International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 10 Issue 8 ||August 2022 ||PP. 322-482

Efficient and Scalable Algorithms for Control, Filtering, Learning, and Coordination in Large-Scale Models of Graph-Based Markov Decision Processes: Applications to Anonymous Influence, Cooperative Multi-Agent Optimization, Disease Epidemics, Social Networks, and Forest Wildfires

Junxia Deng
University of Southern California
California, USA

Ceil Hong. Zhang
Massachusetts Institute of Technology
Massachusetts, USA
zhanghongceil@pku.org.cn

Disclaimer:
The content of the paper belongs to the original author and is only used as a handout for the Advanced Intensive Learning course.

Abstract

This thesis derives a series of algorithms to enable the use of a class of structured models, known as graph-based Markov decision processes (GMDPs), for applications involving a collection of interacting processes. Many large-scale dynamic processes of recent interest are described by GMDPs, including disease epidemics, forest wildfires, robot swarms, and social networks. For the discrete time and discrete space graph-based models we consider in this thesis, each vertex in the graph corresponds to a standard Markov decision process (MDP) and edges between vertices correspond to the coupling interactions which influence the state transitions of the MDP. For example, in a forest wildfire, each vertex corresponds to a tree and edges define the surrounding trees which influence the probability of a tree catching on fire. Similarly, in a disease epidemic, each vertex refers to a community and edges describe the surrounding communities which influence the probability of an outbreak occurring based on their infection level. In addition, we consider a property common in large-scale processes called "Anonymous Influence." Simply stated, a GMDP contains Anonymous Influence if the state transitions of the individual MDPs relies on the number of influencing MDPs in particular states, and not the identity of these MDPs. A consequence of using GMDPs is that the equivalent standard MDP representation typically has a significantly large state space, observation space (for the filtering problem), and feasible action space (for the optimal control problem). Therefore, a major objective of this thesis is to derive suitable algorithms that are efficient and scalable for arbitrarily large models. Furthermore, we address the necessary aspects of using GMDPs as a modeling framework: learning model parameters from time series data, optimally allocating a limited amount of control resources, producing state estimates online given uncertain measurements, and interacting with a process modeled by a GMDP using a team of cooperative autonomous agents.

We begin by considering the problem of optimally allocating a limited amount of control resources when the underlying model state is fully observable. We leverage approximate dynamic programming methods based on linear programs to circumvent enumerating the state and action spaces, which allows us to build a scalable framework for producing approximate value and state-action function approximations. From these functions, a constrained policy can be derived which strictly enforces a capacity constraint. We also propose a novel control framework based on bond percolation on a lattice to more easily address potentially heterogeneous properties in GMDP models. For both of

these approaches, we develop analysis techniques to evaluate the quality of our approximations and to determine if the process is controllable. We evaluate our control techniques on simulations of forest wildfires and disease epidemics, and show that they are effective on considerably large models and outperform comparable methods.

Next, we relax our assumption of a fully observable state and consider the problem of producing state estimates online given only noisy measurements. Here, we leverage techniques from variational inference to derive an approximately optimal message passing scheme which is similar in spirit to belief propagation methods, and prove that our approach optimizes the evidence lower bound. We show that our filtering scheme is at least as accurate as other methods while requiring two orders of magnitude less time to produce an estimate. We further validate the need for a fast and accurate online filter by developing a certainty-equivalence framework to enable our control methods in applications with state uncertainty. We show that this framework is able to achieve comparable results to the case of a fully observable state with only a minimal increase in control effort.

We then turn to the problem of learning the necessary parameters of a GMDP using time series data, and relax the assumption that the model parameters are specified a priori. We use the Expectation-Maximization framework to derive a method that approximately optimizes the likelihood of the data with favorable complexity for large GMDPs. We use publicly available data, on the daily counts of Novel Coronavirus (COVID-19) in California by county and twitter interactions on a topic, to train GMDP models and show that they better explain the observed data compared to a completely independent process model assumption.

Lastly, we develop three agent-based frameworks based on a team of cooperative autonomous aerial vehicles which we apply to the example of forest wildfires. We relax various assumptions that we used in our control and state estimation methods. In the control problem, we no longer assume that control actions can be applied arbitrarily in the forest at each time step. Instead, agents must consider travel time and a limited view of the forest before deciding how to move to quickly extinguish a wildfire as a team. In the filtering problem, we introduce severe partial observability and communication constraints, and agents must coordinate meetings to share information and enable effective coverage of a wildfire. We also pose a general multi-agent cooperative optimization problem and discuss distributed techniques and conditions required to recover the optimal centralized solution. We use this class of optimization problems to motivate a more general approach than prior work to multi-agent coordination strategies.

All of our proposed algorithms have the feature of addressing efficiency and scalability to enable the use of potentially very large GMDPs. Our methods will enable GMDPs to be used for arbitrarily complex applications while still allowing for theoretical analysis and justification. Furthermore, while there are still avenues to investigate, our methods provide the groundwork for continuing to relax assumptions and address more general modeling cases. We believe our methods will spur further interest in leveraging GMDPs to address natural phenomena.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Structured models with significant representational power have become the tool of choice for environmental and relational modeling due to a wealth of available data. The ability to model large systems as a collection of interacting subsystems has spurred the development of algorithms to translate data sets into meaningful decision making strategies for a single autonomous agent as well as large teams of cooperative autonomous agents. Nevertheless, the development of algorithmic and analysis frameworks have lagged behind the direct application of structured models to robotics-related problems, which demonstrate the benefits and value of using these type of models. Furthermore, it has become apparent that a wide range of natural phenomena are well described by structured models, from large-scale dynamic spatial processes such as forest wildfires and disease epidemics, to information based processes such as social networks and computer viruses. Many of the connections between application domains have only become apparent recently due to the increasing popularity of data-driven modeling techniques.

The aim of this thesis is to address the lack of a complete suite of algorithms to enable the use of a class of structured models with discrete time and a discrete state space known as graph-based Markov decision processes (GMDPs). A GMDP consists of a collection of individual discrete time and discrete space Markov decision processes (MDPs), which are the subsystems that make up the larger overall system. A graph, consisting of a vertex set and an edge set, succinctly summarizes the structure of the system by describing the relationships between the individual MDPs. A unique identifier is assigned to each MDP, which forms the vertex set, and edges between vertices describe the coupling interactions in the state transitions of the MDPs. For example, in a forest wildfire, each tree is a MDP which is affected by neighboring trees. In a disease epidemic, each community is an MDP affected by neighboring communities, and in a social network, each user is an MDP influenced by other people in their network. Furthermore, we consider a common property in large graph-based models called "Anonymous Influence." A GMDP contains Anonymous Influence when the MDP state dynamics rely on the number of influencing neighbors in particular states, and not

the identity of these neighbors. Not only does Anonymous Influence allow us to significantly improve the computational complexity of our algorithms, it also allows us to propose and validate different model structures for different applications.

Graph-based Markov models have been studied in a variety of problem settings. Given a reward function and the assumption that the underlying state is fully observable, the problem is best described as a GMDP. The task is to solve the optimal control problem and the solution is a policy mapping states to actions. In the absence of control with only noisy measurements of the state available, the model is best described as a graph-based hidden Markov model (GHMM). In this case, the task is to solve the filtering problem by producing a posterior distribution over the state space given a history of measurements. In addition, we also consider the problem of learning model parameters from a time history of observations. Finally, considering a reward function and control actions along with state uncertainty, the problem is best described as a graph-based partially observable MDP (GPOMDP), for which the task is to determine a policy mapping the belief of the state to actions. For convenience, we generally refer to the model as a GMDP and specifically note the problem assumptions and objective when deriving our algorithms.

An equivalent, standard MDP description can be created from a given GMDP, where the state dynamics is the product space of all of the individual MDPs. Typically, this description results in a Markov model with significantly large state space, observation space (for the filtering problem), and feasible action space (for the control problem). In addition, the model structure does not translate to tractable exact solution methods that are analogous to traditional MDP methods. Therefore, we turn to established optimization techniques in literature to develop approximate methods to address different GMDP problems.

To more easily enable the use of GMDPs as a modeling framework, we develop the necessary algorithms for the traditional Markov model problem formulations: learning model parameters, producing constrained control policies, and estimating the underlying state given noisy measurements. Our algorithms focus on efficiency and scalability to address the potentially large GMDPs with arbitrary structure. We note that considering control constraints and model uncertainty are essential for modeling real phenomena. Without control constraints, the optimal unconstrained policy is often straightforward to specify. For a forest wildfire, the optimal unconstrained policy is to provide fire retardant to every tree at every time step, and in a disease epidemic, the optimal unconstrained policy is to provide medical resources to every community at every time step. Therefore, the control problem is only meaningful when a constraint is applied to the total control effort available, as we consider in our control approaches in Chapter 3.

Model uncertainty shows up in two aspects in realistic processes. First, the underlying state is not directly observable and instead only noisy measurements at each time step are available. We address this problem in Chapter 4 and develop a fast and efficient online filter to produce accurate state estimates. A fast online filter is critical in enabling the use of our control schemes

with measurement uncertainty, as well as enabling cooperative multi-agent teams to reason about environmental uncertainty. Therefore, it is critical that an appropriate filtering method is developed for the GMDP framework. Second, model parameters cannot be known exactly a prior but data can be used to best fit a GMDP model. In Chapter 5, we develop a learning algorithm based on a time history of observations for each MDP in the GMDP, thereby enabling us to directly learn models from data sets.

All of our algorithms are based on established optimization techniques in literature. We formulate problems as an optimization statement to discuss the sources of computational complexity and intractably. This approach then enables us to consider different types of approximations and the implications of such approximations, which are necessary as exact methods are tractable only for trivial models. Furthermore, there is a significant amount of prior work on different optimization techniques, their computational complexity, and their solution quality. Therefore, we directly leverage this foundation to simplify the development of algorithms for GMDPs.

With a suite of effective algorithms for GMDPs in hand, we present frameworks based on cooperative multi-agent teams consisting of autonomous aerial vehicles in Chapter 6. Adding realistic agent models in combination with an environmental model represented by a GMDP results in a GPOMDP with additional control and measurement constraints. Due to the complexity of determining POMDP solutions, we turn to state-of-the-art techniques in multi-agent systems to derive effective and scalable algorithms. While there are a variety of multi-agent methods proposed in literature, many are formulated specifically for a given problem formulation. In contrast, we also consider a more general formulation of cooperative multi-agent problems to propose decentralized solution techniques. We use our frameworks to provide examples of how robotics applications can leverage GMDPs as a modeling framework, and we believe they provide a guide for developing future approaches involving autonomous agents.

It should be noted that the results presented in this thesis form an algorithmic basis to continue to develop and improve algorithms for GMDPs. Despite the extensive literature on structured models and Markov models, there is a noticeable gap in addressing models with an arbitrary coupling structure. While we make major strides in addressing the necessary problems for GMDPs, there are a number of clear avenues for future research, which we remark on at the end of this thesis. Next, we summarize current methods in literature for structured Markov models, note their shortcomings in regards to the models we consider in this thesis, and emphasize our contributions in relation to the relevant prior work.

1.1 Related Work

The popularity of Markov modeling frameworks has naturally led to the formulation of structured models with coupling interactions in the state dynamics [1, 2]. Some of the first approaches for

the control problem considered ideas of strong and weak coupling interactions [3, 4, 5], and exploiting interaction strength to develop suitable methods. Building on these ideas, structured solution methods were developed for models with large state spaces, not necessarily containing additional structure, including feature-based dynamic programming [6], approximate value trees [7], and factored representations [8]. Similarly, loopy belief propagation [9] and generalized belief propagation [10] were developed as structured approaches to the inference problem for models with large state and measurement spaces. While models with structure were not necessarily part of the problem formulation in these works, the solution methods provide insight into developing approaches that specifically address model structure.

The introduction of Markov models with structure began with the factorial HMM [11], where the authors considered a collection of individual HMMs coupled only through the measurement and proposed model learning algorithms. Shortly after, other learning algorithms were developed for a similar style of model called coupled HMMs (CHMMs) [12, 13]. In addition, the filtering problem for FHMMs was considered [14], and the authors identified key challenges for inference in the FHMM framework. The introduction of the FHMM spurred considerable interest in the control problem with a fully observable state, which lead to the factorial MDP (FMDP) formulation. An early survey paper [15] discussed various types of structure considered in Markov models and solution techniques, specifically for decision making methods.

The first solution methods for FMDPs focused on approximate dynamic programming techniques, using function projections [16, 17] and efficient linear programming with structured value function approximations [17, 18]. Analysis techniques were also developed to evaluate the approximate quality of the linear programming approach to approximate dynamic programming [19]. The same type of approaches were also translated to the case of factored POMDPs [20, 21] to reduce the complexity of producing an exact solution. Despite the improvements, the authors noted that the resulting complexity was still a challenge in practice.

Modern methods have continued to improve upon the foundation developed for FHMMs and FMDPs to consider more general models and to improve the solution methods. The GMDP was introduced by Forsell et al. [22], and the linear programming approach with structured value functions was derived for this more general model description. Approximate policy iteration methods were also developed by the same authors [23]. Novel solution techniques, based on variational planning [24] and junction graphs [25], have also been proposed to produce higher quality approximate solutions. In addition, Viega et al. extend the factored value function approach to factored POMDPs [21] and instead propose an approximate solution method which they show performs well in practice. For the inference problem, few approaches have been proposed specifically for GHMMs, with the notable exceptions a Gibbs sampling approach [26] and an auto-encoding variational inference approach [27]. However, neither of these approaches address the problem of online inference as measurements are taken.

Finally, Robbel et al. consider an additional structural property called Anonymous Influence, where the state dynamics are based on the number of variables in a particular state, and not the identity of the variables [28]. The authors derive an efficient exact solution method based on linear programming and show they are able to scale to larger models than previously considered. We also make extensive use of this property in the work we present in this thesis. We note that the prior work we have discussed has been applied to large models, on the order of 10^{50} total states and 10^{15} total actions. Nevertheless, we aim to use even larger models in our algorithms, with over 10^{1000} total states. We draw upon the lessons learned and insights provided by prior work to address significantly large GMDPs.

1.2 Approach

The mission of this thesis is to develop a complete set of algorithms to enable the use of GMDPs in modeling realistic natural phenomena. We specifically are interested in the optimal control problem with a capacity constraint, the online filtering problem, and the model learning problem for GMDPs. Furthermore, we develop frameworks based on a cooperative team of autonomous aerial vehicles to demonstrate robotics-based applications of GMDPs. We begin with the optimal control problem with a capacity constraint for the case of a fully observable state in Chapter 3. We relax the assumption of a fully observable state in Chapter 4 and develop a fast online filtering scheme, as well as propose a certainty-equivalence framework to enable the use of our control frameworks. In Chapter 5, we no longer consider the GMDP model parameters as specified a priori, and instead learn the parameters from a time history of observations for each MDP in the GMDP. We then propose two frameworks based on a cooperative team of autonomous aerial vehicles in Chapter 6 to address the persistent control and coverage problems for a forest wildfire. We also propose a more general problem formulation for cooperative team problems and develop decentralized optimization techniques which can recover the optimal centralized solution. Our proposed algorithms are validated on simulation experiments of significantly large GMDP models for forest wildfires, disease epidemics, and social networks.

