistic model where each edge is associated with its own transition probability and the infection transmission events are independent among all edges. To solve this NP-hard problem, they formulate a nonlinear integer program with high-order multilinear terms. Then they propose a quadratic formulation and solve it with a state-of-the-art MIP solver. The solution provides a lower bound and a feasible solution to the original problem.

Note that all the above studies with controls (immunization) work on static problems where vaccinations are applied before the epidemic begins to spread. This is analogous to the upfront conservation problem by Sheldon et al. [51] where all the purchases are made at time 0. In contrast, we address a dynamic, adaptive version of epidemics controlling problems where the decision of vaccinations and quarantines is made at every time point for the most recent state and available medical resources.

4.2.2 Stochastic Firefighting

Fire spreading is another important diffusion network control problem. The protection of forests/landscapes from wild fire is important, as the fire can cause loss and damages to human life, property and the ecological system.

Introduced by [27], the traditional *Firefighter problem* is defined on a graph of land patches and a source node s. At any time, each node in the graph can have one of the following states: *susceptible*, *protected*, or *burning*. In the beginning, only s is burning and all the other nodes are susceptible. Then at every time step, the fire spreads from each burning node to all of its susceptible neighbors, unless a susceptible node is protected at or before current time step. Burning nodes and protected nodes remain burning and protected respectively afterwards. The diffusion process terminates when the fire no

longer spreads and the goal of this problem is to find a protection strategy to maximize the safe nodes (either protected or remain susceptible and isolated from burning nodes) in the end with a budget constraint B (at most B nodes can be protected at each time step).

This Firefighter problem is a deterministic, discrete-time model of the spread on a graph. It is NP-complete even on trees with maximum degree three [21]. There has been tremendous work (see e.g. [22] for a survey), most of which focuses on special graphs, such as trees and grids. Anshelevich et al. [2] considers variants of the Firefighter problem on general graphs. The spreading-vaccination version of the problem, where the protection/vaccination also spreads deterministically to neighbors as the fire spreads, can be reduced to maximizing a submodular function with a partition-matroid constraint. There is a randomized algorithm with $(1 - 1/e)$ approximation and a greedy algorithm with $1/2$ approximation for solving it. Recently, Spencer [52] captures the key tensions in containing invasive species with unreliable biological control agents, i.e. each protection/vaccination has a probability of effectiveness. Since their objective function is still submodular in this stochastic setting, the approximation algorithms from [2] are extended.

Spatial wildfire spread model has been studied by researchers in Forest engineering for decades. See [53] for a review of early attempts for modeling the behaviour of wildland fires and how to simulate wildfire spreading across landscape.

Wildfire management is another research direction researchers are more interested recently. Examples include reduction of fire suppression costs [30], resource sharing [54], fire fuel reduction, etc. With more collaboration between Forest Management and Artificial Intelligence, Reinforcement Learning has been applied to solve various forest planning problems [46]. [14] and [15] model the decision making problem as a large scale spatial-temporal planning problem and present a policy gradient approach to handle it. Another work on introducing AI approach to wildfire management problem is given in [30], where Monte Carlo method is used to show that sometimes it is more cost-effective to allow a light fire to burn.

4.3 Problem Statement

In this section, we introduce the formal notation of stochastic diffusion networks and the control actions so that it can represent most diffusion network control problems. The main stochastic planning problem is defined afterwards.

4.3.1 Basic Notation

A stochastic diffusion network involves a directed network (graph) $G = (V, E)$ where the node set is $V = \{v_1, \ldots, v_N\}$. At any time t, the state of each node v_a is an element in the set of node states $S = \{x_1, \ldots, x_M\}$, which is denoted as v_a^t. Generally, a diffusion process is observed as nodes change their states when a behavior (information/idea/disease) cascades from node to node. To better describe a diffusion process, we assume that S is partitioned into three subsets $S = S_{Sus} \bigcup S_{Inf} \bigcup S_R$. S_{Sus} is the set of *susceptible* states. A node with a state $x_m \in S_{Sus}$ is not hit (affected) by the cascading yet but can be hit later. S_{Inf} contains *infectious* states that mean the node has been affected and is able to propagate the contagious behavior to other neighbors. $x_m \in S_R$ indicates the node can resist infections from others.

There are two events in a diffusion process. *Diffusion events* happen on edges of G as an infectious node propagates the contagion to a susceptible neighbor and changes that neighbor's state. In particular, suppose a is in an infectious state and b is in a susceptible state at t, then each edge $e_{ab} \in E$ is associated with a probabilistic model $P_{ab}(v_a^t, v_b^t, v_b^{t+1}), v_a^t \in S_{Inf}, v_b^t \in S_{Sus}, v_b^{t+1} \in S_{Inf}$, indicating the single-step probability that the behavior would spread from node a to node b at time t and b becomes infectious in the next time step. The mass function requires $\sum_{v_b^{t+1} \in S_{Inf}} P_{ab}(v_a^t, v_b^t, v_b^{t+1}) = 1, \forall v_a^t \in S_{Inf}, \forall v_b^t \in S_{Sus}$. We further assume that the diffusion events taking place on different edges are independent. Thus a node gets infected *iff* at least one of its parents pass the contagion to it. *Local events* describe how the state of a node changes without interacting with other nodes. We use $P_a(v_a^t, v_a^{t+1})$ to denote the probability of local state transition and $\sum_{v_a^{t+1}} P_a(v_a^t, v_a^{t+1}) = 1, \forall v_a^t \in S$.

Let $\{O_i\}$ be the set of binary control operations that can be applied to a set of nodes for changing their state. Each operation may have some preconditions specifying that the operation is only feasible on nodes with certain state values. Usually, with

limited control resources, each control operation is assigned with a cost $Cost(O_i)$. The operation model $T(v_a^t, O_i^t, v_a^{t+W(O_i)})$ models the chance that O_i, which is performed at time t, becomes effective after a time window $W(O_i)$ and changes the state of a at time $t + W(O_i)$. In most cases we have $W(O_i) = 1$, meaning that the operation modifies the state of a node in the following time step. However, for some operations such as vaccinations, a short time period $W > 1$ is needed in reality. This is to stimulate the real case that an individual's immune system takes a few days to produce antibodies successfully. Note that it is feasible to do no controls at all, which we denote as a NOOP operation.

At every decision epoch, based on the current state s, a policy $\pi(s)$ for diffusion network control gives a set of control operations, which is also named as one (control) action. In our settings, at every time step $t \in \{0, 1, \ldots, H-1\}$, we compute a new control action π^t to be performed immediately, where H is the time horizon of interest. But the decision epoch may be less frequent depending on the specific problem. The policy is also required not to violate any other domain specific constraints. We assume that at every time step, the control operations are always first applied before diffusion happens. Thus, given any policy, we can simulate a diffusion control process by deciding the value of each node at every time step according to the control model and diffusion model.

4.3.2 Stochastic Planning Problem

A general goal of any stochastic planning problem can be represented as finding a policy π^* that maximizes an objective function **OBJ**:

$$\pi^* \in \arg\max_{\pi} \mathrm{OBJ} \tag{4.1}$$

Depending on the specific application, the objective function may vary. But in diffusion network control problems, the ultimate purpose is to optimize an expectation of some measure of the long-term diffusion process. We present a general formulation of the objective function in (4.2). First, we use S_G to denote the set of goal states we would like each node to achieve and define a node reward $r(x_m), \forall x_m \in S_G$. Thus the diffusion utility is calculated by summing up the rewards for all the nodes with a goal

state. In some applications, it is also important to evaluate a plan based on its control actions. We define an operation measure $c(\pi^t)$ which is allowed to be different from the cost function of operations. To have a long-term evaluation, we include the time steps of interest in a set T_G and accumulate the evaluation at each time step in T_G. For instance, if we are only interested in the final influence of a diffusion process, then $T_G = \{H\}$ where H is the time horizon of interest. Another extreme example is $T_G = \{0, 1, \ldots, H\}$, which counts the diffusion realization of every single time.

$$\text{OBJ} = E\left[\sum_{t\in T_G}\Big[\sum_{a\in V, v_a^t\in S_G} r(v_a^t) - \sum_{O_i^t} c(O_i)\Big]\right] \tag{4.2}$$

4.3.3 Futures in Diffusion Networks

Note that in Equation (4.2), the expectation is taken over the stochastic realization of a diffusion process. For presentation purpose, we introduce the notion of a general future F, which is a random variable encoding the random outcomes that dictate the future diffusion and any other stochastic over some horizon H. Besides, a future is independent of actions within horizon H, but can incorporate any policy to show the impact of the actions. Therefore, the ultimate purpose of generating a future is to remove all the randomness in both the diffusion model and control model.