Our methods rely on established techniques and results from the fields of Markov models, statistical physics, optimization, and the control of dynamical systems. For the optimal control of GMDPs, we assume that actions can be applied arbitrarily to individual MDPs at each time step, up to a specified capacity constraint. For the filtering problem, we assume a sensor model which is able to return a measurement for each MDP. Similarly, in the learning problem, we assume a time history of data exists for each MDP that we wish to include in the GMDP. For our multi-agent frameworks, we assume the existence of a suitable aerial vehicle platform which is equipped with sensors, processing power, and weight carrying capacity to enable the use of our algorithms. First, we assume the vehicles are equipped with downward facing cameras and high-frequency radios. For the cameras, we

assume that a perception pipeline converts images into a measurement, as we discuss in our filtering and multi-agent algorithms. Second, we assume the vehicles can process the onboard sensors with little to no latency and that the only computational bottleneck may be in our algorithms. We note that we do not require high polling rates for the sensors and that we specifically develop lightweight algorithms to minimize complications with deployment on an embedded computer. Third, we assume the existence of a lower-level controller which can translate the discrete trajectories output by our algorithms into suitable actuator commands. These assumptions can reasonably be implemented on a sub-class of Unmanned Aerial Vehicles (UAVs), such as quadrotors or quadcopters.

The methods developed in this thesis are based on the following domains.

1. **Markov models.** We utilize key results for the control, filtering, and learning problems for standard Markov models to develop our algorithms. In Chapter 3, we rely on bounds on value and state-action functions to develop our approximate dynamic programming framework. In Chapter 4, we make use of the recursive Bayesian filter in developing an appropriate online filtering scheme. In Chapter 5, we utilize the standard methods in model learning for HMMs, including the Expectation-Maximization framework and the Forward-Backward algorithm, to develop our learning algorithm for GMDPs. Finally, in our persistent control framework in Chapter 6, we make use of deep reinforcement learning techniques to derive a decentralized control policy for the agents.

2. **Statistical physics.** We make use of bond percolation on an infinite square lattice to propose an alternative framework for generating control policies in Chapter 3. We also use bond percolation to analyze whether or not a given control policy will stabilize a process modeled by a GMDP.

3. **Optimization techniques.** We formulate the majority of the problems in this thesis as optimization statements, and make use of established techniques to develop methods for GMDPs. We make use of existing linear program solvers in Chapter 3 for our approximate dynamic programming framework. For our filtering approach in Chapter 4, we leverage existing approximate techniques in the variational inference framework, such as the mean-field approximation. We also make use of the evidence-based lower bound. For our frameworks in Chapter 6, we utilize existing methods for the team orienteering problem in the persistent coverage problem. We also make use of the Bellman-Ford algorithm to compute the longest path on a directed acyclic graph. Lastly, we use consensus techniques and the alternating direction method of multipliers framework to solve a more general class of cooperative team problems.

4. **Nonlinear control.** We leverage existing methods to control an aerial vehicle so that we can implement our frameworks in Chapter 6. Specifically, we use an established 3D trajectory and altitude controller for quadrotor vehicles. The quadrotors can be commanded to maintain

a constant altitude and move laterally to achieve the discrete motion model we use in our algorithms.

1.3 Applications

Graph-based Markov decision processes (GMDPs) have a wide variety of applications across many different domains which are described by a collection of interacting subsystems. Notably, structured Markov models have already been used in recognition tasks, biomedical modeling, disease epidemics, forest wildfires, music theory, social networks and user interactions, and freeway traffic models. Typically, these applications are presented with a specific coupling structure that can be considered a special case of the more general GMDP formulation we present in this thesis. Therefore, by developing learning, control, and filtering algorithms, we can streamline the use of GMDPs in these applications. For example, given the increasing amount of available data on real world phenomena, certain aspects of disaster management and response could be automated. Imagine the outbreak of a new disease. As data is collected on new cases, model parameters could be learned and updated, while also simultaneously estimating the true extent of the outbreak. In addition, as medical resources are developed and distributed, the best allocation to contain the outbreak could also be determined. We illustrate this idea by using a disease epidemic model as one of several example applications in this thesis.

One of the biggest challenges of data driven modeling approaches is being able to specify arbitrarily complex models, since it must be tractable to perform certain tasks for the model to be useful. When modeling complex environmental processes, such as autonomous vehicles navigating unknown environments, simplifying assumptions are introduced to render the problem tractable. Otherwise, decision making strategies would require an infeasible amount of computation time to produce a solution. We believe that the ability to model complex scenes and environments with GMDPs will allow for more complex behavior and decision making strategies for autonomous agents navigating the real world. Our control (Chapter 3), filtering (Chapter 4), and learning (Chapter 5) algorithms provide a complete pipeline to allow an arbitrary level of complexity in modeling the spaces in which autonomous agents will need to understand and navigate in the future, such as busy intersections and crowded freeways.

Future work on autonomous robots will ultimately focus on coordinating large teams to more effectively carry out difficult and dangerous tasks. This is abundantly clear in addressing forest wildfires, which are expected to become more common, more dangerous, and more damaging, due to contributing factors such as climate change. The frameworks we present in Chapter 6 provide a glimpse into what multi-agent systems will be able to achieve in the future. Persistent surveillance will allow first responders to better distribute resources and utilize their tools, such as improving the priority of evacuation orders and evaluating impact to the local community. Eventually, we

expect that autonomous teams will also take over the persistent control problem as well, drastically reducing the risk and danger to firefighters. Furthermore, these ideas can translate to other environmental disasters as well, such as tracking oil spills and other hazardous material incidents. Our optimization statement for cooperative team problems in Chapter 6 suggests a framework for generally approaching this type of problem using distributed optimization techniques without requiring a specific model formulation.

1.4 Contributions

The main contributions of this thesis are summarized below and the following references represent publications by the author.

1. **Two frameworks for producing and analyzing approximately-optimal control policies which explicitly enforce a global capacity constraint.** We first propose an approximate dynamic programming approach based on linear programming with an approximation quality metric. We apply this framework to simulations of forest wildfires and disease epidemics and show that our method outperforms a comparable approximate dynamic programming approach with limited control effort available [29]. We also propose a control and analysis framework based on bond percolation on a infinite square lattice which more easily allows for rule-based policies and heterogeneous model properties. Here, we show that this framework also outperforms other approximate dynamic programming and percolation based methods on simulations of a forest wildfire near an urban area [30].

2. **An approximate online filtering scheme to produce state estimates.** We leverage variational inference and the mean-field approximation to derive a scalable and efficient message-passing algorithm to produce state estimates in real time. Our method is comparably accurate to the recursive Bayesian filter and loopy belief propagation, which we adopt for the online filtering problem, while requiring significantly less time [31].

3. **A combined framework for using constrained control policies with measurement uncertainty.** We propose a certainty-equivalence framework to address the constrained control problem under measurement uncertainty. We combine our filtering approach with our approximate dynamic programming control approach and show that only a minimal increase in control effort is required to provide comparable results to the case of a fully observable state. Furthermore, we show that a small increase in error in the state estimate can result in a totally ineffective policy, which further validates our filtering approach and certainty-equivalence framework. Our combined filter and controller can control GMDPs while other architectures with no filter, or other filters, are not able to control the same GMDP [32].

4. **A learning algorithm to determine model parameters from historical data.** We relax the previous assumption that the model parameters are always known a priori, and instead learn the model parameters from data. We use the Expectation-Maximization approach to develop a learning algorithm which approximately optimizes the log likelihood of the data and which has favorable complexity for large GMDPs. We show that on two publicly available data sets, our learned GMDPs better explain the observed data compared to a model with a complete independence assumption [33].

5. **Two frameworks for applying a multi-agent system to a process described by a GMDP.** We define a realistic autonomous aerial agent model and no longer assume that control actions can be arbitrarily applied in the GMDP at each time step or that measurements of all MDPs in the GMDP are available at each time step. First, we develop a framework to generate a decentralized control policy for a cooperative team of autonomous aerial agents to contain and extinguish a forest wildfire. In this framework, agents have limited control capacity, can only observe a small part of the forest, and communicate a limited amount of information. We show that we are able to produce an effective coordination scheme over a baseline heuristic, using deep reinforcement learning [34]. We also consider the problem of a cooperative team of autonomous aerial agents providing persistent surveillance of a forest wildfire. In this framework, agents have limited view of the forest and can only communicate when very close to other agents. We develop a novel scheduling scheme for agents to communicate and coordinate their efforts which can be executed in a decentralized manner. We show that our framework provides coverage comparable to the case when agents are always able to communicate and outperforms the case when agents cannot communicate at all [35].

6. **A framework for distributed optimization of general cooperative multi-agent problems.** We expand on our previous frameworks by formulating a more general optimization statement describing cooperative multi-agent problems. We leverage methods from consensus and the alternating direction method of multipliers framework to develop distributed optimization algorithms, and we discuss the conditions under which the agents can recover the optimal centralized solution. We demonstrate our distributed algorithms on synthetic data and on a persistent surveillance problem, and we show that our methods improve upon existing distributed coordination methods [36].

1.5 Organization

The remainder of this thesis is organized as follows. Chapter 2 presents the theoretical and algorithmic background necessary for the remaining chapters. We present the notation used throughout this thesis, review necessary definitions in graph theory, optimization, and Markov models, and define

the Graph-based Markov model. We also present the process models used as application domains for demonstrating and evaluating our algorithms.

In Chapter 3, we present two control frameworks for producing a policy that explicitly satisfies a control effort constraint under the assumption that the underlying state is fully observable. Our first approach is based on approximate dynamic programming, and the result is a framework to determine approximate value and state-action functions, from which a policy can be extracted. We exploit Anonymous Influence and an additional model property, which we call "Symmetry," to improve the computational complexity of this approach. Our second approach leverages bond percolation on an infinite lattice and branching processes to allow for rule-based policies, and is more suited for addressing heterogeneous properties of GMDPs. For both approaches, we derive analysis techniques to evaluate the approximation quality of the frameworks and to determine if the control policy will stabilize a GMDP. We demonstrate the effectiveness of our frameworks on simulation experiments of considerably large GMDPs, with over 10^{1000} total states and over 10^{10} feasible actions.

In Chapter 4, we address the problem of producing state estimates online as noisy measurements are taken, and relax the assumption that the underlying state is fully observable. Our approach is based on mean-field approximations in the variational inference framework, and results in a message-passing scheme similar in spirit to belief propagation methods. Again, we exploit Anonymous Influence to improve the computational complexity of our filter. Our approach produces estimates online that are at least as accurate as comparable methods while requiring two orders of magnitude less time on simulation experiments of forest wildfires and disease epidemics. Furthermore, we demonstrate the need for a fast online filter by proposing a certainty-equivalence approach for addressing the constrained control problem under measurement uncertainty. We show that by using our filter with the approximate dynamic programming control framework in Chapter 3, we can achieve comparable results to the case of a fully observable state with a minimal increase in control effort on simulation experiments of forest wildfires.

In Chapter 5, we relax the assumption that the model parameters are known a priori and consider the problem of learning the parameters directly from a time history of observations for each MDP in a GMDP. Our approach is based on the Expectation-Maximization optimization method for maximizing the log likelihood of the data, and the result is an iterative method amenable to parallelization with favorable complexity for large GMDPs. Here we use Anonymous Influence to test and evaluate differential structural assumptions in GMDP models. We use publicly available data on the Novel Coronavirus (COVID-19) in California and on Twitter interactions to fit GMDPs and show that they explain the data better than a model based on a complete independence assumption.

In Chapter 6, we first develop two frameworks for applying a cooperative team of autonomous aerial vehicles to a forest wildfire modeled by a GMDP. We define realistic agent models, which include motion and partial observability constraints. We develop a coordination framework for the

team to effectively suppress a wildfire which is based on deep reinforcement learning, and show that it improves over a baseline heuristic. We then develop a framework for the team to effectively coordinate in performing persistent coverage of a wildfire, when communication can only occur when agents are very close together. For this problem, we develop a novel scheduling scheme which can be executed in a decentralized way, and show that it outperforms a baseline in which no communication is possible and a baseline in which the team is in constant communication. Finally, we propose a more general formulation of cooperative multi-agent problems and develop decentralized optimization techniques to recover the optimal centralized solution. We discuss the conditions under which agents are able to recover the optimal solution with limited communication and demonstrate our methods on synthetic data as well as a persistent surveillance problem.

In Chapter 7, we provide concluding remarks for the research we have presented in this thesis. We summarize our contributions and remark on the capabilities our algorithms have provided over existing prior work. Finally, we consider future lines of research based on this work as well as potential new applications.

Chapter 2

GMDP Model Formulation and Problem Description

In this chapter, we define the mathematical foundation of the work in this thesis. We begin with the mathematical notation that will be referenced throughout this thesis. We then review established definitions in graph theory, optimization, and Markov models. We formally define the graph-based Markov model and additional structural assumptions. We conclude with a description of the process models we use as applications to illustrate and benchmark our algorithms.

2.1 Mathematical Preliminaries

We begin with the notation used throughout this thesis. The d-dimensional Euclidean space is $\mathbb{R}^d$ and the d-dimensional space of integers is $\mathbb{Z}^d$. The d-dimensional space of non-negative integers is $\mathbb{Z}^d_{\geq 0}$ and the d-dimensional space of strictly positive integers is $\mathbb{Z}^d_{>0}$, where the inequalities are applied element-wise. The ℓ^p norm is $\|\cdot\|_p$ and the Frobenius matrix norm is $\|\cdot\|_F$. The Hadamard product (element-wise product) of two matrices A and B is denoted by $A \circ B$. The d-dimensional identity matrix is $\mathbf{I}_d$ and the d-dimensional vector of ones is $\mathbf{1}_d$. For vectors and matrices, the operators $=$, $\geq$, and $\leq$ are element-wise comparisons. We use $\intercal$ to refer to the transpose of a vector or matrix.

We denote the indicator function $\mathbb{I}(\cdot)$ which is one when the argument condition is met and is zero otherwise. In general, lowercase variables (z) refer to a element of a vector space or a set, and script variables ($\mathcal{Z}$) represent sets. Superscripts are used to indicate variables at a given time step t (z^t, $\mathcal{Z}^t$). A collection of variables within a range of time steps is also indicated in the superscript, e.g., $z^{1:t} = \{z^1, \ldots, z^t\}$. Subscripts are used to label variables (z_i^t, $\mathcal{Z}_i^t$) and to indicate collections of variables, $z_{\mathcal{A}}^t = \{z_i^t \mid i \in \mathcal{A}\}$. When it is convenient to express a collection of variables as a

vector instead of a set, we use brackets to refer to elements of the vector, e.g., $[z^t_{\mathcal{A}}]_i$ with $z^t_{\mathcal{A}} \in \mathbb{R}^{|\mathcal{A}|}$. Similarly, we use $[A]_{i,j}$ to refer to element i, j of matrix A. Probability distributions over variables are written $p(z^t)$, joint distributions are written $p(z^t, a^t)$, and conditional distributions are written $p(z^t \mid a^t)$. Marginalization of sets of variables from a distribution are specified in the summation, e.g., $\sum_{z^t_{\mathcal{A}}} p(z^t_{\mathcal{A}})$. Expectations taken under a distribution p are written $\mathbb{E}_p[\cdot]$.

2.2 Graph Theory

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with vertex set $\mathcal{V} = \{1, \ldots, n\}$ containing n vertices and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. A graph is called undirected if for any edge between two vertices $e = (i, j) \in \mathcal{E}$ then the reverse edge exists as well, $(j, i) \in \mathcal{E}$. Conversely, a graph is called directed if this condition is not met for all edges in the edge set. An important quantity for graph-based Markov models and for our algorithms is the notion of a neighbor set, which is defined as follows.

Definition 1 (Neighbor Set)**.** The neighbor set $\mathcal{N}(\mathcal{S}) : \mathcal{S} \subseteq \mathcal{V} \to \mathcal{T} \subseteq \mathcal{V}$ is defined as,

$$\mathcal{N}(\mathcal{S}) = \bigcup_{i \in \mathcal{S}} \{j \mid (j, i) \in \mathcal{E}\}.$$

In the case that $\mathcal{S} = \{i\}$, this is the typical neighbor set of vertex i. Furthermore, for directed graphs $\mathcal{G}$, the quantity $|\mathcal{N}(\{i\})|$ is the in-degree for vertex i. However, our notion of a neighbor set also expresses the notion of neighbors of a set of vertices $\mathcal{S} \subseteq \mathcal{V}$. For simplicity, we let $\mathcal{N}(\{i\}) = \mathcal{N}(i)$ when referring to the neighbor set of vertex i.

We also make use of a specific class of graphs called directed acyclic graphs (DAGs). The directed aspect refers to the fact that in a DAG, for any vertex $i \in \mathcal{V}$, there is no sequence of directed edges which forms a path that starts at vertex i and returns to vertex i. For a DAG, we are interested in finding weighted paths of longest length starting from a given vertex, and we use the Bellman-Ford algorithm to solve this problem.

2.3 Optimization

Many problems discussed in this thesis are formulated as optimization problems. Stating robotics and decision making problems as optimization problems allows us to gain insight into the problem structure, leverage existing analysis and solution techniques, and understand the effect of applying approximations and heuristics. The minimization and maximization problems that return the optimal value of a scalar objective function J over the decision variable $z \in \mathbb{R}^d$ are denoted,

$$J^\star(z^\star) = \underset{z \in \mathbb{R}^d}{\text{minimize}}\ J(z) \qquad \text{and} \qquad J^\star(z^\star) = \underset{z \in \mathbb{R}^d}{\text{maximize}}\ J(z), \tag{2.1}$$

respectively. The respective optimization problems that return the optimal decision variable are,

$$z^\star = \arg\min_{z \in \mathbb{R}^d} J(z) \qquad \text{and} \qquad z^\star = \arg\max_{z \in \mathbb{R}^d} J(z). \tag{2.2}$$

We also review two important types of constrained optimization problems that we will make use of frequently. A constrained optimization problem is generally written as,

$$\begin{aligned} \underset{z \in \mathbb{R}^d}{\text{minimize}} \quad & f_0(z) \\ \text{subject to} \quad & f_i(z) \leq 0 \ \forall i \in \{1, \ldots, n\}, \end{aligned}$$

where the scalar functions $f_i : \mathbb{R}^d \to \mathbb{R}, i \in \{0, 1, \ldots, n\}$ define the objective function and the constraints on the decision variable. The first type of optimization problem we make use of is the linear program, where the functions f_i are linear functions of the decision variable. In this case, the functions f_i can be written as,

$$\begin{aligned} f_0(z) &= c^\intercal z, \\ f_i(z) &= A_i z - b_i, \end{aligned}$$

where $c \in \mathbb{R}^d, A_i \in \mathbb{R}^{n \times d}$, and $b_i \in \mathbb{R}^n$ parameterize the objective functions and constraints. We take advantage of existing solvers, such as interior-point methods [37, 38], to solve linear programs in our algorithms.

The second type of optimization problem we make use of is the quadratic program. In this case, the functions f_i can be written as,

$$\begin{aligned} f_0(z) &= z^\intercal Q z + c^\intercal z, \\ f_i(z) &= A_i z - b_i, \end{aligned}$$

where $Q \in \mathbb{R}^{d \times d}, c \in \mathbb{R}^d, A_i \in \mathbb{R}^{n \times d}$, and $b_i \in \mathbb{R}^n$ parameterize the objective functions and constraints. For quadratic programs, we use existing solvers such as interior point or gradient-based methods [39].

In general, the optimization problems we consider are nonlinear and nonconvex and we derive tailored algorithms to solve these problems.

2.4 Markov Models

We review the standard discrete-time and discrete-space Markov model formulation, along with the control and filtering problems for discrete Markov models [40]. We then describe the discrete-time and discrete-space graph-based Markov model.

A discrete Markov model consists of a time-varying state x^t which is drawn from a finite discrete state space $\mathcal{X}$ and is defined at discrete time steps $t \in \mathbb{Z}_{>0}$. Transitions between states at any two

consecutive time steps are described by the time-homogeneous distribution,

$$p(x^t \mid x^{t-1}), \tag{2.3}$$

with the property that $\sum_{x^t} p(x^t \mid x^{t-1}) = 1 \ \forall x^{t-1} \in \mathcal{X}$. In addition, the transition distribution has the Markov property, where only the state at $t-1$ is required in order to define the transition probabilities.

The discrete Markov decision process (MDP) introduces a control action a^{t-1} which influences state transitions,

$$p(x^t \mid x^{t-1}, a^{t-1}), \tag{2.4}$$

and the objective is to find a policy $a^t = \pi(x^t)$ which maximizes the infinite-horizon discounted reward,

$$J = \mathbb{E}_p \left[\sum_{t=1}^{\infty} \gamma^{t-1} R(x^t, \pi(x^t), x^{t+1}) \mid x^1 \right], \tag{2.5}$$

where the expectation is conditioned on the initial state x^1, γ is the discount factor, and the reward function R specifies the reward for taking actions and being in different states. Furthermore, we specifically consider models in which the action is discrete and takes on one of two values, $a^t \in \mathcal{A} = \{0, 1\}$. Intuitively, this corresponds to choosing whether or not to apply control at a given time t.

The discrete hidden Markov model (HMM) model considers the case where the state cannot be directly observed and instead only measurements of the hidden state are available. In this thesis, we consider models where the observation $y^t \in \mathcal{Y}$ is drawn from a discrete set (e.g., $\mathcal{Y} = \mathcal{X}$) and where the observation lives in a continuous space (e.g., $\mathcal{Y} = \mathbb{R}^d$). For both cases, a measurement distribution is introduced,

$$p(y^t \mid x^t), \tag{2.6}$$

which describes the likelihood of observations given the hidden state. For the online filtering problem, the objective is to produce the posterior distribution $p(x^t \mid y^1, \ldots, y^t)$ at each time step t. Bayes' rule can be used to derive the optimal filter which is a recursive relationship,

$$p(x^t \mid y^{1:t}) \propto p(y^t \mid x^t) \sum_{x^{t-1}} p(x^t \mid x^{t-1}) p(x^{t-1} \mid y^{1:t-1}), \tag{2.7}$$

where we have used the shorthand $y^{1:t} = \{y^1, \ldots, y^t\}$ to refer to the history of measurements. The filter is initialized with a prior at the initial time step, $p(x^1)$.

2.4.1 Graph-based Markov Models

We now describe the main aspects of the graph-based Markov model [26]. The model includes a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V} = \{1, \ldots, n\}$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each vertex $i \in \mathcal{V}$

corresponds to a standard Markov model with latent state $x_i^t \in \mathcal{X}_i$, action $a_i^t \in \mathcal{A}_i = \{0, 1\}$, and measurement $y_i^t \in \mathcal{Y}_i$. An important feature of graph-based Markov models is the idea that the state dynamics of model i are influenced by its neighbors,

$$p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}). \tag{2.8}$$

In other words, the probability of transitioning from a state x_i^{t-1} to a state x_i^t for a Markov model in the graph-based Markov model only depends on the previous state x_i^{t-1}, the previous state of its neighbors $x_j^t \in \mathcal{N}(i) \subseteq \mathcal{V}$ in the graph, and the action a_i^t.

It is possible form a standard Markov model description, also called an "aggregate" model description, which combines all Markov models in the graph into a single set of model parameters. The aggregate state x^t is defined by the combination of all individual models, $x^t \in \mathcal{X} = \prod_{i=1}^n \mathcal{X}_i$. Likewise, the action space and measurement spaces are similarly defined, $a^t \in \mathcal{A} = \{0, 1\}^n$ and $y^t \in \mathcal{Y} = \prod_{i=1}^n \mathcal{Y}_i$, respectively. The state dynamics for the aggregate state x^t are then defined as,

$$p(x^t \mid x^{t-1}, a^{t-1}) = \eta \prod_{i=1}^n p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}), \tag{2.9}$$

where η is a normalization factor that ensures $\sum_{x^t} p(x^t \mid x^{t-1}, a^{t-1}) = 1 \; \forall x^{t-1} \in \mathcal{X}, a^{t-1} \in \mathcal{A}$. The measurement likelihood for the aggregate state is described by the distribution,

$$p(y^t \mid x^t) = \prod_{i=1}^n p_i(y_i^t \mid x_i^t). \tag{2.10}$$

We note that this likelihood model can be extended to a more general case, with $p_i(y_i^t \mid x_i^t, x_{\mathcal{M}(i)}^t)$ and $\mathcal{M}(i) \subseteq \mathcal{V}$. We remark where this extension is possible when deriving our algorithms.

We now describe the control and filtering problems for this class of structured Markov models. In a graph-based MDP (GMDP), the reward function is additively composed of individual functions r_i which are associated with each MDP i,

$$R(x^t, a^t, x^{t+1}) = \sum_{i=1}^n r_i(x_{O(i) \cup \mathcal{N}(O(i))}^t, a_{O(i)}^t, x_{O(i)}^{t+1}). \tag{2.11}$$

Each MDP reward function is defined over a subset of variables, $O(i) \subseteq \mathcal{V} \; \forall i \in \mathcal{V}$, and typically $|O(i)| \ll |\mathcal{V}|$. The objective is to find a control policy $a^t = \pi(x^t)$ to maximize the infinite-horizon discounted reward,

$$J = \mathbb{E}_p \Big[\sum_{t=1}^{\infty} \gamma^{t-1} R(x^t, \pi(x^t), x^{t+1}) \mid x^1 \Big],$$

where the expectation is conditioned on the initial state x^1 and γ is the discount factor. A capacity constraint is enforced on the action $a^t = \pi(x^t)$, and γ is the discount factor. In particular, we are

interested in problems where a global capacity constraint is enforced on the action a^t, where the feasible action set is,

$$\mathcal{A}_c = \{a^t \in \mathcal{A} \mid \sum_{i=1}^{n} a_i^t \leq C\}, \tag{2.12}$$

and $C \in \mathbb{Z}_{\geq 0}$ is the maximum allowed capacity.

For the graph-based HMM (GHMM), there is no control action and the state is not directly observable. As in the case of the standard HMM model, the objective of the online filtering problem is to produce the posterior distribution $p(x^t \mid y^{1:t})$ at each time step t, where we again use the shorthand $y^{1:t}$ for the history of measurements. Applying the structure in the state dynamics and measurement likelihood for graph-based Markov models leads to the recursive Bayesian filter,

$$p(x^t \mid y^{1:t}) \propto \Big(\prod_{i=1}^{n} p_i(y_i^t \mid x_i^t)\Big) \sum_{x^{t-1}} p(x^{t-1} \mid y^{1:t-1}) \Big(\prod_{i=1}^{n} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1})\Big), \tag{2.13}$$

which is initialized by a prior at the initial time, $p(x^1)$.

Finally, we also describe the combined problem of optimizing the infinite-horizon discounted reward when the underlying state cannot be directly observed. In this case, the model is best described as a graph-based partially observable Markov decision process (GPOMDP). The infinite-horizon discounted reward is now conditioned on an initial state and the history of measurements,

$$J = \mathbb{E}_p\Big[\sum_{t=1}^{\infty} \gamma^{t-1} R(x^t, a^t, x^{t+1}) \mid x^1, y^{2:t}\Big]. \tag{2.14}$$

An exact solution for a GPOMDP is mapping from each possible belief over states to the optimal action, $a^t = \pi(p(x^t \mid y^{1:t}))$.

2.4.2 Anonymous Influence

We describe the main structural assumption for GMDPs that we make use of throughout the work in this thesis. We consider the case where (2.8) is based on the number of neighbors in particular states rather than the identity of these neighbors. This property is called "Anonymous Influence" and we summarize the relevant ideas [41, 28].

For a set of n discrete variables $x_i \in \mathcal{X}_i = \{0, 1, \ldots, D \in \mathbb{Z}_{\geq 0}\}$, the count aggregator (CA) is a vector $z \in \mathbb{Z}_{\geq 0}^D$ where each element describes the number of variables taking on a particular value, $[z]_j = \sum_{i=1}^{n} \mathbb{I}(x_i = j)$ and $j \in \{0, 1, \ldots, D\}$. The CA is therefore the mapping $\prod_{i=1}^{n} \mathcal{X}_i \to \mathbb{Z}_{\geq 0}^D$.