First, to eliminate the randomness introduced by the diffusion process, the simulation of a future instance requires H coin flips for each edge following the diffusion model and for each node following the local state transition model. If there are different probabilities in multiple cases, we do the realization for each case. For example, individuals at certain state have a higher infectiousness. Particularly, assume that the spreading probability $p_{ab}(v_a^t, v_b^t, v_b^{t+1})$ is p_1 when $v_a^t = x_m$ but p_2 with a node state $v_a^t \neq x_m$. Then we first flip one coin with p_1, set the diffusion result to 1 if Head is seen and 0 if we see Tail. Also we flip coin again with p_2 and record the result similarly. In addition, when the control operations have stochastic effects, it is also required to determinize such randomness. Then at each time step in horizon H, we flip coins according to the operation model.

In this way, a realization f of future F deterministically decides the state of each node at any particular future time under any conditions. Such a future can be visualized

as a deterministic network as in Figure 1. In the network, all the nodes in G repeat at every time step and an edge between a at time t and b at time $t+1$ indicates that if node a and b satisfy the diffusion preconditions at time t then the contagion will spread from a to b at time $t+1$.

For a fixed realization f, we effectively get a deterministic problem, against which any policy can be evaluated. We say a policy π is feasible for a future f *iff* at any time step, actions returned by π can be applied to f without violating any constraint in the domain. For a future f, a feasible policy π, and initial state s_0, we evaluate the diffusion result achieved during the horizon H by a reward function $R(s_0, f, \pi)$, which is equal to the measure in (4.2) of a single future f. The optimal solution to our problem can then be expressed as finding a policy π^* that maximizes the expected reward, i.e.

$$\pi^* \in \arg\max_{\pi} \mathrm{E}[\mathrm{R}(s_0, F, \pi)], \tag{4.3}$$

where the expectation is taken over the future variable F.

4.4 Approach Schema

Here we present our approach schema that incorporates multiple lookahead policies, as well as how our computation of a policy solution can be accelerated compared to their previous implementations. In particular, we evaluate the candidate actions by simulating multiple futures using the Sample Average Approximation (SAA) and optimizing a goal *w.r.t.* the samples via a compact Mixed Integer Program (MIP). For large-scale (MIPs) which can not be solved by off-the-shelf optimizers, we suggest an efficient way of computation with Lagrangian Relaxation technique.

4.4.1 Lookahead Policies

Lookahead policies are a group of policies that make a decision by optimizing future actions on forecasted scenarios over a certain time horizon. Compared to some elementary policies that do not consider the impact of decisions in the future, lookahead policies explicitly use the future information. Yet usually, the optimization is computed approximately due to the difficulty of direct optimization of a stochastic function.

Here we focus on four different types of lookahead policies. Though all the policies explicitly optimize the actions on some futures, they are different in the horizon of futures, horizon of action impact and how much policy consistency is required among different futures. We denote the horizon of future F as T_F, the horizon of a policy π as T_π, and the policy in future f_k at time t as $\pi_{k,t}$. A basic requirement for any policy, including lookahead policies, is that the first actions in the K futures are consistent. That is, $\Pi = \{\pi \mid \pi_{1,0} = \pi_{2,0} = \cdots = \pi_{K,0}\}$.

The **GreedyZero** policy is the most myopic lookahead policy in our framework. It only looks ahead to the next time step, *i.e.* the future horizon is 2 and action horizon is 1. Thus, for GreedyZero, the diffusion after the next time step is zero. Its policy space is $\Pi_{GZ} = \Pi \bigcap \{\pi \mid T_\pi = 1, T_F = 2\}$.

The **HNoop** policy looks further into the future than GreedyZero as it considers the diffusion process until the real horizon H. However, similar to GreedyZero, it does not consider the possibility of taking actions after the first decision epoch, *i.e.* only NOOP actions are taken afterwards and the action horizon is still 1. The corresponding policy space is $\Pi_{HNoop} = \Pi \bigcap \{\pi \mid T_\pi = 1, T_F = H\}$.

HNoop policy will often work well, when the consideration of future actions is not critical. But when this is not the case, we might expect policies that consider future actions to have an advantage. Interestingly, the conceptual definition of the HNoop policy corresponds exactly to the myopic policy proposed in the only prior work on adaptive conservation planning in Golovin et al. [2011]. However, their computation of HNoop is carried out via a greedy algorithm that considered greedily adding node operations into the operation set one at a time until the some constraints are violated. Given certain submodularity assumptions their greedy algorithm comes with approximation guarantees. Our framework provides an alternative approach to compute HNoop that is not purely greedy, but relies on decomposition for efficiency and optimality.

The **Hindsight** policy is a non-myopic policy as it makes use of both the future states and the future decisions for a long horizon. There is no consistency requirement for actions after the first time step. Thus there could be different policies in each future. The policy space is $\Pi_{HS} = \Pi \bigcap \{\pi \mid T_\pi = H, T_F = H\}$.

The main idea behind the Hindsight policy is to compute an upper bound of the optimal action value and then select the best action. The traditional implementation of Hindsight requires an enumeration of all the possible actions. For each action, its

value is estimated by sampling multiple futures that takes this action first, optimizing each future, and averaging their rewards. Policies on different futures can be different, resulting a higher reward value than the optimal.

The **Straightline** policy is the same as Hindsight except that at every time step, the decisions in every future should be the same. The solution is an offline policy that can be directly applied for T time steps. Its policy space is $\Pi_{SL} = \Pi \bigcap \{\pi \mid \pi_{1,t} = \pi_{2,t} = \cdots = \pi_{K,t}, \forall t > 0, T_\pi = H, T_F = H\}$.

Formally, for each lookahead policy above, the corresponding optimization problem is limited to a specific policy space. For instance, the optimal Hindsight solution is

$$\pi^*_{HS} \in \arg \max_{\pi \in \Pi_{HS}} \mathrm{E}[\mathrm{R}(s_0, F, \pi)], \tag{4.4}$$

An exact solution of this stochastic optimization problem is usually beyond the capabilities of existing planners since it is defined in terms of an expectation over a complicated diffusion distribution in large-scale networks. In the following, we develop an approach for approximating these lookahead policies within one framework, which is a reformulation of our HOP algorithm in Chapter 3. Since their essential steps are the same, here we present the procedure briefly.

4.4.2 A Framework for Computing Lookahead Policies

In the following, we describe our approach mostly using notations of Hindsight policy. However, it is straightforward to generate the details for other lookahead policies.

Deterministic Optimization Problem. The objective function (4.4) demonstrates the stochasticity of a diffusion network, which we deal with by employing *Sample Average Approximation* (SAA), so that the stochastic problem is transformed into a deterministic one as follows, where the infinite number of futures are replaced with K futures. Theoretically, as K increases, the solution of problem (4.5) converges to that of problem (4.3).

$$\pi^*_{HS} \in \arg \max_{\pi \in \Pi_{HS}} \frac{1}{K} \sum_{k=1}^{K} \mathrm{R}(s_0, f_k, \pi), \tag{4.5}$$

Encoding with Mixed Integer Programs. A direct way of computing (4.5) is to translate it to a Mixed Integer Program (MIP) that represents the objective function as well as all the constraints in the specific diffusion network domain. In the Appendix, we give the MIP encodings we use, but any other approaches also work as long as the MIP correctly represents the corresponding problem. By adding a negative sign to the objective function and replacing max with min, we have a general form of MIP for (4.5) in (4.6), where CON includes every domain specific constraint.

$$\begin{aligned} &\min_{\pi,X} -\frac{1}{K}\sum_{k=1}^{K} \mathrm{R}(s_0, f_k, \pi) \\ s.t. \quad &\mathrm{CON}\bigcup \pi \in \Pi_{HS}. \end{aligned} \tag{4.6}$$

In our experience, (4.6) can be directly solved via a standard MIP optimizer, such as IBM CPLEX optimizer, Gurobi MIP solver, etc. Unfortunately, as the number of futures increases to ensure an empirically good approximation result, the MIP of a practical problem often becomes unsolvable for generic MIP solvers, which is observed as either failing to handle a very huge MIP or taking too much time to complete the computation.