A mixed-mode function (MMF) is a real-valued function that takes as input a CA and other variables not represented by CAs. Mixed-mode functions are important in graph-based models as they allow us to drastically reduce the number of parameters required to represent the model. As a result, many of our algorithms are significantly more computationally efficient and are able to scale

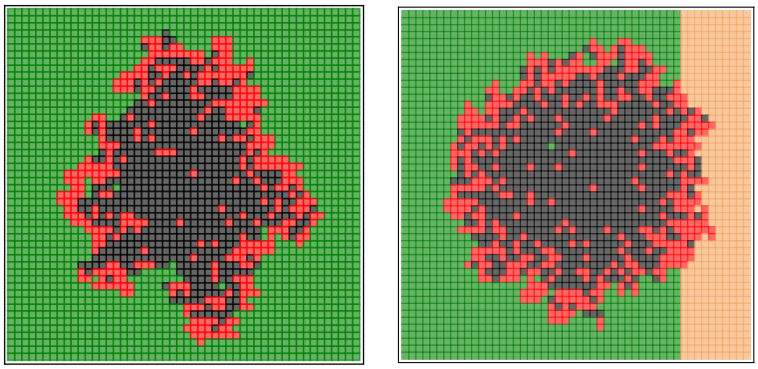

Figure 2.1: A forest wildfire modeled by a graph-based Markov decision process (GMDP), where green are healthy trees, red are fires, and black are burnt areas. (left) A wildfire burning down a forest of trees. (right) A wildfire threatening an urban region (light brown) next to a forest.

to models with large state and action spaces. A MMF can be described as the mapping $\mathbb{Z}_{\geq 0}^{D} \times \mathcal{S} \to \mathbb{R}$, where we generically use $\mathcal{S}$ to represent the space of other variables not represented by a CA.

For a GMDP where all MDPs have the same discrete domain $x_i^t \in \{0, 1, \ldots, D\}$, the state dynamics (2.8) for each MDP requires specifying (at most) $(D+1)^{|\mathcal{N}(i)|+2}$ values. If a CA z_i^{t-1} is used to represent the influence of other MDPs, then (2.8) can be represented by a MMF,

$$p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) = p_i(x_i^t \mid x_i^{t-1}, z_i^{t-1}, a_i^{t-1}). \tag{2.15}$$

Therefore,

$$(D+1)^2 \cdot \tbinom{|\mathcal{N}(i)|+D}{|\mathcal{N}(i)|} = \frac{(D+1)^2}{D!} \prod_{j=1}^{D} |\mathcal{N}(i)| + j \tag{2.16}$$

values (at most) are required using MMFs, where $\binom{n}{k}$ is the binomial coefficient. This representation is a potentially significant reduction in parameters.

2.5 Process Models

We provide details on the three different example applications which we use in this thesis, and discuss the resulting model complexity as well.

2.5.1 Forest Wildfires

The forest is modeled as a finite 2D lattice of dimensions $L \times W$ with $n = LW$ total nodes; see Fig. 2.1. Each node $i \in \{1, \ldots, n\}$ on the lattice represents a tree and the tree state x_i^t takes one of three values, $\mathcal{X}_i = \{H, F, B\} = \{\text{healthy}, \text{on fire}, \text{burnt}\}$. The position of node $i \in \{1, \ldots, n\}$ on

Table 2.1: Linear-type tree transition probabilities for wildfire model. Blank entries are zero.

		x_i^{t+1}		
		H	F	B
x_i^t	H	$1-\alpha_i f_i^t$	$\alpha_i f_i^t$	
	F		$\beta_i - \Delta\beta_i a_i^t$	$1-\beta_i+\Delta\beta_i a_i^t$
	B			1

Table 2.2: Power-type tree transition probabilities for wildfire model. Blank entries are zero.

		x_i^{t+1}		
		H	F	B
x_i^t	H	$1-\alpha_i^{f_i^t}$	$\alpha_i^{f_i^t}$	
	F		$\beta_i - \Delta\beta_i a_i^t$	$1-\beta_i+\Delta\beta_i a_i^t$
	B			1

the lattice is given by $q_i \in \mathcal{Q} = \mathbb{Z}^2 \cap \{[1, L] \times [1, W]\}$. An undirected graph is used to represent the trees and influence between trees, with the vertex set $\mathcal{V} = \{1, \ldots, n\}$. Edges exist between trees if they are neighbors on the lattice. An important quantities in this model is,

$$\text{number of neighbors which are on fire: } f_i^t = \sum_{j \in \mathcal{N}(i)} \mathbb{I}(x_j^t = F),$$

which influences the state dynamics of the trees. We consider two different models, which are summarized in Tables 2.1 and 2.2. A tree that is healthy transitions to on fire only if at least one tree in its neighbor set is on fire where α_i describes the likelihood of fire spreading from a tree on fire to a healthy tree. A tree on fire will either remain on fire or transition to burnt in a single time step. The parameters β_i and $\Delta\beta_i$ describe the average number of time steps a fire will persist and the effectiveness of control actions, respectively. Control actions are binary and reflect the choice of whether or not to apply fire retardant on a tree, $a_i^t = \{0, 1\}$. Finally, a tree that is burnt will remain burnt for all time. The difference between the two dynamics models for trees is how the number of neighbors on fire influences the transition of a tree from healthy to on fire. It is easier to ensure the state dynamics satisfy the Markov model properties for the power-type dependence, but on the other hand, the linear-type dependence is simpler to develop approximate dynamic programming techniques. We also note that the parameters α_i, β_i, and $\Delta\beta_i$ can be varied across the lattice to produce a wildfire that spreads at different rates for different directions on the lattice.

We consider heterogeneous forests as well, where some nodes on the lattice correspond to urban areas and other nodes correspond to trees. For urban areas, the state x_i^t takes on one of four values, $\mathcal{X}_i = \{H, F, B, R\} = \{\text{healthy}, \text{on fire}, \text{burnt}, \text{removed}\}$. The dynamics for an urban area are similar to the tree dynamics and the values α_i and β_i parameterize the transition probabilities; the main

Table 2.3: Urban area transition probabilities for wildfire model. Blank entries are zero.

		x_i^{t+1}			
		H	F	B	R
x_i^t	H	$(1-a_i^t)\alpha_i^{f_i^t}$	$(1-a_i^t)(1-\alpha_i^{f_i^t})$		a_i^t
	F		$\beta_i - \Delta\beta_i a_i^t$	$1-\beta_i+\Delta\beta_i a_i^t$	
	B			1	
	R				1

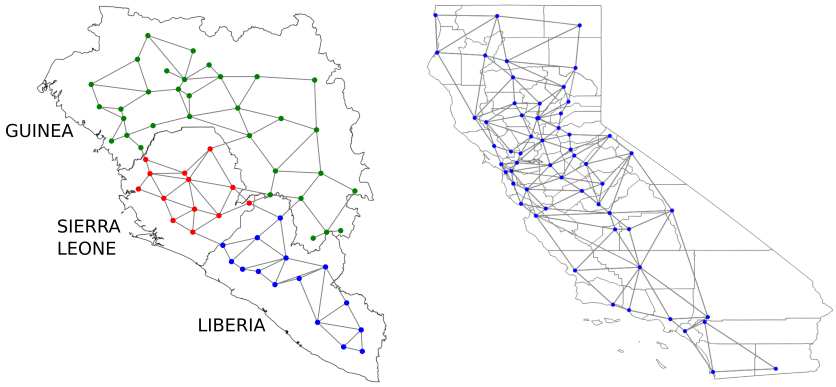

Figure 2.2: A graph-based MDP is useful for describing disease epidemics. (left) A graph-based model based on the administrative areas of Guinea, Liberia, and Sierra Leone describes the 2014 West Africa Ebola outbreak. (right) A graph-based model based on the 58 counties of California, USA describes the 2020 Novel Coronavirus (COVID-19) pandemic. (both) Gray lines indicate major transportation routes between communities.

difference is the control of healthy urban areas. Treating a healthy urban area removes it from the lattice; it is no longer capable of catching on fire or spreading fire to other trees or urban areas. This represents a controlled burn or structure removal that firefighters use to limit the spread of a wildfire. In addition, treating an urban area on fire increases the likelihood it burns out. Table 2.3 summarizes the dynamics of urban areas.

2.5.2 Virus Epidemics

We introduce a graph-based model to describe three West African countries that were affected by the 2014 Ebola outbreak, see Fig. 2.2. We use an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each vertex $i \in \mathcal{V}$ corresponds to a different administrative area within Guinea, Sierra Leone, and Liberia. Edges between administrative areas indicate major transportation routes. We refer to each administrative area as a community, and each community has one of three values, $\mathcal{X}_i = \{S, E, R\} =$

Table 2.4: Community transition probabilities for Ebola model. Blank entries are zero.

		x_i^{t+1}		
		S	E	R
x_i^t	S	$1-\alpha_i f_i^t$	$\alpha_i f_i^t$	
	E		$1-\Delta\beta_i a_i^t$	$\Delta\beta_i a_i^t$
	R			1

$\{\text{susceptible}, \text{infected}, \text{recovered}\}$. For this model, an important neighbor-based quantity is,

$$\text{number of neighbors which are infected: } f_i^t = \sum_{j \in \mathcal{N}(i)} \mathbb{I}(x_j^t = E).$$

Table 2.4 summarizes the state dynamics for each community. Similar to the wildfire model, a healthy community will only transition to infected if there is at least one neighbor which is infected. A key difference is that communities will remain in the infected state unless control is applied, in which case the community may remain infected or transition to recovered. A recovered community will remain recovered for all time. The choice of control action at each time step, $a_i^t \in \{0, 1\}$, corresponds to distributing medical resources in order to limit the spread of the virus.

We also consider a graph-based model to describe the spread of the Novel Coronavirus (COVID-19) in California, USA in 2020, which is used in our model fitting algorithm. An undirected graph is used, where each vertex $i \in \mathcal{V}$ corresponds to a county in California, of which there are 58 total. Edges between counties indicate major transportation connections; see Fig. 2.2. The state of each county x_i^t has one of four values, $\mathcal{X}_i = \{1, 2, 3, 4\}$, which is an Alert Level corresponding to the current level of risk and number of cases. The state dynamics are influenced by the following neighbor-based quantity,

$$\text{number of neighbors above Alert Level 2: } f_i^t = \sum_{j \in \mathcal{N}(i)} \mathbb{I}(x_j^t \in \{3, 4\}). \tag{2.17}$$

For a given county i, the quantity f_i^t can take on $|\mathcal{N}(i)|+1$ values, $f_i^t \in \{0, 1, \ldots, |\mathcal{N}(i)|\}$. Therefore, the county dynamics can be represented by a transition matrix $\Lambda_i \in \mathbb{R}^{4(|\mathcal{N}(i)|+1)\times 4}$, where each row must sum to one. We use publicly available health data in order to learn the state dynamics and measurement likelihood parameters. More details are provided in Chapter 5.

2.5.3 Social Networks

We also consider a social network model for our model fitting algorithm, which is based on Twitter interactions. An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is used to represent different users and connections between users. Each vertex $i \in \mathcal{V}$ corresponds to a unique user. An edge exists between user j to

user i if user i interacts with user j, e.g., through mentioning or retweeting user j. The state of each user x_i^t has one of five values,

$$\mathcal{X}_i = \{--, -, \circ, +, ++\} = \{\text{negative}, \text{leaning negative}, \text{neutral}, \text{leaning positive}, \text{positive}\},$$

which corresponds to the user's opinion on a given topic. For this model, we consider the following neighbor-related quantity,

$$\text{number of neighbors which are not neutral on topic: } f_i^t = \sum_{j \in \mathcal{N}(i)} \mathbb{I}(x_j^t \neq \circ). \tag{2.18}$$

For each user i, the quantity f_i^t can take on $|\mathcal{N}(i)| + 1$ values, $f_i^t \in \{0, 1, \ldots, |\mathcal{N}(i)|\}$. The dynamics for each user can then be represented by a transition matrix $\Lambda_i \in \mathbb{R}^{3(|\mathcal{N}(i)|+1)\times 3}$, where each row must sum to one. We collect publicly available tweets on a single topic in order to learn the state dynamics and measurement likelihood parameters. More details are provided in Chapter 5.

2.5.4 Model Complexity

We illustrate the sources of complexity in using GMDPs as a modeling framework by discussing the forest wildfire model we introduced with examples values. First, the number of possible forest configurations is astronomical — a forest of N trees has 3^N possible configurations. A 100 tree forest has more states than there are grains of sand on Earth, and a 250 tree forest has more states than atoms in the universe [42]. For comparison, there are approximately 26,000 trees in New York's Central Park, which is a relatively small sized forest. Second, in the control problem, if we impose a limit of C total trees which can be treated at each time step, there are $\binom{N}{C}$ feasible actions. For the 250 tree forest, a capacity of $C = 5$ results in 10^9 feasible actions. Third, in the online filtering problem, if the measurement space for each tree is the same as the state space $y_i^t \in \mathcal{Y}_i = \mathcal{X}_i$, then the are also an astronomical number of possible observation configurations to consider.

Finally, when we consider applying multi-agent teams to a process modeled by a GMDP in Chapter 6, we introduce an agent model with motion and sensing constraints. As a result, a GPOMDP is the appropriate model description and it is difficult to succinctly describe the additional complexity incurred by adding the agent model. These tractability issues motivate our approximate techniques which we develop throughout this thesis.

2.6 Model Abstractions

The GMDP modeling framework has significant representational power, by allowing for an arbitrary level of abstraction when defining the individual subsystems to form the overall process. An important question for a given GMDP is whether or not the chosen level of abstraction is suitable

for describing a particular physical process, i.e., whether or not the model is too high-level or too detailed to draw insight and conclusions. This question has long been a staple of modeling and analysis in classical engineering problems which typically involve solving systems of partial differential equations, e.g., finite element analysis [43, 44]. For the case of probabilistic models, and in particular discrete Markov models, analysis and verification methods are typically developed for a specific problem or data set. For the example of forest wildfires, Markov models have been developed with assumptions and parameters motivated by insight and analysis from deterministic models [45, 46]. In particular, the FARSITE model [47] is a well-known deterministic model of forest fire growth and spread which has served as the basis for probabilistic models. For information-theoretic processes, such as social networks, metrics have been generated based on standard statistical methods for model verification [48]. While methods have been developed for verifying Markov models (e.g., see [49] for a survey), verifying the influence structure of a coupled Markov model has received comparatively less attention. It is clear that this type of analysis will also be essential in order to fully realize the benefits of the GMDP modeling framework, and in this thesis we provide the initial steps to build these analysis tools. In Chapter 5, we develop metrics to evaluate the quality of a GMDP (and therefore the chosen level of abstraction) and in Chapter 7, we discuss future directions to expand our analysis.

Chapter 3

Control Policies with Global Capacity Constraints

In this chapter, we propose two approaches to generate control policies which explicitly satisfy a global capacity constraint, given that the underlying process state is fully observable. The first approach is based on standard Markov decision process solution techniques, where approximate dynamic programming is used to compute a value or state-action function, from which a policy can then be extracted. The second approach adopts ideas from statistical physics to frame the control problem as percolation on a lattice. For each approach, we derive analysis techniques to evaluate the approximation quality. We demonstrate through simulations of forest wildfires and the 2014 West Africa Ebola outbreak that our methods are effective compared to state-of-the-art approaches. Furthermore, our methods are computationally efficient and are able to scale to models with much larger state spaces than currently considered in relevant literature. The material in this chapter appears in publications [29, 30].