Dual Decomposition Algorithm. In order to scale (4.6) to the number of futures, we explore a separable structure in it. Now we have separate decision variables for each future and add binary variables $\mathbf{d}$ to force agreement of operations at the first time step, then problem (4.6) is re-written as

$$\begin{aligned} &\min_{\pi_k,X_k} -\frac{1}{K}\sum_{k=1}^{K} \mathrm{R}(s_0, f_k, \pi_k) \\ s.t. \quad &\mathrm{CON}\bigcup\{\pi_{k,0} = \mathbf{d}, k = 1, \ldots, K\}. \end{aligned} \tag{4.7}$$

Now the problem reveals a separable structure that the policies of futures are only coupled via the policy constraints on the first action. If the constraints are removed then the combined objective function can be solved by optimizing over each future independently. With Lagrangian dual decomposition, we relax the coupling constraints on the first action and get the dual

$$q(\boldsymbol{\lambda}) = \min_{\{\mathbf{X_k}, \pi_\mathbf{k}\}} -\frac{1}{K}\sum_{k=1}^{K} \mathrm{R}(s_0, f_k, \pi_k) + \sum_{k,i} \lambda_{k,i}\pi_{k,0,i}$$
$$\text{s.t. } \bigcup_k \mathrm{CON}_k \text{ and } \{\lambda_{k,i}\} \in \Lambda$$
$$\Lambda = \{\{\lambda_{k,i}\} \mid \sum_{k=1}^{K} \lambda_{k,i} = 0, \forall i\},$$

which can be maximized iteratively using projected subgradient descent. The update step at iteration j is given by

$$\boldsymbol{\lambda}_k^{(j+1)} = \boldsymbol{\lambda}_k^{(j)} + \alpha_{j+1}\left[\bar{\pi}_{k,0}^{(j)} - \frac{\sum_{k'=1}^{K} \bar{\pi}_{k',0}^{(j)}}{K}\right], \tag{4.8}$$

where $\bar{\pi}_k$ can be found by solving a minimization problem involving a single future f_k and hence is much more tractable than the full MIP, i.e.

$$\bar{\pi}_k = \arg\min_{\mathbf{X_k}, \pi_\mathbf{k}} -\frac{1}{K}\mathrm{R}(s_0, f_k, \pi_k) + \sum_i \lambda_{k,i}\pi_{k,0,i},$$

Putting everything together, we get the complete dual optimization algorithm for a lookahead policy similar to Algorithm 2.

One important issue in this algorithm is the extraction of a feasible solution. When the algorithm converges, the solutions to the independent subproblems (i.e. the $\pi_{k,0}$) are consistent and hence represent a feasible primal solution. Then any of the $\pi_{k,0}$ are optimal primal solutions and we output the resulting action as the policy action. However, in general there can be a duality gap, and we are not guaranteed that optimizing the dual will produce feasible primal solutions. From our experience with different applications, different domains may require a domain specific heuristic for extracting feasible solution. In general, it is reasonable to choose an action that is voted for by more futures. It is also possible to include heuristics for action preference in this step. For instance, in influenza

control, we can choose to add an isolation control to the feasible solution as long as it is selected by at least one future, while a vaccination control is selected only when it is the majority choice.

Correctly controlling the step size α_j can have a large impact on efficiency, since each iteration involves solving K smaller MIP problems (one MIP encoding per future). We follow the same, relatively standard, step size control as [37], where the step size is computed according to the gap between the feasible primal solution quality and the dual solution quality. In particular, after extracting a feasible solution for the primal, let APX_j be the sum of rewards on every future and $DUAL_j$ be the dual objective value, we set

$$\alpha_j = \frac{APX_j - DUAL_j}{\sum_{k,t,i}(\bar{\pi}^{(j)}_{k,t,i})^2}$$

4.5 Experiments

To evaluate the performace of our lookahead policies in domains with diffusion network control, we apply our approach to an epidemic control problem and the stochastic fire-fighting problem. For each domain, we describe the domain settings, along with our implementation details and experimental results.

4.5.1 Epidemic Control of Influenza

Our application involves the spreading of influenza among people. Even though influenza is usually not fatal and people infected with influenza can recover by themselves after a period of time, the disease transfers fast and a large-scale breakout can still affect people's normal life and leads to severe illnesses in the young and the old. In the control of influenza, we aim to minimize the number of infected individuals as well as the total cost to reduce infection.

4.5.1.1 Domain Settings

We model influenza virus spreading among people as follows. In the population network, every individual person is a node and each edge represents the contact between individuals that can lead to virus transmission. We use the SEIR model [29] to describe the state of a node. At any time, a node could be S (susceptible), or E (exposed), or I (infectious), or R (recovered). Susceptible people are healthy but vulnerable and have a chance of getting infected by other infectious people. When an individual gets infected, his/her state is deterministically exposed until the incubation period of the virus ends, after which the individual becomes infectious. Here we assume that the individual with influenza virus can naturally generate antibodies and recover after an infection period. Then they will remain recovered for the rest of the time horizon.

To precisely describe the progress of the epidemic, we further expand E and I assuming that the incubation period is 2 days and infectious period is 5 days. As a result, the set of possible states of a node includes 9 values: $S = \{$*susceptible, exposed-1, exposed-2, infectious-1, infectious-2, infectious-3, infectious-4, infectious-5, recovered*$\}$. We also denote some subsets of S so that the following formulation is easier to understand: $S_{Sus} = \{$*susceptible*$\}$, $S_{Inf} = \{$*exposed-1, exposed-2, infectious-1, infectious-2, infectious-3, infectious-4, infectious-5*$\}$, and $S_R = \{$*recovered*$\}$.

The virus can only spread from an infectious node a to a susceptible node b with a probability p_{ab} if there is an edge (a, b) in the network. Therefore,

$$P_{ab}(v_a^t, v_b^t, v_b^{t+1}) = \begin{cases} p_{ab}, & \text{if } v_a^t \in S_{Inf}, v_b^t = \text{susceptible}, v_b^{t+1} = \text{exposed-1} \\ 1 - p_{ab}, & \text{if } v_a^t \in S_{Inf}, v_b^t = \text{susceptible}, v_b^{t+1} = \text{susceptible} \\ 0, & \text{o.w.} \end{cases}$$

$$P_a(v_a^t, v_a^{t+1}) = \begin{cases} 1, & \text{if } v_a^t = \text{susceptible}, v_a^{t+1} = \text{susceptible} \\ 1, & \text{if } v_a^t = \text{exposed-1}, v_a^{t+1} = \text{exposed-2} \\ 1, & \text{if } v_a^t = \text{exposed-2}, v_a^{t+1} = \text{infectious-1} \\ 1, & \text{if } v_a^t = \text{infectious-(d)}, v_a^{t+1} = \text{infectious(d+1)}, d = 1, \ldots, 4 \\ 1, & \text{if } v_a^t = \text{infectious-5}, v_a^{t+1} = \text{recovered} \\ 1, & \text{if } v_a^t = \text{recovered}, v_a^{t+1} = \text{recovered} \\ 0, & \text{o.w.} \end{cases}$$

Following the independent contagion model [35], the transmission events are independent.

There are two types of operations on an individual a for controlling the epidemic diffusion: vaccinations (V_a) and quarantines (Q_a), with costs of c_v and c_q respectively. Hence the operation set $\{O_i\} = \{V_a\} \bigcup \{Q_a\}$. A vaccination can only be applied to a susceptible node, and is effective with a probability of p_v. We also define W as the period of time that a vaccine takes to produce a good immune response. In other words, if a vaccination is effective, its effect still takes several days to be realized. Before the time window is over, a vaccine receiver still has a chance to be infected just as if there is no vaccine. A quarantine is only available to an infectious node, and is effective with a probability of p_q. We assume that p_q is close to 1 due to the fact that a quarantine isolates an individual so that there is little possibility to spread disease to others. Formally, we get the following transmission model.

$$T(v_a^t, V_a, v_a^{t+W}) = \begin{cases} p_v, & \text{if } v_a^t = \text{susceptible}, v_a^{t+1} = \text{recovered} \\ 1 - p_v, & \text{if } v_a^t = \text{susceptible}, v_a^{t+1} = v_a^t \\ 0, & \text{o.w.} \end{cases}$$

$$T(v_a^t, Q_a, v_a^{t+W}) = \begin{cases} p_q, & \text{if } v_a^t \in S_{Inf}, v_a^{t+1} = \text{recovered} \\ 1 - p_q, & \text{if } v_a^t \in S_{Inf}, v_a^{t+1} = v_a^t \\ 0, & \text{o.w.} \end{cases}$$

At any time point, the single-step measure here is the number of unhealthy individuals (either exposed or infectious), plus the total cost for vaccinations and quarantines, i.e. $S_G = \{recovered\}, T_G = \{0, \ldots, H\}$, and

OBJ $= E\left[\sum_{t=0}^{H} [\sum_{v_a^H = \text{recovered}} 1 - \sum_{V_a=1} c_v - \sum_{Q_a=1} c_q]\right]$.