3.1 Introduction

In this chapter, we consider controlling processes that are modeled by a fully-observable discrete time and discrete space graph-based Markov decision problem (GMDP) which we introduced in Chapter 2.4.1. This class of models is well-suited to describing the control of large-scale spreading processes, such as forest wildfires, disease epidemics, computer viruses, and social networks [50, 51, 52, 53, 54]. We specifically consider controlling GMDPs with a global control capacity constraint as capacity constraints are crucial in modeling the control of spatial processes, such as wildfires and disease epidemics. For example, the optimal unconstrained policy for fighting a wildfire is to apply fire retardant to every tree at every time. Similarly, the optimal unconstrained policy for a disease

epidemic is to treat every person with medicine at every time. Hence, these problems are only rendered meaningful if a constraint is applied to the total control effort available, as we consider in this chapter. In addition, the modeling of spreading processes naturally leads to GMDPs with large state spaces and large constrained action spaces. As a result, standard solution techniques cannot be directly applied as they scale poorly with these quantities; we review relevant prior work in the following section.

We develop two approaches in order to address large GMDPs. The first approach is formulated as a set of linear programs (LPs) based on the approximate linear programming method for solving MDPs. We provide one algorithm for value functions and one algorithm for state-action functions. Both algorithms generate approximations by solving for the weights of a factored basis representation and we provide sub-optimality bounds for each algorithm. This approach is most appropriate for GMDPs with a property common in large-scale spatial processes called "Anonymous Influence." Simply state, a GMDP has Anonymous Influence if the state dynamics of the MDPs rely on the number of neighbor MDPs in particular states and not the identity of these neighbors. Furthermore, we reduce the computational complexity of this approach by re-using LP solutions when possible, a property we call "Symmetry." We also provide a method for deriving a policy, from either a value function or a state-action function, which explicitly satisfies a global capacity constraint.

The second approach is based on using the Galton-Watson branching process to forecast the process growth over several time steps. Using this approximation, constrained rule-based policies can be generated which prioritize different nodes in the graph to achieve multiple control objectives. We also draw connections to bond percolation from statistical physics to analyze the process stability. In contrast to our LP approach, this approach can more easily handle heterogeneous properties of GMDPs, which we consider in three aspects. First, the process may spread at different rates for different nodes in the graph. Second, each node may have a unique discrete space and discrete time Markov model describing its state evolution. Third, nodes may have different priorities for control.

For both of our approaches, we use simulations of large GMDPs to illustrate the effectiveness of our control approaches, including controlling a forest wildfire (10^{1192} states) and a forest wildfire near an urban area (10^{1255} states) with limited fire retardant, and controlling an Ebola outbreak (10^{29} states) with limited medical resources. The Ebola model is derived from data on the 2014 West Africa Ebola outbreak [55].

The remainder of this chapter is organized as follows. We review related work in Section 3.2. We derive our approximate linear programming approach in Section 3.3 and we drive our percolation-based approach in Section 3.4. We present simulation results comparing our approaches and demonstrating their effectiveness relative to prior work in Section 3.5. Concluding remarks for this chapter are provided in Section 3.6.

3.2 Related Work

As we discussed in Chapter 2.4.1, a standard MDP formulation can be created from a GMDP description. However, traditional MDP descriptions are inappropriate for structured model descriptions as the state and action spaces of the aggregate MDP description typically grow exponentially in the number of constituent MDPs. The Factored MDP (FMDP) [8] and Graph-based MDP (GMDP) [22] frameworks have been formulated to compactly represent structured MDPs. Nevertheless, the compact representation does not generally translate to tractable exact solution methods that are analogous to traditional MDP methods [16]. As a result, approximate solution techniques have been proposed.

Our first approach leverages approximate LPs (ALPs) in this chapter. ALP methods circumvent the explicit enumeration of the state space by using a basis function approximation of the value function or the $\mathcal{Q}$-function. Guestrin et al. [17] used variable elimination (VE) to efficiently generate constraints which results in an exponential dependence on the tree-width instead of the full state space. The method proposed by Forsell et al. [22] uses a specific basis function form and requires solving a linear program for each constituent MDP instead of for the aggregate MDP linear program. Chen et al. [25] proposed a VE approach that solves linear programs for the constituent MDPs and then enforces consistency constraints. Robbel et al. [28] developed a VE approach for systems where only the number of variables in a state is important, not the identity of the variables. This property, called Anonymous Influence, is also exploited in our methods.

With the exception of the approach by Forsell et al. [22], prior work considers all MDPs in the graph to compute a policy. In contrast, we consider GMDPs where each MDP belongs to one of a small number of equivalence classes, a property which we call Symmetry. Our solution technique only requires solving an LP once for each class with the resulting policy then applied to each MDP belonging to that class, resulting in a large computational benefit when there are a small number of classes. We note that [22] is the most relevant prior work but the authors do not consider control constraints or state-action functions.

The assignment of limited resources to a set of coupled or decoupled MDPs has also been considered in literature. Constrained MDP formulations [56] allow explicit control constraints but require traditional MDP descriptions. Approximate methods are proposed due to process stochasticity and large state and action spaces, including Lagrangian relaxation [57, 58], approximate dynamic programming [5, 59], Monte Carlo tree search [60], and receding horizon optimization [61]. However, these methods are intractable for the high-dimensional state and action spaces of GMDPs. We develop an approximate method for applying and satisfying a global capacity constraint that is tractable for large GMDPs. Our approximation is similar in spirit to the approach by Meuleau et al. [5].

Control methods for stochastic processes defined over a graph have also been proposed; a recent survey is [62]. However, these methods are most appropriate for continuous-time dynamical systems

models and many do not scale to large models. We instead focus on discrete time and discrete space graph-based models to develop a tractable approach.

Our second approach is based on using the Galton-Watson branching process as a model approximation and leveraging bond percolation to analyze the process stability. Percolation models have been studied extensively but have been used almost exclusively for modeling phenomena without control, such as in physics, materials science, and others [63, 64, 65]. Notable exceptions are [66, 67], which first proposed control policies within the percolation model framework. However, these works are limited to the homogeneous case which greatly simplifies the model and policy analysis. We significantly extend this work by directly addressing heterogeneous processes and we allow for general randomized and deterministic policy descriptions. In addition, while [67] develops a model approximation to analyze the stability of a policy, we create a different approximation that allows us to predict quantities of the stochastic process.

3.3 Constrained Value and State-Action Functions Via Approximate Linear Programming

We now derive approximately optimal constrained value and state-action functions, from which we then derive a constrained control policy. The use of approximate value functions is more common in prior work, but its more difficult to enforce a capacity constraint on the control action. In contrast, state-action functions are less studied in relevant prior work, but are easier to use with our capacity-constrained formulations, as we discuss in Section 3.3.3. We present both approaches to provide a broader set of tools for controlling large-scale spreading processes.

We briefly review the control model we introduced in Chapter 2.4.1 for GMDPs. We assume binary actions, $a_i^t \in \{0, 1\}\ \forall i \in \mathcal{V}$, and enforce a capacity constraint. The feasible action set at each time step is,

$$\mathcal{A}_c = \{a^t \in \mathcal{A} \mid \sum_{i=1}^{n} a_i^t \leq C\}, \tag{3.1}$$

and $C \in \mathbb{Z}_{\geq 0}$ is the maximum allowed capacity. We first derive an approach based on approximate value functions.

3.3.1 Approximate Constrained Value Functions

We consider approximate value functions which are a sum of local basis functions,

$$V_w(x^t) = \sum_{i=1}^{n} w_i^\intercal h_i(x_{O(i)}^t), \tag{3.2}$$

which mirrors the structure of the GMDP reward function (2.11), where $w_i \in \mathbb{R}^{k_i}$ and $h_i : \mathcal{X}_{O(i)} \to \mathbb{R}^{k_i}$. Each basis function h_i typically relies on state information from a few MDPs, $O(i) \subseteq \mathcal{V}$ and $|O(i)| \ll |\mathcal{V}|$. The purpose of this basis representation is to leverage the additive structure of the reward function and to greatly reduce the complexity of solving a linear program to determine the approximate value function. We use the following bound from [68] to derive a tractable method for solving for the weights of the value function, and for determining the approximation error relative to the optimal value function.

Proposition 1 (Value function approximation error [68])**.** *The maximum difference between an approximate value function $V_w(x^t)$ and the optimal value function $V^\star(x^t)$ is $\delta = \underset{x^t \in \mathcal{X}}{\text{maximize}}\ |V_w(x^t) - V^\star(x^t)|$ and is bounded,*

$$\delta \leq \frac{2\gamma}{1-\gamma} \underset{x^t \in \mathcal{X}}{\text{maximize}}\ |V_w(x^t) - (\mathcal{B}V_w)(x^t)|,$$

where $(\mathcal{B}V)(x^t)$ is the Bellman operator for value functions,

$$(\mathcal{B}V)(x^t) = \underset{a^t \in \mathcal{A}}{\text{maximize}}\ \mathbb{E}_p \left[R(x^t, a^t, x^{t+1}) + \gamma V(x^{t+1}) \right],$$

and the expectation is taken with respect to the dynamics model $p(x^{t+1} \mid x^t, a^t)$.

This bound is useful as the right hand side (R.H.S.) involves quantities that can be approximated and minimizing the R.H.S. explicitly minimizes the approximation error relative to the optimal value function. Minimizing $\phi = \max_{x^t \in \mathcal{X}} |V_w(x^t) - (\mathcal{B}V_w)(x^t)|$ leads to the non-linear program,

$$\begin{aligned} \underset{\substack{w_i \in \mathbb{R}^{k_i} \\ \phi \in \mathbb{R}}}{\text{minimize}} \quad & \phi \\ \text{subject to} \quad & \phi \geq V_w(x^t) - (\mathcal{B}V_w)(x^t), \\ & \phi \geq (\mathcal{B}V_w)(x^t) - V_w(x^t), \forall x^t \in \mathcal{X}, \end{aligned} \tag{3.3}$$

where the Bellman operator for the models considered in this work is,

$$\begin{aligned} (\mathcal{B}V_w)(x^t) &= \underset{a^t \in \mathcal{A}_c}{\text{maximize}}\ \mathbb{E}_p \left[\sum_{i=1}^{n} r_i(x^t_{O(i) \cup \mathcal{N}(O(i))}, a^t_{O(i)}, x^{t+1}_{O(i)}) + \gamma w_i^\intercal h_i(x^{t+1}_{O(i)}) \right] \\ &= \underset{a^t \in \mathcal{A}_c}{\text{maximize}}\ \sum_{i=1}^{n} \mathbb{E}_p \left[r_i(x^t_{O(i) \cup \mathcal{N}(O(i))}, a^t_{O(i)}, x^{t+1}_{O(i)}) + \gamma w_i^\intercal h_i(x^{t+1}_{O(i)}) \right], \end{aligned}$$

with the expectation taken with respect to the aggregate dynamics model (2.9). Computing operations over the full state space and the feasible action set is intractable so we develop upper and lower bounds, $(\underline{\mathcal{B}V_w})(x^t) \leq (\mathcal{B}V_w)(x^t) \leq (\overline{\mathcal{B}V_w})(x^t)$. With these bounds, the following constraints

are imposed,

$$\begin{aligned}
&\phi \geq V_w(x^t) - (\underline{\mathcal{B}V_w})(x^t) \geq V_w(x^t) - (\mathcal{B}V_w)(x^t),\\
&\phi \geq (\overline{\mathcal{B}V_w})(x^t) - V_w(x^t) \geq (\mathcal{B}V_w)(x^t) - V_w(x^t),\\
&\quad \forall x^t \in \mathcal{X},
\end{aligned}$$

and the original non-linear program constraints are still satisfied. Let the expected immediate reward and future value for every MDP be,

$$g_i(x^t_{O(i)\cup\mathcal{N}(O(i))}, a^t_{O(i)}) = \mathbb{E}_p\left[r_i(x^t_{O(i)\cup\mathcal{N}(O(i))}, a^t_{O(i)}, x^{t+1}_{O(i)}) + \gamma w_i^\intercal h_i(x^{t+1}_{O(i)})\right].$$

A lower bound is any action that satisfies the constraint. We use $a^t_i = 0\ \forall i \in \mathcal{V}$ and denote this action $\overline{a}^t$,

$$(\underline{\mathcal{B}V_w})(x^t) = \sum_{i=1}^n g_i(x^t_{O(i)\cup\mathcal{N}(O(i))}, \overline{a}^t_{O(i)}).$$

An upper bound is an over-approximation of the constrained Bellman operator by removing the capacity constraint and instead maximizing over the set of actions for each summand,

$$(\overline{\mathcal{B}V_w})(x^t) = \sum_{i=1}^n \underset{a^t_{O(i)}}{\text{maximize}}\ g_i(x^t_{O(i)\cup\mathcal{N}(O(i))}, a^t_{O(i)}).$$

Removing the constraint over-approximates the value of each MDP but is critical in dividing the original intractable non-linear program into n tractable programs. The constraints in (3.3) simplify to,

$$\begin{aligned}
&\phi \geq \sum_{i=1}^n -w_i^\intercal h_i(x^t_{O(i)}) + \underset{a^t_{O(i)}}{\text{maximize}}\ g_i(x^t_{O(i)\cup\mathcal{N}(O(i))}, a^t_{O(i)}),\\
&\phi \geq \sum_{i=1}^n w_i^\intercal h_i(x^t_{O(i)}) - g_i(x^t_{O(i)\cup\mathcal{N}(O(i))}, \overline{a}^t_{O(i)}), \forall x^t \in \mathcal{X}.
\end{aligned} \tag{3.4}$$

We now decompose the approximation error, $\phi = \sum_{i=1}^n \phi_i$, to impose the following constraints instead,

$$\begin{aligned}
&\phi_i \geq -w_i^\intercal h_i(x^t_{O(i)}) + \underset{a^t_{O(i)}}{\text{maximize}}\ g_i(x^t_{O(i)\cup\mathcal{N}(O(i))}, a^t_{O(i)}),\\
&\phi_i \geq w_i^\intercal h_i(x^t_{O(i)}) - g_i(x^t_{O(i)\cup\mathcal{N}(O(i))}, \overline{a}^t_{O(i)}),\\
&\quad \forall x^t_{O(i)\cup\mathcal{N}(O(i))}.
\end{aligned}$$

This over-approximates ϕ by adding structure to the error contribution of each MDP but reduces

the coupled non-linear program into n separate non-linear programs. In addition, the constraints in (3.4) are still satisfied after adding this structure. The maximum operator is then replaced by adding a constraint for each action which results in the linear program,

$$\begin{aligned}
\underset{\substack{w_i \in \mathbb{R}^{k_i} \\ \phi_i \in \mathbb{R}}}{\text{minimize}} \quad & \phi_i \\
\text{subject to} \quad & \phi_i \geq -w_i^\intercal h_i(x^t_{O(i)}) + g_i(x^t_{\mathcal{N}(O(i))}, a^t_{O(i)}), \\
& \quad \forall x^t_{O(i)\cup\mathcal{N}(O(i))}, a^t_{O(i)}. \\
& \phi_i \geq w_i^\intercal h_i(x^t_{O(i)}) - g_i(x^t_{O(i)\cup\mathcal{N}(O(i))}, \overline{a}^t_{O(i)}), \\
& \quad \forall x_{O(i)\cup\mathcal{N}(O(i))}.
\end{aligned} \tag{3.5}$$

Each program contains $k_i + 1$ variables and $|\mathcal{X}_{O(i)\cup\mathcal{N}(O(i))}|\left(|\mathcal{A}_{O(i)}|+1\right)$ constraints, and there is one linear program associated with each MDP in the GMDP. The quantity $\phi = \sum_{i=1}^n \phi_i$ is a sub-optimality estimate of V_w compared to the optimal value function $V^\star$. Furthermore, we can exploit Symmetry to reduce the number of programs that must be solved, as typically multiple MDPs will have identical solutions ϕ_i, w_i. In particular, we use the same basis approximation for all MDPs in the same equivalence class.