4.5.1.2 Experimental results

To evaluate our approach in epidemic control problems, we randomly generate population networks of various sizes, connections and initially affected individuals. The Gurobi MIP Solver is used to solve an MIP if needed.

Performance of Dual Decomposition. We first compare the performance of the Gurobi solver with DD as they compute a decision of a lookahead policy by solving

the MIP. Here we just use a small network that contains 1,000 nodes and 10 initially affected individuals. The time horizon H here is 20 and vaccine effectiveness window W is 1.

Table 4.1: Objective Value and Computation Time

K	Gurobi-OBJ	DD-OBJ	Gurobi-Time (s)	DD-Time (s)
10	90.4	90.4	63.4	127.5
20	90.4	90.4	123.5	254.2
30	90.6	90.5	202.4	376.1
40	89.2	88.0	602.4	494.6
50	96.3	89.7	1038.6	634.8

From Table 4.1, we see how Gurobi and DD behave when computing Hindsight solutions as the number of futures K increases. On these problems, DD can returns near optimal solutions as its OBJ is equal to or very close to the OBJ of Gurobi. The computation time of DD scales linearly with the number of futures. When K is small, Gurobi takes less time than DD as the problem is small. For larger problems where $K > 30$, the time Gurobi needs is much longer and DD shows an advantage of computation complexity.

Performance of Adaptive Planning on A Small Network. For this small network, when 10 futures are used, both Gurobi and DD can return a lookahead solution within a decent time. We evaluate the adaptive control process for 10 times and average the reward of each run. Table 4.2 shows how each policy performs when the vaccine effectiveness window varies. Note that the numbers reported here give the accumulated number of infected population. A larger number indicates a worse control policy.

In addition to four lookahead policies, we also have a Random policy and HubBased policy. Random always takes L operations randomly, where L is a parameter that limiting the number of operations Random can take at each decision step. Similarly, HubBased also takes L operations per decision epoch. But it is smarter as it only conducts operations on top L nodes according to their number of connections. HubBased represents a group of heuristics that vaccinate popular nodes in some previous literatures.

Each row of the table shows the performance of each policy in the same setting. We see that Random, HubBased and GreedyZero are much worse than HNoop, Hindsight and Straightline. This is within our expectation since they use none or very little

information of the future diffusion realization. HNoop performs better than Hindsight though it is more myopic. By checking their decisions at every time step, we believe that HNoop usually takes many more actions, especially vaccinations, than Hindsight as HNoop thinks there would be no chance to take actions in later time steps. For epidemic control, such strategies with early actions are usually more effective. In contrast, Hindsight delays actions until they are necessary. But with a low infection rate (0.06), there is a high degree of randomness and the futures used by the lookahead policies are often very different. Hindsight will do nothing if one node is not infected in all the futures while in reality infection of this node might happen.

We change the value of vaccine effectiveness window and repeat the evaluation. When W is larger, a vaccine takes longer to be effective. For an individual, the chance of getting infected is higher. Thus more nodes get infected when Random and HubBased polices are used. However, lookahead policies, especially those with a long horizon of the futures, are able to see how W affects the action result. They are able to take actions earlier in order to effectively control the breakout. Therefore, HNoop, Hindsight and Straightline are able to keep the number of infections under control. GreedyZero only looks ahead towards the next time step. For $W > 1$, GreedyZero cannot see the benefit of vaccinations. Then it can perform even worse than Random and HubBased.

Table 4.2: The Number of Infection on A Small Network (Gurobi)

W	Random	HubBased	GreedyZero	HNoop	Hindsight	Straightline
1	796	680.2	744.5	158.9	269.9	191.9
3	806.3	736.8	842.4	186.7	238.3	167.1
5	843.1	750.4	850.6	161.1	247.5	165.9

Performance of Adaptive Planning on A Large Network. A large network is needed to push our evaluation towards an instance that is closer to reality. We create a population network with 10, 000 nodes and 10 initially infected nodes. The time horizon H is increased to 50. Since our lookahead policies cannot be computed by directly using a MIP solver, we apply DD to get lookahead solutions. For each policy, we still run 10 control simulations and report the average numbers of infection in Table 4.3. The result is consistent with the results on a small network.

Table 4.3: The Number of Infection on A Large Network (DD)

Random	HubBased	GreedyZero	HNoop	Hindsight	Straightline
1295.9	956.8	1056.8	583.7	707.2	558.2

4.5.2 Stochastic Firefighting

Now we present our exploration in stochastic firefighting problem. Unlike most previous work, we are particularly interested in dynamic strategies to prevent and suppress fire, which is more realistic.

4.5.2.1 Domain Settings

Similar to the settings in RCW domain, we divide an area into small patches, where a patch is an atomic land unit. At any time, a land patch a can be *susceptible*, *burning*, *burnt* or *protected*. A *Susceptible* patch is not on fire yet but is vulnerable to the fire from other patches. A *burning* patch a is on fire and the fire can spread to any *susceptible* neighbor b with a probability p_{ab}. If the fire on a *burning* patch is not suppressed, the patch will be *burnt* at the next time step. When a *susceptible* patch is successfully protected from catching fire or the fire on a *burning* patch is put out, the patch becomes *protected*. We assume that for the remaining time steps within the time horizon H, *burnt* patches remain *burnt* and protected patches remain protected. In addition, there are two types of actions: *protection* (P_a) and *suppression* (S_a), with a cost $c_p(a)$ and $c_s(a)$ respectively. Since the actions may not successfully protect the patch due to complicated fire conditions, we assign each action an effectiveness probability p_p and p_s. A standard encoding in our general framework is as follows.

$S = \{susceptible, burning, burnt, protected\}, S_{Sus} = \{susceptible\}, S_{Inf} = \{burning\}, S_R = \{protected, burnt\}, \{O_i\} = \{P_a\} \bigcup \{S_a\}, W = 1.$

$$P_{ab}(v_a^t, v_b^t, v_b^{t+1}) = \begin{cases} p_{ab}, & \text{if } v_a^t = \text{burning}, v_b^t = \text{susceptible}, v_b^{t+1} = \text{burning} \\ 1 - p_{ab}, & \text{if } v_a^t = \text{burning}, v_b^t = \text{susceptible}, v_b^{t+1} = \text{susceptible} \\ 0, & \text{o.w.} \end{cases}$$

$$P_a(v_a^t, v_a^{t+1}) = \begin{cases} 1, & \text{if } v_a^t = \text{susceptible}, v_a^{t+1} = \text{susceptible} \\ 1, & \text{if } v_a^t = \text{burning}, v_a^{t+1} = \text{burnt} \\ 1, & \text{if } v_a^t = \text{protected}, v_a^{t+1} = \text{protected} \\ 1, & \text{if } v_a^t = \text{burnt}, v_a^{t+1} = \text{burnt} \\ 0, & \text{o.w.} \end{cases}$$

$$T(v_a^t, P_a, v_a^{t+1}) = \begin{cases} p_p, & \text{if } v_a^t = \text{susceptible}, v_a^{t+1} = \text{protected} \\ 1 - p_p, & \text{if } v_a^t = \text{susceptible}, v_a^{t+1} = v_a^t \\ 0, & \text{o.w.} \end{cases}$$

$$T(v_a^t, S_a, v_a^{t+1}) = \begin{cases} p_s, & \text{if } v_a^t = \text{burning}, v_a^{t+1} = \text{protected} \\ 1 - p_s, & \text{if } v_a^t = \text{burning}, v_a^{t+1} = v_a^t \\ 0, & \text{o.w.} \end{cases}$$

The ultimate goal in this domain is to minimize the total number of burnt patches. It is equivalent to have $S_G = \{burnt\}, T_G = \{H\}$ and OBJ $= E[\sum_{v_a^H = \text{burnt}} 1]$.