Theorem 1. *For a GMDP containing $s \leq n$ equivalence classes, the value function ALP method requires solving s linear programs and $\sum_{k=1}^s |\mathcal{C}_k|\phi_k$ is the sub-optimality error.*

Proof. The constraints in Program (3.5) are uniquely defined by the reward function r_i, dynamics p_i, and basis approximation $w_i^\intercal h_i$. By definition, all MDPs in the same equivalence class have identical reward functions r_i and dynamics p_i. Therefore, using the same basis approximation $w_i^\intercal h_i$ for all MDPs in the same equivalence class results in identical Programs (3.5). As a result, for n equivalence classes, the solution to each program is unique as no two MDPs share the same class. When there are $s < n$ classes, there are at most s unique solutions for all n linear programs. In this case, only one linear program per equivalence class must be solved as the solution for the per-MDP program (3.5) is identical for all MDPs within a class. □

3.3.2 Approximate Constrained State-Action Functions

We now derive a novel ALP approach to produce a state-action function (i.e., a Q-function) to approximate a constrained value function. Our approximate state-action function approach is based on the following bound [68].

Proposition 2 (State-action function approximation error [68])**.** *The difference $\delta = \max_{x^t \in \mathcal{X}} |V_w(x^t) - V^\star(x^t)|$ is bounded by the maximum difference between the approximate state-action function*

$Q_w\left(x^t, a^t\right)$ and the Bellman operator on $Q_w(x^t, a^t)$,

$$\delta \leq \frac{2}{1-\gamma} \underset{x^t \in \mathcal{X}, a^t \in \mathcal{A}}{\text{maximize}} \left| Q_w(x^t, a^t) - \left(\mathcal{B}Q_w\right)(x^t, a^t) \right|,$$

where $\left(\mathcal{B}Q\right)(x^t, a^t)$ is the Bellman operator for state-action functions,

$$\left(\mathcal{B}Q\right)(x^t, a^t) = \mathbb{E}_p \left[R(x^t, a^t, x^{t+1}) + \gamma \underset{a^{t+1} \in \mathcal{A}}{\text{maximize}} \; Q(x^{t+1}, a^{t+1}) \right].$$

We assume the approximate state-action function Q_w form,

$$Q_w(x^t, a^t) = \sum_{i=1}^{n} w_i^\intercal b_i(x^t_{O(i)}) + a_i^t w_i^\intercal c_i(x^t_{O(i)}), \tag{3.6}$$

with $w_i \in \mathbb{R}^{k_i}$ and $b_i, c_i : \mathcal{X}_{O(i)} \subseteq \mathcal{X} \to \mathbb{R}^{k_i}$, specifically for our capacity constrained formulations which we describe in the next section. Minimizing $\phi = \underset{x^t \in \mathcal{X}, a^t \in \mathcal{A}_c}{\text{maximize}} \left| Q_w(x^t, a^t) - \left(\mathcal{B}Q_w\right)(x^t, a^t) \right|$ results in the non-linear program,

$$\begin{aligned}
\underset{\substack{w_i \in \mathbb{R}^{k_i} \\ \phi \in \mathbb{R}}}{\text{minimize}} \quad & \phi \\
\text{subject to} \quad & \phi \geq Q_w(x^t, a^t) - (\mathcal{B}Q_w)(x^t, a^t) \\
& \phi \geq (\mathcal{B}Q_w)(x^t, a^t) - Q_w(x^t, a^t), \\
& \qquad \forall x^t \in \mathcal{X}, a^t \in \mathcal{A}_c,
\end{aligned}$$

where the non-linearity is due to the maximization in the Bellman operator. We follow a similar procedure as before to develop a tractable and scalable method. The constrained Bellman operator is,

$$\begin{aligned}
\left(\mathcal{B}Q\right)(x^t, a^t) = \mathbb{E}_p \Bigg[& \sum_{i=1}^{n} r_i(x^t_{O(i) \cup \mathcal{N}(O(i))}, a^t_{O(i)}, x^{t+1}_{O(i)}) \\
& + \gamma \max_{a^{t+1} \in \mathcal{A}_c} \sum_{i=1}^{n} w_i^\intercal b_i(x^{t+1}_{O(i)}) + a_i^{t+1} w_i^\intercal c_i(x^{t+1}_{O(i)}) \Bigg].
\end{aligned}$$

We construct upper and lower bounds for the constrained Bellman operator to instead impose the constraints,

$$\begin{aligned}
& \phi \geq Q_w(x^t, a^t) - (\underline{\mathcal{B}Q_w})(x^t, a^t), \\
& \phi \geq (\overline{\mathcal{B}Q_w})(x^t, a^t) - Q_w(x^t, a^t), \forall x^t \in \mathcal{X}, a^t \in \mathcal{A}_c.
\end{aligned}$$

Function arguments are omitted at times for clarity in the following discussion. A lower bound is

any action a^{t+1} that satisfies the control constraint. A convenient choice is $a_i^{t+1} = 0 \; \forall i \in \mathcal{V}$ thus,

$$\left(\underline{\mathcal{B}Q_w}\right)(x^t, a^t) = \sum_{i=1}^{n} \mathbb{E}_p \left[r_i(x^t_{O(i)\cup\mathcal{N}(O(i))}, a^t_{O(i)}, x^{t+1}_{O(i)}) + \gamma w_i^\intercal b_i(x^{t+1}_{O(i)}) \right].$$

An upper bound is found by removing the capacity constraint and choosing actions to improve the total value of Q_w,

$$\left(\overline{\mathcal{B}Q_w}\right)(x^t, a^t) = \sum_{i=1}^{n} \mathbb{E}_p \Big[r_i(x^t_{O(i)\cup\mathcal{N}(O(i))}, a^t_{O(i)}, x^{t+1}_{O(i)}) + \gamma w_i^\intercal b_i(x^{t+1}_{O(i)}) + \gamma \text{maximize } \{0, w_i^\intercal c_i(x^{t+1}_{O(i)})\} \Big].$$

The maximization in the upper bound is replaced by two linear constraints and the error ϕ is decomposed as a sum $\sum_{i=1}^{n} \phi_i$. The result is the following per-MDP linear program,

$$\begin{aligned}
\underset{\substack{w_i \in \mathbb{R}^{k_i} \\ \phi_i \in \mathbb{R}}}{\text{minimize}} \quad & \phi_i \\
\text{subject to} \quad & \phi_i \geq w_i^\intercal b_i + a_i^t w_i^\intercal c_i - \mathbb{E}_p \left[r_i + \gamma w_i^\intercal b_i \right], \\
& \phi_i \geq \mathbb{E}_p \left[r_i + \gamma w_i^\intercal b_i \right] - w_i^\intercal b_i - a_i^t w_i^\intercal c_i, \\
& \phi_i \geq \mathbb{E}_p \left[r_i + \gamma w_i^\intercal b_i + \gamma w_i^\intercal c_i \right] - w_i^\intercal b_i - a_i^t w_i^\intercal c_i, \\
& \quad \forall x^t_{O(i)\cup\mathcal{N}(O(i))}, a^t_{O(i)},
\end{aligned} \tag{3.7}$$

where we have omitted the function arguments for clarity. Each program contains $k_i + 1$ variables and $3|\mathcal{X}_{O(i)\cup\mathcal{N}(O(i))}||\mathcal{A}_{O(i)}|$ constraints. Solving Program (3.7) for each MDP does not enforce that the total control effort will satisfy the capacity constraint. However, allowing infeasible actions results in a more conservative approximation of the true constrained value function since adding constraints to Program (3.7) cannot lower the error ϕ_i. Similar to Theorem 1, we again exploit Symmetry to reduce the number of linear programs that must be solved, by using the same basis approximation for all MDPs in the same equivalence class.

Theorem 2. *For a GMDP containing $s \leq n$ unique equivalence classes, the approximate state-action function ALP method requires solving s linear programs and $\sum_{k=1}^{s} |\mathcal{C}_k| \phi_k$ is the approximation error.*

Proof. The approximate state-action function Q_w is determined after solving for the weights w_i. The Program (3.7) for two MDPs have identical solutions w_i, ϕ_i if both are in the same equivalence class and the same basis functions are used. Therefore, for n classes, n programs are solved to determine Q_w. Only s programs are solved for $s < n$ classes as the solution is identical for all MDPs in the same class. □

3.3.3 Deriving the Resulting Constrained Policy

In the previous two sections, we derived methods for constructing an approximate value or state-action function. However, it is still necessary to extract a policy, which may be non-trivial given the $\binom{n}{C}$ possible feasible constrained actions. Therefore, we now introduce a class of linear programs that have a capacity constraint and an explicit solution, and then discuss building a policy for an approximate value or state-action function.

Proposition 3 (Capacity Constrained Linear Program)**.** *The integer linear program,*

$$\underset{a^t \in \mathcal{A}_c}{\text{maximize}} \ \lambda + \sum_{i=1}^{n} \mu_i a_i^t, \tag{3.8}$$

where $\lambda, \mu_i \in \mathbb{R}$, has an explicit solution. Assume that $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_n$. An optimal solution is,

$$a_i^t = \begin{cases} 1 & \text{if } i \leq C \text{ and } \mu_i \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

Proof. The values μ_i can always be sorted a priori. The solution is optimal as changing it cannot improve the objective. Consider an optimal solution described as $a_i^t = 1$ for $i \in \{1, \ldots, j\}$ with $j \leq C$ and zero otherwise. If $j = C$, then choosing $a_k^t = 1$ for any $k > j$ will violate the constraint. If $j < C$, then choosing $a_k^t = 1$ for $j < k \leq C$ lowers the objective as μ_k must be negative. Finally, switching $a_k^t = 1$ to $a_k^t = 0$ for $k \leq j$ does not improve the objective as μ_k must be non-negative. □

We now discuss the conditions under which our approximate functions result in the policy described by (3.9).

Theorem 3. *For approximate value functions V_w, if the form of the dynamics (2.8), reward functions (2.11), and basis functions (3.2) result in,*

$$\mathbb{E}_p \left[R(x^t, a^t, x^{t+1}) + \gamma V_w(x^{t+1}) \right] = \lambda + \sum_{i=1}^{n} \mu_i a_i^t, \tag{3.10}$$

then the constrained policy is determined by (3.9). Furthermore, for approximate state-action functions of the form in (3.6), the constrained policy is determined by (3.9).

Proof. The approximate value function is determined after solving for the weights of the basis function representation. If the relationship in (3.10) holds, the constrained policy is,

$$\pi(x^t) = \arg\max_{a^t \in \mathcal{A}_c} \ \lambda + \sum_{i=1}^{n} \mu_i a_i^t.$$

Therefore, the policy $\pi(x^t)$ is determined by (3.9). For approximate state-action functions, if the assumed form (3.6) is used then the constrained policy is,

$$\pi(x^t) = \arg\max_{a^t \in \mathcal{A}_c} \sum_{i=1}^{n} w_i^\intercal b_i(x^t_{O(i)}) + a_i^t w_i^\intercal c_i(x^t_{O(i)}). \tag{3.11}$$

Let $\lambda = \sum_{i=1}^{n} w_i^\intercal b_i(x^t_{O(i)})$ and $\mu_i = w_i^\intercal c_i(x^t_{O(i)})$. The constrained maximization (3.11) is equivalent to Program (3.8) and so the policy is determined by (3.9). □

Although the policy (3.9) exactly solves Program (3.8), it is necessary in our derivations to remove the capacity constraint when determining an approximate value function V_w or state-action function Q_w to develop tractable methods. As a result, the approximate functions are analogous to unconstrained solutions found via dynamic programming. The use of (3.9) with an unconstrained approximate solution is therefore an approximation to the true constrained policy determined by a method that explicitly includes the control constraint.

3.3.4 Exploiting Anonymity in Linear Programs

Theorems 1 and 2 describe how we leverage Symmetry to reduce the total number of linear programs that must be solved to fully determine an approximate value or state-action function. Similarly, Anonymous Influence can be exploited to simplify the implementation of the Programs (3.5) and (3.7). For example, consider the wildfire model (Chapter 2.5) and let $O(i) = i \cup \mathcal{N}(i)$ and $|\mathcal{N}(i)| = 4$ for all trees. Without mixed-mode functions (MMFs), Program (3.5) requires enumerating on the order of 10^7 state combinations. By using a MMF, for the basis functions we present in Section 3.5, we only need to consider on the order of 10^3 state combinations. This reduction significantly simplifies the implementation of our framework which is still tractable for graphs where MDPs may have many neighbors or large state spaces. We present results demonstrating the effectiveness of our algorithms in Section 3.5.

3.4 Rule-based Policies and Analysis with Bond Percolation

In the previous section, we derived algorithms based on the approximate linear programming approach for solving MDPs. We now consider a different modeling perspective of GMDPs in order to derive a complementary approach using branching processes and bond percolation. Specifically, the nodes in the GMDP are arranged as a 2D lattice structure and the edges of each node are the corresponding neighbors on the lattice. This perspective allows us to leverage tools from population models and statistical physics in order to generate and analyze rule-based policies, which is more difficult to do in approximate dynamic programming frameworks. Furthermore, it is also easier to address heterogeneous model properties, such as varying model parameters or using unique

Markov models for each graph node. While our lattice-based framework is capable of describing any graph-based growth process, several quantities depend directly on the specific model formulation. Therefore, we use a heterogeneous model of forest wildfire near an urban area as an example to illustrate our approach. We described this model in Chapter 2.5.