Diffusion Model on Synthetic Maps. In our empirical study, we create grid maps as our study areas (Figure 4.1). Each cell represents a patch. Red cells are initially *burning*. Green and yellow cells are *susceptible*. Yellow ones have more fuel than the green hence yellow cells are more likely to catch fire. Adapted from Karafyllidis et al. [33], our fire spreading model calculates the spread probability p_{ab} by considering fuel level (amount of flammable material) of patch b, distance between a and b, their height difference and the real-time weather condition. This model is designed based on known knowledge of fire spreading. In particular, the more fuel an area contains, the faster the fire will spread. Fire may jump over some distance but mainly spreads to nearby patches. Moreover, it is known that fire shows a higher spreading rate when climbing up an upward slope. Weather is a very important factor that affects how the fire spreads, for which we include temperature and wind. Either a higher temperature or faster wind will lead to a faster fire spreading. The wind direction also affects fire spreading rate in different directions. For instance, when other factors are the same and the wind is blowing from South to North, the fire tends to move faster towards North than South. Formally, p_{ab} is computed by the following equation.

$$p_{ab}(t) = \text{fuel}(b) \cdot \text{TMP}(t) \cdot \frac{1}{\text{Dist}(a,b)} \cdot \text{Wind}(a,b,t) \cdot \text{Height}(a,b) \tag{4.9}$$

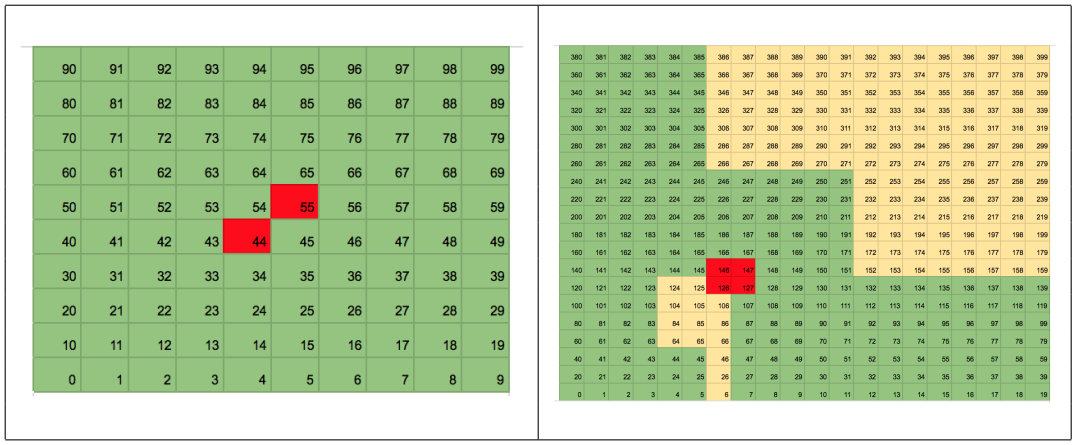

Figure 4.1: Grid maps of Firefighting. Each cell represents a land patch. Red cells are initially *burning*. Green and yellow cells are *susceptible*. Yellow ones have more fuel than the green hence yellow cells are more likely to catch fire.

In our calculation of the fire spreading model, fuel(a) is a random number in [0, 1]. TMP(t) is set to 1 for simplicity but is allowed to be a variable. Wind(a, b, t) is a 8-dimension vector that specifies its influence in each direction. For example, if the wind blows from South to North with a high speed and patch a is on fire, then the influence vector $\langle 1.0, 1.0, 0.8, 1.3, 1, 1.1, 1, 1.1\rangle$ indicates the value of Wind(a, b, t) if any b is in the *West*, *East*, *South*, *North*, *SW*, *NW*, *SE*, *NE* of a respectively. The influence vector of a slower wind could be $\langle 1.0, 1.0, 0.9, 1.1, 1, 1.04, 1, 1.04\rangle$.

4.5.2.2 Experimental results

We conduct experiments for this Firefighting domain on grid maps with different sizes and a time horizon $H = 20$. The wind blows from South to North during the whole period, but with a high speed in the first 10 time steps and becomes slower after $t = 10$. Whenever an MIP needs to be solved directly, we apply the Gurobi MIP solver to get a solution. All the experiments are run on a single machine with 8GB memory.

Performance of MIP and DD. We first test the performance of MIP and DD on a smaller map in Figure 4.1 and present the objective value and computation time in Table 4.4. In all the test cases here, DD always returns optimal solutions, as its objective value is always the same as Gurobi's. As the problem gets larger, DD has more computation time savings than Gurobi. It is worth noting that from our observation, most times DD needs a lot of iterations to reach convergence. But it is able to find the best feasible solution within a few iterations. Hence there could be more time savings by limiting the number of iterations to a small number without hurting the quality of the DD solution.

Table 4.4: Objective Value and Computation Time

K	Gurobi-OBJ	DD-OBJ	Gurobi-Time (s)	DD-Time (s)
10	3.3	3.3	45.0	70.4
20	2.9	2.9	78.6	123.1
30	3.5	3.5	169.1	210.7
40	3.1	3.1	293.6	257.8
50	3.5	3.5	2231.9	352.3

Evaluation of Adaptive Planning on A Small Problem. Now we report the rewards of our policies that are computed by Gurobi and DD. We still use the small grid map with 100 cells. Given that Gurobi and DD perform similarly, we only present the result of Gurobi in Table 4.5. The reward here is the number of burnt cells in the end of the time horizon.

Compared to Random policy, all the lookahead policies keep many more cells from being burnt, though GreedyZero performs worse than other lookahead policies. Hindsight and Straightline outperform HNoop, showing the benefit of considering future actions.

We further find that with our experimental settings, the budget of one time step is not enough to suppress the fire completely. Therefore, it is necessary and reasonable to also consider future budget for making a better decision for the current time step.

It is seen that Hindsight and Straightline make better decisions with 20 futures compared to 10 futures, yet such increase of futures does not help GreedyZero and HNoop attain more reward. After a closer examination of their solutions and the domain itself, we believe that Hindsight and Straightline need more futures for a good estimation of the candidate actions for the following reasons. First, the fire spreading events in this firefighting domain are highly stochastic. In addition, different from other policies,

Hindsight and Straightline are also required to make decisions for time steps after 0. They can benefit from more futures to see the real distribution of the diffusion.

Table 4.5: Rewards of MIP

K	Random	GreedyZero	HNoop	Hindsight	Straightline
10	72.6	41.3	27.6	20.2	21.1
20	72.6	42.8	27.5	8.4	11.8

Adaptive Planning on A Large Problem. When the map gets larger, direct solution of an MIP using a generic MIP solver becomes impractical. Here for the adaptive planning on a large map (Right of Figure 4.1), we only evaluate the performance of policies based on DD optimizations. We use 20 futures and set the number of iterations to 10 for computing lookahead policies. The results, presented in Table 4.6, show that lookahead policies largely outperform the Random policy, and the best policies are Hindsight and Straightline.

Table 4.6: Rewards of MIP

Random	GreedyZero	HNoop	Hindsight	Straightline
346.5	101.3	57.2	34.5	31.8

4.6 Summary

We address a class of probabilistic planning problems that involves controlling of diffusion process in networks. We formally set up the problem and propose a framework for solving it via lookahead policies. With our framework, lookahead decisions can be directly computed by using off-the-shelf MIP optimizers when the problem is small. For large problems that MIP solvers cannot solve in reasonable time, we provide a dual decomposition algorithm for approximating the lookahead solutions. Thus our work can also be seen as an alternative implementation of computing lookahead solutions efficiently. We study and apply our approach to two important domains with diffusion process: epidemic control of Influenza and stochastic firefighting problem. Experimental results confirm that dual decomposition returns near-optimal solutions within decent time. In addition, our lookahead policies outperform baselines in each domain and HNoop, Hindsight and Straightline are often the best policies in each application.