3.4.1 Heterogeneous Bond Percolation

We now introduce the bond percolation model to analyze the lattice-based Markov model behavior as a function of the model parameters α_i and β_i and the control policy. The 2D bond percolation model consists of nodes on an infinite square lattice where a bond exists between two nodes with a probability independent of other nodes [63]; this probability is called the bond percolation parameter. There exists a critical value for this parameter above which there is a path of connected nodes of infinite length [69, 70].

The forest wildfire, and other stochastic growth processes, are considered bond percolation with persistence [51]. For a wildfire, the "persistence" nature is due to nodes on fire being able to spread fire over multiple time intervals until it transitions to burnt or until it has no healthy neighbors.

The bond percolation parameter for two nodes i and j is denoted by p_{ij}. For the wildfire process, consider two neighboring nodes where node i is on fire and node $j \in \mathcal{N}(i)$ is healthy; the node type does not modify the following derivation. In the absence of control, $a_i^t = 0 \ \forall i \in \mathcal{V}, \forall t \in \mathbb{Z}_{>0}$, the probability that node i never causes node j to transition to on fire is,

$$\begin{aligned} 1 - p_{ij} &= \sum_{t=1}^{\infty} \beta_i^{t-1}(1-\beta_i)(\alpha_j)^t \\ &= \frac{(1-\beta_i)\alpha_j}{1-\beta_i\alpha_j}, \end{aligned}$$

based on the dynamics in Tables 2.2 and 2.3. After algebraic manipulation,

$$p_{ij}(\alpha_j, \beta_i) = \frac{1-\alpha_j}{1-\alpha_j\beta_i}. \tag{3.12}$$

In general, $p_{ij} \neq p_{ji}$, due to the potential uniqueness of α_i and β_i for different nodes. This property is a significant difference from the homogeneous case where $p_{ij} = p_{ji} = p \ \forall i, j$. We build a model approximation that directly addresses the uniqueness of the percolation parameter, which we present in Section 3.4.2.

Given a lattice-based model, the parameter p_{ij} is computed for all pairs of nodes and represents the likelihood of fire continuing to spread if the node i were to catch on fire. The following theorem provides conditions for two types of process behavior for percolation models.

Theorem 4 (Theorems 3.1, 3.2 [70])**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a countably infinite connected graph, with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$, that represents the square lattice. Let $p' = \{p'_e \in [0,1] \mid e \in \mathcal{E}\}$ be the*

set of percolation parameters where one parameter is associated with each edge.

(i). If

$$\forall e \in \mathcal{E} \quad p'_e \leq \frac{1}{2}, \tag{3.13}$$

there exists almost surely (a.s.) no infinite cluster.

(ii). If there exists $\delta > 0$ such that,

$$\forall e \in \mathcal{E} \quad p'_e \geq \frac{1}{2} + \delta, \tag{3.14}$$

then there exists a.s. exactly one infinite cluster.

The parameter p'_e, which is associated with a graph edge, is equivalent to the parameter p_{ij}, which is computed for a pair of lattice nodes by (3.12). For the wildfire process, a "cluster" is a subset of nodes that are either on fire or burnt and a path exists between any two nodes. Therefore, we refer to a process as subcritical if the condition (3.13) is met as the majority of trees and urban areas are not expected to be eventually burnt. Conversely, a process is supercritical if the condition (3.14) is met and the majority of trees and urban areas are expected to be eventually burnt.

While Theorem 4 is valid for infinite square lattices, very large finite lattices exhibit similar behavior. We also include finite lattice effects in our model approximation. Given Theorem 4, the following theorem relates model parameters to a supercritical wildfire.

Theorem 5 (Critical Parameters for Percolation)**.** *If there exists a $\delta > 0$ such that,*

$$\forall i, j \in \mathcal{V} \qquad \frac{1 - \alpha_j}{1 - \alpha_j \beta_i} \geq \frac{1}{2} + \delta, \tag{3.15}$$

then the forest wildfire process is supercritical.

Proof. For a pair of nodes on the lattice the equivalent percolation parameter is determined by (3.12). The statement then follows from (3.14) in Theorem 4. □

The Markov model defined in the previous section describes the probabilistic state evolution of individual nodes. The percolation model, in contrast, specifies a "spreading" probability between nodes and then characterizes the resulting process. Theorem 5 provides a bridge between these two modeling perspectives.

Thus far, we have discussed the properties of the lattice-based Markov model in the absence of control. In the context of classical linear feedback control, the open-loop process dynamics are unstable without control when the condition in (3.14) is met. Feedback control is then used to produce a closed-loop system which is stable. Control actions reduce the percolation parameter p_{ij} by modifying the dynamics of the Markov model and if the condition in (3.13) is met then the system is stable. Therefore, one possible control objective is to stabilize a heterogeneous growth process, which we discuss further in Section 3.4.3.

Based on Tables 2.2 and 2.3, there are two cases to consider for the influence of control actions, depending on whether fire may spread to a healthy tree or a healthy urban area. If node i is on fire (either a tree or an urban area) and its neighbor node j is a healthy tree, then the change in percolation parameter is computed using (3.12) as,

$$\Delta p_{ij,1}(a_i^t, a_j^t) = p_{ij}(\alpha_j, \beta_i) - p_{ij}(\alpha_j, \beta_i - \Delta\beta_i a_i^t),$$

since a healthy tree is not influenced by control. If node i is on fire and its neighbor node j is a healthy urban area,

$$\Delta p_{ij,2}(a_i^t, a_j^t) = p_{ij}(\alpha_j, \beta_i) - p_{ij}(\alpha_j(1 - a_j^t), \beta_i - \Delta\beta_i a_i^t),$$

since urban areas can be removed from the lattice. The effect of control in the percolation model is thus,

$$\Delta p_{ij}(a_i^t, a_j^t) = \begin{cases} \Delta p_{ij,1}(a_i^t, a_j^t) & \text{if } x_i^t = F, x_j^t = H, j \text{ is a tree,} \\ \Delta p_{ij,2}(a_i^t, a_j^t) & \text{if } x_i^t = F, x_j^t = H, j \text{ is an urban area,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.16}$$

for the wildfire process. Lastly, we note that at each time step there is a limited subset of lattice nodes that contribute to the continued spread of the process. For the forest wildfire, these are the nodes that are on fire and have at least one healthy neighbor.

Definition 2 (Growth Boundary)**.** A node on the lattice is part of the growth boundary $\mathcal{B}^t$ at time t if it is on fire, $x_i^t = F$, and at least one neighbor is healthy,

$$h_i^t = \sum_{j \in \mathcal{N}(i)} \mathbb{I}(x_j^t = H) > 0. \tag{3.17}$$

In the next section, we introduce the Galton-Watson branching process to predict the expected future size and stopping time for a stochastic growth process on a finite lattice.

3.4.2 Galton-Watson Branching Process Model

The Galton-Watson branching process has been used to model population dynamics and is defined on a directed acyclic graph [71]. The graph starts with a single (root) node that produces a limited number of children with a prescribed probability distribution. These children become the next parents that can produce another group of children. The graph is organized by generations, which includes a set of parents, the number of children they can produce, and the probability distribution for producing children.

For the wildfire process, we use a branching process to represent the spreading dynamics of each

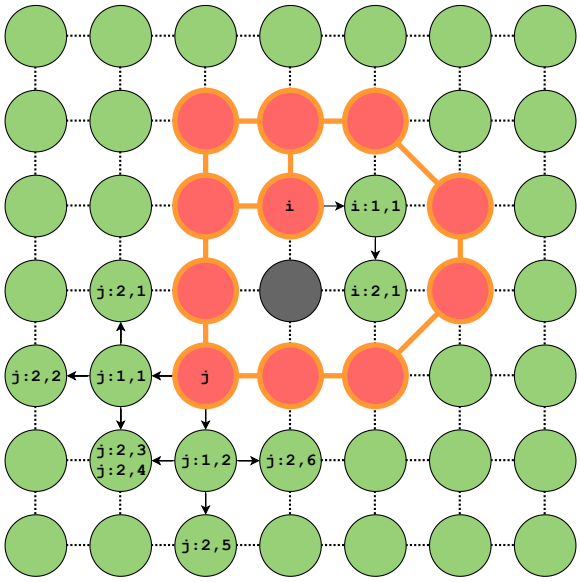

Figure 3.1: Illustration of the boundary for the percolation framework. Example lattice state x^t for the wildfire process where green represents healthy trees, red are on fire, and black are burnt. The orange line indicates the boundary $\mathcal{B}^t$. The branching process model is illustrated for nodes i and j where arrows indicate the possible growth of the process in one and two generations, which is analogous to a prediction of one and two future time steps. Other node labels contain the boundary node, generation number, and a unique identifier, e.g., `j:1,2` refers to the second node that node j could spread fire to in one generation. One node has two labels due to the branching process model; see Fig. 3.2.

node in the boundary $\mathcal{B}^t$ at each time step. The first children are healthy nodes that may transition to on fire due to the boundary node. These children then become the next set of parents that may further spread fire to additional healthy nodes. Therefore, the parent nodes in each generation represent the nodes that may form part of the boundary at future time steps.

We use the term generations to refer to future time steps of the process given the lattice state x^t at a particular time step. Therefore, predicting over multiple generations using branching processes corresponds to predicting statistics of the process over multiple time steps. We are also interested in using the predicted statistics in control policies to effectively control the process.

Percolation on a lattice is not a branching process as there are multiple paths between any two nodes on the lattice. To compute the likelihood of the true process spreading further, it is necessary to enumerate a combinatorial number of paths on the lattice which is not feasible. Therefore, we instead assume each boundary node is a branching process that ignores how other boundary nodes may spread.

Let $\mathcal{GW}_i$ be a heterogeneous Galton-Watson (GW) branching process associated with boundary node $i \in \mathcal{B}^t$ at time t; see Fig. 3.1. Each generation n of the $\mathcal{GW}_i$ process has an associated

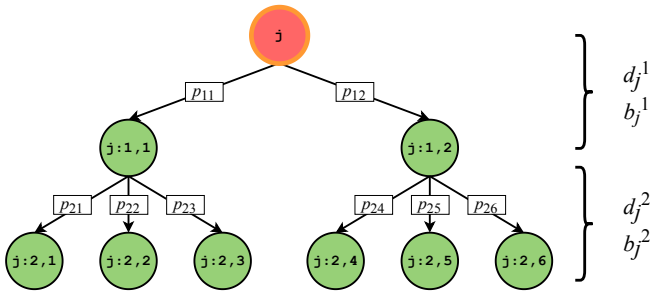

Figure 3.2: Illustration of the Galton-Watson process for the percolation framework. Equivalent branching process representation for boundary node j in Fig. 3.1. The node with two labels in Fig. 3.1 is considered two unique nodes (here, `j:2,3` and `j:2,4`) for the branching model to avoid cycles in the graph. Note that p_{23} may not equal p_{24} due to the different parent nodes as indicated by (3.12). Without control, $d_j^1 = \frac{1}{2}\sum_{k=1}^{2} p_{1k}$ and $b_j^1 = 2$ for generation one and $d_j^2 = \frac{1}{6}\sum_{k=1}^{6} p_{2k}$ and $b_j^2 = 3$ for generation two.

(potentially unique) children distribution Y_i^n. The quantity Z_i^n describes the expected size of the n^{th} generation with $Z_i^0 = 1$. We refer to the collection of processes as $\mathcal{GW}_{\mathcal{B}^t} = \{\mathcal{GW}_i \mid i \in \mathcal{B}^t\}$. The benefit of using GW processes is the simplicity in computing statistics of each generation [72]. The probability that process $\mathcal{GW}_i$ stops at generation n is,

$$s_i^n = p(Z_i^n = 0) = g_i^1(g_i^2(\cdots g_i^n(0))),$$

where g_i^τ is the probability generating function (PGF),

$$g_i^\tau(x) = \sum_{k=0}^{\infty} p(Y_i^\tau = k)x^k.$$

The expected size of each generation is,

$$\mathbb{E}[Z_i^n] = \prod_{\tau=1}^{n} \mathbb{E}[Y_i^\tau].$$

To predict the process growth using GW processes, we must specify the possible children for each parent and the probability distribution for producing children within each generation. In the wildfire model, this corresponds to determining how fire may spread given a lattice state x^t and how likely fire will propagate to different healthy nodes. We first define a function to specify the children of a parent.

Definition 3 (Node Children)**.** For the wildfire model with lattice state x^t, the children of a node

are the neighbor nodes that are healthy,

$$\mathcal{C}^t(i) = \{j \mid j \in \mathcal{N}(i) \text{ and } x_j^t = H\}.$$

When determining the children of parent nodes, nodes that are contained in a previous generation as parents are ignored to prevent cycles in the process. For the wildfire process, we use the binomial distribution to specify the the likelihood of fire spreading and so,

$$g_i^\tau(x) = (1 + (x-1)d_i^\tau)^{b_i^\tau},$$
$$\mathbb{E}[Y_i^\tau] = b_i^\tau d_i^\tau,$$

where we call d_i^τ the child rate and b_i^τ the branching factor. The child rate is based on the percolation parameter to capture the heterogeneous nature of the process,

$$d_i^\tau = \frac{\sum_{r\in\mathcal{P}^{\tau-1}}\sum_{c\in\mathcal{C}^t(r)} p_{rc} - \Delta p_{rc}(a_r^t, a_c^t)}{\sum_{r\in\mathcal{P}^{\tau-1}}|\mathcal{C}^t(r)|}, \tag{3.18}$$

where $\mathcal{P}^{\tau-1}$ refers to the set of parent nodes of generation $\tau - 1$. The branching factor is then the ratio,

$$b_i^\tau = \frac{\sum_{r\in\mathcal{P}^{\tau-1}}|\mathcal{C}^t(r)|}{|\mathcal{P}^{\tau-1}|}. \tag{3.19}$$

Finally, for the set of GW processes $\mathcal{GW}_{\mathcal{B}^t}$,

$$s_{\mathcal{B}^t}^n = \prod_{i\in\mathcal{B}^t} s_i^n \tag{3.20}$$

$$\mathbb{E}[Z_{\mathcal{B}^t}^n] = \sum_{i\in\mathcal{B}^t} \mathbb{E}[Z_i^n] \tag{3.21}$$

are the stopping probability and expected number of nodes on fire at generation n, respectively.

Fig. 3.2 shows the GW process approximation of a boundary node for the example lattice state in Fig. 3.1. The root node, which is also the parent of generation one, is the node on fire itself. The children of generation one are `j:1,1` and `j:1,2` and each may have a unique percolation parameter. The number of children and the percolation parameters are used to determine the child rate (3.18) and branching factor (3.19). The same process repeats for the second generation with the children of generation one now serving as the parents. In this example, there is also a shared child that is double-counted in generation two which is necessary to prevent graph cycles.