Chapter 5: Lessons Learned and Future Work

In this dissertation, we present our work on planning in diffusion networks. We first proposed a non-adaptive policy in conservation planning, where the goal is to encourage species spread in the long term. Given a set of control operations of interest, this policy explores the trade-off between population loss and policy flexibility. We also develop a fully adaptive approach for this conservation planning problem by computing a Hindsight Optimization (HOP) solution at every time step. Our algorithm scales for exponentially large, factored action spaces, and returns near-optimal decisions. Moreover, we generally address the diffusion network control problem via a single framework that incorporates multiple lookahead policies. Evaluations of our approach in several domains show that we can effectively compute high-quality policies. In this Chapter, we summarize a number of lessons we learned and discuss a few directions for future study.

5.1 Lessons Learned

Model-based vs. simulation-based lookahead policies. Our lookahead policies are computed via sampling futures, which in fact represent knowledge of a domain specific model. Simulation-based policies, on the other hand, only have access to a simulator for sampling trajectories, which are sequences of states and actions. A key point is that a future is much more informative than a trajectory. A trajectory can be obtained by evaluating a single policy on a future. But a future is able to incorporate any policy.

From another point of view, there is a trade-off between knowledge representation and sampling complexity. In Monte-Carlo tree search, only a numeric reward of a trajectory is used and overall a large amount of sampling is needed. Yet in our experiments, a small number of futures is enough for returning near optimal solutions (e.g. we use just 10 futures in our online conservation planning).

Size of futures. As previously mentioned, to realize a future, we need to remove all the randomness in the diffusion process. Following our formulation, randomness can be caused by diffusion events, local events and the operation model, which all together

decide the scale of a future. In our study, the probabilistic models are comparatively simple. However, if the probabilistic model has large CPTs, a future may not be scalable. For example, imagine a case where the effectiveness probability of an operation is given by a complete CPT on the state of the network, to realize a future, an exponential number of coin flips are needed since the state space is exponential.

It is also critical to efficiently record the outcomes of sampling all the stochastics at every time step. This requires a careful design of the representation of a future.

MIP encoding. The model knowledge and optimization goal are represented as a single MIP. While in many cases constructing MIP formulations is relatively straightforward, the resulting MIPs can be too large or too weak to be solved by state-of-art solvers. Therefore, it is necessary to optimize a MIP formulation when the problem scale is huge.

In Appendix A, we give examples of our MIP formulations. The MIP in conservation planning is simplified by having a binary variable for each node, meaning the node is occupied or not. The state of *conserved* is expressed via the state constraint that a node is susceptible only when it is in a purchased parcel. The diffusion event is described concisely. A node value would be 1 *iff* at least one occupied neighbor spreads successfully.

Experience with MIP solvers. MIP solvers are required in our approach and we have been using the IBM CPLEX and Gurobi optimizers. Both are state-of-art MIP optimizers that efficiently compute optimal solutions when the problem size is not too huge. Here we share two tricks of using CPLEX and Gurobi for implementing our dual decomposition algorithm.

At every iteration of the dual decomposition algorithm (2), after optimizing subproblems on each future and extracting a feasible solution, we need to re-compute the subproblems with the feasible solution to get APX (Equation 3.3). This recomputation can be done efficiently by maintaining each subproblem in memory and assigning the operation variable with the feasible solution value. Then the solver can start the recomputation from the previous optimal point.

When an MIP problem is too large to get an optimal solution within a reasonable time, we can get a suboptimal and feasible solution instead. The first method is to set a time limit to the computation process and ask for the newest feasible solution found. The other way is to iteratively track the new feasible solution found during the computation and pick the best one in the end.

5.2 Future Work

5.2.1 Scheduling Conservation Designs for Maximum Flexibility

In future work, we plan to consider several improvements to the primal-dual algorithm. Currently, at each iteration, the algorithm randomly picks an unconnected terminal to grow a path from. It is likely that more intelligent selection mechanisms can improve the overall results. We are also interested in developing a primal-dual algorithm that directly incorporates the error tolerance constraint of our early-stopping approach. This would provide a more direct method for trading off reward for improved flexibility. Furthermore, we intend to pursue fully adaptive approaches to this and other conservation problems. One idea is to incorporate our scheduling approach into a replanning algorithm that selects purchases for the current decision epoch based on the most up-to-date information. In particular, at each decision epoch a schedule would be formed and those parcels scheduled to be purchased immediately (those with no flexibility) would be purchased or a subset of those in cases where the immediate budget would be exceeded. Considering more sophisticated approaches that take into account the immediate budget would be a natural and useful extension. It would also be interesting to consider the conservation problem with other variants of the reward function. For some species, rather than caring only the population in the end, the ecological goal may value the spread during the whole period the same or even more. Presumably such models would have different properties and complexities from the one we study in this paper.

5.2.2 Online Planning in Diffusion Networks

Although we treat the various lookahead policies as competitors, it is possible to create a policy-switching planner that combines all of the lookahead policies. At every decision epoch, the new planner evaluates all the four policies for a good action and then picks the best decision.

Another future direction is to extend our framework into a generic planner for solving probabilistic planning problems with exponential state and action spaces. The main difficulty is to encode the problem in a MIP compactly and generally. For example, RDDL [49] is a uniform planner language for representing factored MDPs in dynamic Bayesian net (DBN). Translations of a problem's CPT are available with the RDDL

system. Unfortunately, a direct translation is represented via structures like decision diagrams (trees), which could be exponentially large due to redundancy in the paths. It is more beneficial to use a set of compact rules, but they are usually domain specific.

Bibliography

[1] K. Ahmadizadeh, C. Dilkina, C. P. Gomes, and A. Sabharwal. An empirical study of optimization for maximizing diffusion in networks. In *16th International Conference on Principles and Practice of Constraint Programming*, 2010.

[2] Elliot Anshelevich, Deeparnab Chakrabarty, Ameya Hate, and Chaitanya Swamy. Approximation algorithms for the firefighter problem: Cuts over time and submodularity. In *Algorithms and Computation*, pages 974–983. Springer, 2009.

[3] James Aspnes, Kevin Chang, and Aleksandr Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 43–52. Society for Industrial and Applied Mathematics, 2005.

[4] Norman T. Bailey. *The Matthematical Theory of Infectious Diseases and its Applications.* Hafner Press, 1975.

[5] Andrew G Barto, Steven J Bradtke, and Satinder P Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

[6] R. Bent and P. V. Hentenryck. Regret only! Online stochastic optimization under time constraints. In *AAAI '04: Proceedings of the Nineteenth Conference on Artificial Intelligence*, 2004.

[7] Blai Bonet and Hector Geffner. Action selection for MDPs: Anytime AO* vs. UCT. In *AAAI Conference on Artificial Intelligence*, pages 1749–1755. Citeseer, 2012.

[8] Linda Briesemeister, Patrick Lincoln, and Phillip Porras. Epidemic profiles and defense of scale-free networks. *2003 ACM workshop on Rapid malcode*, 12 2003.

[9] Deepayan Chakrabarti, Yang Wang, Chenxi Wang, Jurij Leskovec, and Christos Faloutsos. Epidemic thresholds in real networks. *ACM Transactions on Information and System Security*, 10, 2008.

[10] H. S. Chang, R. L. Givan, and Chong E. K.P. On-line scheduling via sampling. In *AIPS '00: Artificial Intelligence Planning and Scheduling*, 2011.

[11] M. Charikar, C. Chekuri, T. Cheung, A. Dai, S. Guha, and M. Li. Approximation algorithm for directed Steiner Tree problems. In *The Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[12] E. K.P. Chong, R. L. Givan, and H. S. Chang. A framework for simulation-based network control via hindsight optimization. In *IEEE Conference on Decision and Control (IEEE CDC)*, 2000.

[13] Reuven Cohen, Shlomo Havlin, and Daniel ben Avraham. Efficient immunization strategies for computer networks and populations. *Physical review letters*, 91, 2004.

[14] M. Crowley and D. Poole. Policy gradient planning for environmental decision making with existing simulators. In *Twenty-fifth AAAI Conference on Artificial Intelligence*, 2011.

[15] Mark Crowley, John Nelson, and David Poole. Seeing the forest despite the trees: Large scale spatial-temporal decision making. 2009.

[16] Zoltán Dezső and Albert Barabási. Halting viruses in scale-free networks. In *Physical Review E*. APS, 2002.

[17] Klaus Dietz. Transmission and control of arbovirus diseases. In *Society for Industrial and Applied Mathematics (SIAM)*, pages 104–121, 1975.

[18] L. M.A. Drummond and M. Santos. A distributed dual ascent algorithm for Steiner problems in multicast routing. *Networks*, 53:170–183, 2009.

[19] Víctor M. Eguíluz and Konstantin Klemm. Epidemic threshold in structured scale-free networks. *Physical Review*, 89, 2002.

[20] Roee Engelberg, Jochen Könemann, Stefano Leonardi, and Joseph Seffi Naor. Cut problems in graphs with a budget constraint. In *Journal of Discrete Algorithms*. Elsevier, 2006.

[21] Stephen Finbow, Andrew King, Gary MacGillivray, and Romeo Rizzi. The firefighter problem for graphs of maximum degree three. In *Journal of Discrete Algorithms*. Elsevier, 2007.

[22] Stephen Finbow and Gary MacGillivray. The firefighter problem: A survey of results, directions and questions. In *Journal of Australas. J. Combin*, pages 57–77, 2009.

[23] Sylvain Gelly and Yizao Wang. Exploration exploitation in Go. In *In Twentieth Annual Conference on Neural Information Processing Systems (NIPS)*, 2006.

[24] Noam Goldberg, Sven Leyffer, and Ilya Safro. Optimal response to epidemics and cyber attacks in networks. *Networks*, 66, 06 2015.

[25] D. Golovin, A. Krause, B. Gardner, S. J. Converse, and S. Morey. Dynamic resource allocation in conservation planning. In *Twenty-fifth AAAI Conference on Artificial Intelligence*, 2011.

[26] I. Hanski and O. Ovaskainen. The metapopulation capacity of a fragmented landscape. *Nature*, 404(6779):755–758, 2000.

[27] Bert Hartnell. Firefighter! An application of domination. In *Presentation, 25th Manitoba Conference on Combinatorial Mathematics and Computing, University of Manitoba in Winnipeg, Canada*, 1995.

[28] Ara Hayrapetyan, David Kempe, Martin Pál, and Zoya Svitkina. Unbalanced graph cuts. In *Algorithms– ESA 2005*, pages 191–202. Springer, 2005.

[29] Herbert W. Hethcote. The mathematics of infectious diseases. In *Society for Industrial and Applied Mathematics (SIAM)*, pages 599–653, May 2000.

[30] Rachel M. Houtman, Claire A. Montgomery, Aaron R. Gagnon, David E. Calkin, Thomas G. Dietterich, Sean McGregor, and Mark Crowley. Allowing a wildfire to burn: Estimating the effect on future fire suppression costs. *International Journal of Wildland Fire*, 22:871–882, 2013.

[31] A. Hubbe, W. Ruml, S. Yoon, J. Benton, and M. B. Do. On-line anticipatory planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

[32] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*. Springer, 1992.

[33] Ioannis Karafyllidis and Adonios Thanailakis. A model for predicting forest fire spreading using cellular automata. *Ecological Modelling*, 99:89–97, 1997.

[34] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[35] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. In *Royal Society of London. Series A*, volume 115, pages 700–721, 1927.

[36] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

[37] A. Kumar, X. Wu, and S. Zilberstein. Lagrangian relaxation techniques for scalable spatial conservation planning. In *AAAI '12: Proceedings of the Twenty-sixth Conference on Artificial Intelligence*, 2012.

[38] Nilly Madar, Tomer Kalisky, Reuven Cohen, Daniel ben Avraham, and Shlomo Havlin. Immunization and epidemic dynamics in complex networks. *Physics of Condensed Matter*, 38:269–276, 2004.

[39] AG McKendrick. Applications of mathematics to medical problems. *Proceedings of the Edinburgh Mathematical Society*, 44, 1925.

[40] L. Mercier and P. V. Hentenryck. Performance analysis of online anticipatory algorithms for large multistage stochastic integer programs. In *IJCAI '07: International Joint Conference on Artificial Intelligence*, 2007.

[41] Mark EJ Newman, I. Jensen, and RM Ziff. Percolation and epidemics in a two-dimensional small world. In *Physical Review E*. APS, 2002.

[42] Nils J Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann, 2014.

[43] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. In *Physical Review Letters*. APS, 2001.

[44] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic dynamics in finite size scale-free networks. *Physical Review*, 65, 2002.

[45] Jervis Pinto and Alan Fern. Learning partial policies to speedup MDP tree search. 2014.

[46] Warren B. Powell. Merging AI and OR to solve high-dimensional stochastic optimization problems using approximate dynamic programming. *INFORMS Journal on Computing*, 22:2–17, 2010.

[47] Aswin Raghavan, Roni Khardon, Alan Fern, and Prasad Tadepalli. Symbolic opportunistic policy iteration for factored-action MDPs. In *Advances in Neural Information Processing Systems*, pages 2499–2507, 2013.

[48] Robert M. May Roy M. Anderson, B. Anderson. *Infectious Diseases of Humans: Dynamics and Control*. Oxford, 1991.

[49] Scott Sanner. Relational dynamic influence diagram language (RDDL): Language description. *Unpublished ms. Australian National University*, 2010.

[50] A. Shapiro. Monte Carlo sampling methods. In *Stochastic Programming, Handbooks in Operations Research and Management Science*, volume 10, pages 353–426. 2003.

[51] D. Sheldon, B. Dilkina, A. Elmachtoub, R. Finseth, A. Sabharwal, J. Conrad, C. Gomes, D. Shmoys, W. Allen, O. Amundsen, and B. Vaughan. Maximizing the spread of cascades using network design. In *Uncertainty in Artificial Intelligence (UAI)*, 2010.

[52] Gwen Spencer. Robust cuts over time: Combatting the spread of invasive species with unreliable biological control. In *Twenty-sixty AAAI Conference on Artificial Intelligence*, 2012.

[53] Andrew L. Sullivan. Wildland surface fire spread modelling, 1990–2007. 3: Simulation and mathematical analogue models. *International Journal of Wildland Fire*, 18:387–403, 2009.

[54] Alan Tsang, Kate Larson, and Rob McAlpine. Resource sharing for control of wildland fires. In *AAAI '13: Proceedings of the Twenty-third Conference on Artificial Intelligence*, 2013.

[55] V. V. Vazirani. *Approximation Algorithms*. Springer, Berlin, 2001.

[56] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small world" networks. In *Nature*. Nature Publishing Group, 1998.

[57] R. T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28:271–287, 1984.

[58] G. Wu, E. Chong, and R. Givan. Burst-level congestion control using hindsight optimization. *IEEE Transactions on Automatic Control*, 47:979–991, 2002.

[59] S. Yoon, A. Fern, R. L. Givan, and S. Kambhampati. Probabilistic planning via determinization in hindsight. In *AAAI '08: Proceedings of the Twenty-third Conference on Artificial Intelligence*, 2008.

[60] S. Yoon, W. Ruml, J. Benton, and M. B. Do. Improving determinization in hindsight for online probabilistic planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2010.

APPENDICES

Appendix A: MIP Encoding

For reference, we include our formulations of Equation 4.6 in each application in this appendix.

A.1 Conservation Planning

Let $v_a^{k,t}$ be a state variable for each node, which is binary indicating node a is occupied or not in future k at time t, binary $P_i^{k,t}$ be operation variable for each parcel, meaning if parcel i is purchased or not in future k at time t, $\bar{P}_i^t$ be the consistent operation variable at time 0, and B(t) be the budget at every time step.

$$\min -\frac{1}{K}\sum_{k=1}^{K}\sum_{a\in V} v_a^{k,H} \tag{A.1}$$

$$\text{initialization } v_a^{k,0} = 0/1, \forall k, a$$

$$\text{state constraint } v_a^{k,t} \leq \sum_{a\in i}\sum_{t'=0}^{t} P_i^{k,t'}, \forall k, t, a$$

$$\text{purchase constraint } \sum_{t=0}^{H-1} P_i^{k,t} \leq 1, \forall k, i$$

$$\text{diffusion } v_b^{k,t+1} \leq \sum_{(a,b)\in f_k} v_a^{k,t}, \forall k, t, b$$

$$\text{consistent first action } P_i^{k,t} = \bar{P}_i^t, \forall k, i, t = 0,$$

$$\text{budget constraint } \sum_i P_i^{k,t} \times c_i \leq B(t), \forall k, t$$

A.2 Epidemic Control of Influenza

Here the node state is a value from set $S = \{$*susceptible, exposed-1, exposed-2, infectious-1, infectious-2, infectious-3, infectious-4, infectious-5, recovered*$\}$. We use 8 bits to represent the state of node a in future k at time t, i.e. binary variables $v_{a,d}^{k,t}, d = 0, \ldots, 8$, each corresponding to a state value in S and we allow only one bit is 1. $V_a^{k,t}$ and $Q_a^{k,t}$ are vaccinations and quarantines respectively. $\bar{V}_a^t$ and $\bar{Q}_a^t$ are the consistent operation variable at $t = 0$.

$$\min \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T}\sum_{a\in V}[(v_{a,1}^{k,t}+v_{a,2}^{k,t}+\cdots+v_{a,7}^{k,t}) \tag{A.2}$$

$$+(V_a^{k,t}\times c_v+Q_a^{k,t}\times c_q)]$$

$$\text{initialization } v_{a,d}^{k,0}=0/1,\forall k,a,d$$

$$\text{state constraint } \sum_{d=0}^{d=8} v_{a,d}^{k,t}=1,\forall k,t,a$$

$$\text{vaccination pre-constraint } V_a^{k,t}\le v_{a,0}^{k,t},\forall k,t,a$$

$$\text{quarantine pre-constraint } Q_a^{k,t}\le \sum_{d=3}^{7} v_{a,d}^{k,t},\forall k,t,a$$

$$\text{successful action result } v_{a,8}^{k,t+1}\ge V_a^{k,t},\forall k,t,a \text{ with a successful V flip}$$

$$\text{successful action result } v_{a,8}^{k,t+1}\ge Q_a^{k,t},\forall k,t,a \text{ with a successful Q flip}$$

$$\text{diffusion w/t actions } v_{a,8}^{k,t+1}\ge v_{a,7}^{k,t}+v_{a,8}^{k,t},\forall k,a,t<T-1$$

$$\text{infection } u_a^{k,t}\le \sum_{d=3}^{8} v_{a,d}^{k,t},\forall k,t,a$$

$$\text{infection } u_a^{k,t}+Q_a^{k,t}\le 0,\forall k,t,a$$

$$\text{infection } u_a^{k,t}-\sum_{d=3}^{8} v_{a,d}^{k,t}+Q_a^{k,t}\ge 0,\forall k,t,a$$

$$\text{infection } z_a^{k,t+1}\ge u_b^{k,t},\forall k,t,a,b \text{ is a parent of } a$$

$$\text{infection } z_a^{k,t+1}\le \sum_{b \text{ is a parent of } a} u_b^{k,t},\forall k,t,a$$

$$\text{diffusion w/t actions } V_a^{k,t}+Q_a^{k,t}-v_{a,0}^{k,t}-z_a^{k,t+1}+v_{a,1}^{k,t+1}+1\ge 0,\forall k,t,a,$$

$$\text{diffusion w/t actions } V_a^{k,t}+Q_a^{k,t}-v_{a,0}^{k,t}+z_a^{k,t+1}+v_{a,1}^{k,t+1}\ge 0,\forall k,t,a,$$

$$\text{state change w/t actions } V_a^{k,t}+Q_a^{k,t}+v_{a,0}^{k,t}+v_{a,d}^{k,t}-v_{a,d+1}^{k,t+1}\ge 0,\forall k,t,a,d=1,2,\ldots,7$$

$$\text{state change w/t actions } V_a^{k,t}+Q_a^{k,t}+v_{a,0}^{k,t}-v_{a,d}^{k,t}+v_{a,d+1}^{k,t+1}\ge 0,\forall k,t,a,d=1,2,\ldots,7$$

$$\text{failed action result } 1-Q_a^{k,t}+v_{a,d}^{k,t}-v_{a,d+1}^{k,t+1}\ge 0,\forall k,t,a \text{ with a failed Q flip},d=3,4,\ldots,7$$

$$\text{failed action result } 1-Q_a^{k,t}-v_{a,d}^{k,t}+v_{a,d+1}^{k,t+1}\ge 0,\forall k,t,a \text{ with a failed Q flip},d=3,4,\ldots,7$$

$$\text{diffusion w/t actions } 1-V_a^{k,t}-z_a^{k,t+1}+v_{a,1}^{k,t+1}\ge 0,\forall k,t,a \text{ with a failed V flip},$$

$$\text{diffusion w/t actions } -V_a^{k,t}+z_a^{k,t+1}+v_{a,0}^{k,t+1}\ge 0,\forall k,t,a \text{ with a failed V flip},$$

$$\text{consistent first action } V_a^{k,t}=\bar{V}_a^t,\forall k,a,t=0,$$

$$\text{consistent first action } Q_a^{k,t}=\bar{Q}_a^t,\forall k,a,t=0$$

A.3 Stochastic Firefighting

We use 4 bits to represent the state of node a in future k at time t, i.e. binary variables $v_{a,d}^{k,t}, d = 0, \ldots, 3$, each corresponding to a state value in $S = \{$*susceptible, burning, burnt, protected*$\}$. For each node, only one state variable is 1. $P_a^{k,t}$ and $S_a^{k,t}$ are protections and suppressions respectively. $\bar{P}_a^t$ and $\bar{S}_a^t$ are the consistent operation variable at $t = 0$.

$$\min \frac{1}{K} \sum_{k=1}^{K} \sum_{a \in V} v_{a,2}^{k,T} \tag{A.3}$$

initialization $v_{a,d}^{k,0} = 0/1, \forall k, a, d$

state constraint $\sum_{d=0}^{d=3} v_{a,d}^{k,t} = 1, \forall k, t, a$

protection pre-constraint $P_a^{k,t} \leq v_{a,0}^{k,t}, \forall k, t, a$

suppression pre-constraint $S_a^{k,t} \leq v_{a,1}^{k,t}, \forall k, t, a$

successful action result $v_{a,3}^{k,t+1} \geq P_a^{k,t}, \forall k, t, a$ with a successful P flip

successful action result $v_{a,3}^{k,t+1} \geq S_a^{k,t}, \forall k, t, a$ with a successful S flip

infection $u_a^{k,t} \leq v_{a,1}^{k,t}, \forall k, t, a$

infection $u_a^{k,t} + S_a^{k,t} \leq 1, \forall k, t, a$

infection $u_a^{k,t} - v_{a,1}^{k,t} + S_a^{k,t} \geq 0, \forall k, t, a$

infection $z_a^{k,t+1} \geq u_b^{k,t}, \forall k, t, a, b$ is a parent of a

infection $z_a^{k,t+1} \leq \sum_{b \text{ is a parent of} a} u_b^{k,t}, \forall k, t, a$

diffusion w/t actions $P_a^{k,t} + S_a^{k,t} - v_{a,0}^{k,t} - z_a^{k,t+1} + v_{a,1}^{k,t+1} + 1 \geq 0, \forall k, t, a$

diffusion w/t actions $P_a^{k,t} + S_a^{k,t} - v_{a,0}^{k,t} + z_a^{k,t+1} + v_{a,1}^{k,t+1} \geq 0, \forall k, t, a$

state change w/t actions $v_{a,2}^{k,t+1} - v_{a,2}^{k,t} \geq 0, \forall k, t, a,$

state change w/t actions $v_{a,3}^{k,t+1} + v_{a,3}^{k,t} \geq 0, \forall k, t, a,$

state change w/t actions $S_a^{k,t} - v_{a,1}^{k,t} + v_{a,2}^{k,t+1} \geq 0, \forall k, t, a,$

failed action result $S_a^{k,t} - v_{a,2}^{k,t+1} \leq 0, \forall k, t, a$ with a failed S flip,

failed action result $1 - V_a^{k,t} - z_a^{k,t+1} + v_{a,1}^{k,t+1} \geq 0, \forall k, t, a$ with a failed P flip

failed action result $-V_a^{k,t} + z_a^{k,t+1} + v_{a,0}^{k,t+1} \geq 0, \forall k, t, a$ with a failed P flip

consistent first action $P_a^{k,t} = \bar{P}_a^t, \forall k, a, t = 0$

consistent first action $S_a^{k,t} = \bar{S}_a^t, \forall k, a, t = 0$

budget constraint $\sum_{a \in V} P_a^{k,t} \times c_p(a) + S_a^{k,t} \times c_s(a) \leq B(t), \forall k, t$