Algorithm 1 summarizes the use of the GW process approximation for each time step. Since the boundary $\mathcal{B}^t$ is not known exactly after the first generation, the estimated quantities (3.20) and (3.21) are used with the policy (line 8). For example, the UBT policy (Section 3.4.3) uses the known quantity $|\mathcal{B}^t|$ for the the first generation and then the generated estimate $\mathbb{E}[Z_{\mathcal{B}^t}^\tau]$ for subsequent

Algorithm 1 Branching Process Model

Input: Lattice state x^t, control policy
Output: Predicted boundary size and stopping time
Determine growth boundary $\mathcal{B}^t$
Associate a $\mathcal{GW}$ process with each node in the boundary $\mathcal{B}^t$
for $\tau = 1, \ldots, n$ generations **do**
    for each process $\mathcal{GW}_i, i \in \mathcal{B}^t$ **do**
        Determine children of generation τ
        Calculate child rate d_i^τ with policy (3.18)
        Calculate branching factor b_i^τ (3.19)
        Add unique children as parents of next generation $\tau + 1$
    Compute stopping time $s_{\mathcal{B}^t}^\tau$ (3.20)
    Compute expected boundary size $\mathbb{E}[Z_{\mathcal{B}^t}^\tau]$ (3.21)

generations.

By predicting the process growth over multiple generations, it is possible to build more effective policies. For example, in Fig. 3.1, the boundary node i will stop spreading after two generations with probability one. Therefore, this node can safely be ignored and control resources should instead be used for other nodes. In the next section, we introduce control policies that satisfy a resource limit and characterize the conditions for a policy to stabilize a supercritical process.

3.4.3 Defining Control Policies and Stability Analysis

We first define two benchmark randomized policies based on previous work. All of the following policies strictly satisfy a resource limit $C \geq 0$, so that $\sum_{i \in \mathcal{V}} a_i^t \leq C \ \forall t \in \mathbb{Z}_{>0}$.

Definition 4 (Uniform Boundary Treatment (UBT) [66])**.** Choose C fires in the boundary $\mathcal{B}^t$ with uniform probability to treat at each time step.

Definition 5 (Degree Weighted Treatment (DWT), based on [29])**.** Choose C fires in the boundary $\mathcal{B}^t$ with probability,

$$\frac{h_i^t}{\sum_{i \in \mathcal{B}^t} h_i^t},$$

at each time step, where h_i^t is the number of healthy neighbors (3.17). The quantity h_i^t can be interpreted as the out-degree of each boundary node.

Next, we define two novel deterministic control policies. The first policy, receding horizon treatment (RHT), uses the percolation parameters over multiple generations of a boundary node to estimate its rate of growth, which we call the "volatility" of a node on fire.

Definition 6 (Receding Horizon Treatment (RHT))**.** Rank all nodes in the boundary $\mathcal{B}^t$ in order of highest to lowest estimated volatility (Algorithm 2) where the parameter k is the number of

generations to consider. Treat C nodes on fire with the highest estimated volatility at each time step.

The second policy, Urban Safety Treatment (UST), is summarized in Algorithm 3. This policy first checks if any node on the boundary $\mathcal{B}^t$, after predicting k generations, will include an urban area. If so, the urban areas are ranked, in order of highest to lowest, by the number of boundary nodes that included a given urban area in their associated branching process. Up to C urban areas are then removed. Any remaining control is used to treat nodes on fire in the boundary. The ordering of boundary nodes includes their proximity to the right-most lattice edge (line 4) due to the arrangement of urban areas on the lattice; see Section 3.5. Lastly, if urban areas are removed, then nodes on fire with low volatility (line 3) are ignored. This is intended to eliminate boundary nodes that will stop spreading fire after k generations; an example of this is node i in Fig. 3.1.

Given a control policy, the following theorem provides a condition for a policy to be considered stabilizing.

Theorem 6 (Stabilizing Policy)**.** *A policy is stabilizing if,*

$$\forall t \in \mathbb{Z}_{>0} \quad \forall i \in \mathcal{B}^t, j \in \mathcal{N}(i): \quad p_{ij} - \Delta p_{ij}(a_i^t, a_j^t) \leq \frac{1}{2}. \tag{3.22}$$

Proof. By (3.13), a policy must reduce the percolation parameter below $\frac{1}{2}$ in order to change a supercritical process to a subcritical process. Consider the lattice state x^t at time t. Any node that is not part of the boundary $\mathcal{B}^t$ cannot spread fire to a healthy node in which case $p_{ij} = 0$. Therefore, the policy must modify the percolation parameter of all nodes that can spread fire, which by definition are those in the boundary $\mathcal{B}^t$, to satisfy (3.13) at t. Thus, if the policy achieves this for all times t, then (3.13) is always satisfied, and the statement follows. □

Analysis of policies using Theorem 6 must be done on a case by case basis as it depends on the policy description. For the UBT policy, only nodes on fire in the boundary will be treated so $a_j^t = 0$ $\forall t$ in (3.16). At each time t, this policy chooses boundary nodes with uniform probability without replacement and so each node has probability $C/|\mathcal{B}^t|$ of being treated. Therefore, if the following holds,

$$\forall t \in \mathbb{Z}_{>0} \quad \forall i \in \mathcal{B}^t, j \in \mathcal{N}(i): \quad p_{ij} - \frac{C}{|\mathcal{B}^t|}\Delta p_{ij}(a_i^t = 1, a_j^t = 0) \leq \frac{1}{2},$$

then the UBT policy is stabilizing according to Theorem 6. While this condition cannot be computed a priori, it can serve as real-time feedback to indicate if the resource limit C is insufficient to stabilize the process.

For the DWT policy, there is no simple analytical description of the probability for weighted sampling without replacement [73]. For the DWT, RHT, and UST policies, (3.22) must be evaluated

Algorithm 2 Estimated Volatility

function $V_{\text{est}}(r, k)$
  if $k = 0$ **then return** $\frac{\sum_{c \in \mathcal{C}^t(r)} p_{rc}}{|\mathcal{C}^t(r)|}$
  else return $\sum_{c \in \mathcal{C}^t(r)} V_{\text{est}}(c, k-1)$

Algorithm 3 Urban Safety Treatment (UST)

if fire can reach an urban area in k generations **then**
  Remove up to C urban areas that are reachable by at least one fire, with priority determined by the number of fires that reach a given urban area.
  Remove fires with $V_{\text{est}} < k$ from consideration for treatment with remaining control.
Use remaining control to treat nodes in the boundary $\mathcal{B}^t$ with the highest ranking according to,
  1. proximity to the right-most lattice edge and,
  2. estimated volatility (Algorithm 2).

at each time step to determine if each policy is stabilizing. We present simulation results for the previously described policies in the next section.

3.5 Simulation Experiments

We compare our control methods on simulations of forest and urban wildfires and the 2014 West Africa Ebola outbreak. We call our approximate linear programming framework Approximate Constrained Scalable Allocation of Resources (ACSAR). We defined several rule-based policies in Section 3.4 and we refer to them by their respective names.

3.5.1 ACSAR Performance

We first benchmark our method on forest wildfires simulations. For all simulation experiments of wildfires, model parameters of $\alpha_i = 0.2 \ \forall i \in \mathcal{V}$ and $\beta_i = 0.9 \ \forall i \in \mathcal{V}$ were used for the wildfire model (Chapter 2.5) with the linear-type tree state dynamics (Table 2.1). We use a forest size of 50×50 which has 10^{1192} total states and at the initial time step, all trees are healthy except for a 4×4 grid of fires in the center of the forest. Simulations terminate when there are no more fires. The control effectiveness parameter used was $\Delta\beta_i = 0.54 \ \forall i \in \mathcal{V}$, the discount factor was $\gamma = 0.95$, and the control capacity was $C = 4$.

We use the following reward and basis functions in conjunction with our approximate value

function approach,

$$r_i(x^t_{i\cup\mathcal{N}(i)}) = \mathbb{I}(x^t_i = H) - \mathbb{I}(x^t_i = F)\sum_{j\in\mathcal{N}(i)}\mathbb{I}(x^t_j = H),$$
$$w_i^\intercal h_i(x^t_{i\cup\mathcal{N}(i)}) = [w_i]_0 + [w_i]_1\,\mathbb{I}(x^t_i = H) + [w_i]_2\,\mathbb{I}(x^t_i = F)\sum_{j\in\mathcal{N}(i)}\mathbb{I}(x^t_j = H),$$

where $w_i \in \mathbb{R}^3\ \forall i \in \mathcal{V}$. Furthermore, we assume that every tree has four neighbors, $|\mathcal{N}(i)| = 4\ \forall i \in \mathcal{V}$, so that there is one equivalence class. The derived policy is then applied to the original graph model. The Program (3.5) requires computing an expectation of the basis functions, which is conditioned on a given configuration of the states $x^t_{i\cup\mathcal{N}(i)\cup\mathcal{N}(\mathcal{N}(i))}$. The result is the following expression after applying the dynamics in Table 2.1,

$$\begin{aligned}\mathbb{E}_p\left[w_i^\intercal h_i(x^{t+1}_{i\cup\mathcal{N}(i)})\right] = {}& [w_i]_0 + [w_i]_1\,\mathbb{I}(x^t_i = H)p(x^{t+1}_i = H \mid x^t_i, x^t_{\mathcal{N}(i)}) + \\ & [w_i]_2\Big(\mathbb{I}(x^t_i = H)p(x^{t+1}_i = F \mid x^t_i, x^t_{\mathcal{N}(i)}) + \\ & \quad \mathbb{I}(x^t_i = F)p(x^{t+1}_i = F \mid x^t_i, a^t_i)\Big)\sum_{j\in\mathcal{N}(i)}\mathbb{I}(x^t_j = H)p(x^{t+1}_j = H \mid x^t_j, x^t_{\mathcal{N}(j)}).\end{aligned} \tag{3.23}$$

The resulting approximate linear program is then,

$$\begin{aligned}\underset{\substack{w_i\in\mathbb{R}^3\\ \phi_i\in\mathbb{R}}}{\text{minimize}} \quad & \phi_i \\ \text{subject to} \quad & \phi_i \geq [w_i]_0 + [w_i]_1\,\mathbb{I}(x^t_i = H) + [w_i]_2\,\mathbb{I}(x^t_i = F)\sum_{j\in\mathcal{N}(i)}\mathbb{I}(x^t_j = H) \\ & \quad - \mathbb{I}(x^t_i = H) + \mathbb{I}(x^t_i = F)\sum_{j\in\mathcal{N}(i)}\mathbb{I}(x^t_j = H) \\ & \quad - \gamma\,(\text{Eq. (3.23)})\,, a^t_i = 0, \forall\ x^t_{i\cup\mathcal{N}(i)\cup\mathcal{N}(\mathcal{N}(i))}. \\ & \phi_i \geq -[w_i]_0 - [w_i]_1\,\mathbb{I}(x^t_i = H) - [w_i]_2\,\mathbb{I}(x^t_i = F)\sum_{j\in\mathcal{N}(i)}\mathbb{I}(x^t_j = H) \\ & \quad + \mathbb{I}(x^t_i = H) - \mathbb{I}(x^t_i = F)\sum_{j\in\mathcal{N}(i)}\mathbb{I}(x^t_j = H) \\ & \quad + \gamma\,(\text{Eq. (3.23)})\,\forall\ x^t_{i\cup\mathcal{N}(i)\cup\mathcal{N}(\mathcal{N}(i))}, a^t_i.\end{aligned}$$

To implement this program, we exploit Anonymous Influence to drastically reduce the number of constraints that must be specified. Computing the expectation (3.10) to determine the control policy then yields the following action weights,

$$\mu_i = -\gamma\,[w_i]_2\,\mathbb{I}(x^t_i = F)\Delta\beta\sum_{j\in\mathcal{N}(i)}\mathbb{I}(x^t_j = H)(1 - \alpha f^t_j).$$

Table 3.1: ACSAR results for different value and state-action function approximations. Data are the median percent of remaining healthy trees over 100 simulations, with the subscript and superscript denoting the first and third quartile, respectively. Without control, the majority of the forest burns down. Our control approach is much more effective and results in lower approximation error, compared to prior work.

Control Method	Approximation Error ϕ_i	Remaining Healthy Trees
No Control	—	$1.0^{+0.0}_{-0.0}\%$
Prior $V_w(x^t)$ [22]	2.30	$1.2^{+0.4}_{-0.3}\%$
ACSAR $V_w(x^t)$	1.98	$98.4^{+0.4}_{-0.6}\%$
ACSAR $Q_w(x^t, a^t)$	0.84	$98.4^{+0.4}_{-0.7}\%$

Solving the program yields $[w_i]_2 = -1.43$, and so this policy treats fires in priority of the number of neighboring healthy trees and their likelihood of remaining healthy at the next time step. We compare our approach with the basis approximation proposed by Forsell et al. [22], which can also be used with our capacity constrained formulation 3.8. The basis functions are,

$$w_i^\intercal h_i^{\text{prior}}(x_i^t) = [w_i]_0 \mathbb{I}(x_i^t = H) + [w_i]_1 \mathbb{I}(x_i^t = F) + [w_i]_2 \mathbb{I}(x_i^t = B),$$

The action weight for the policy (3.9) when using this basis with the same reward function as before is,

$$\mu_i = \gamma \Delta \beta \mathbb{I}(x_i^t = F)([w_i]_2 - [w_i]_1).$$

Therefore, the resulting policy is to randomly treat trees on fire at each time step. We also construct an approximate Q_w function using ACSAR to illustrate a complementary approach. We use the following reward and basis functions,

$$\begin{aligned}
r_i(x_i^t, x_i^{t+1}) &= \mathbb{I}(x_i^t = H) - (1 - a_i^t)\mathbb{I}(x_i^{t+1} = F), \\
w_i^\intercal b_i(x_i^t) &= [w_i]_0 + [w_i]_1 \mathbb{I}(x_i^t = H) + [w_i]_2 \mathbb{I}(x_i^t = F), \\
a_i^t w_i^\intercal c_i(x_{i \cup \mathcal{N}(i)}^t) &= a_i^t [w_i]_3 \mathbb{I}(x_i^t = F) \sum_{j \in \mathcal{N}(i)} \mathbb{I}(x_j^t = H),
\end{aligned}$$

where $w_i \in \mathbb{R}^4 \ \forall i \in \mathcal{V}$.

Table 3.1 summarizes the results for ACSAR using two different basis functions as well as the basis functions from prior work. We present the median percent of remaining healthy trees, along with the first and third quartile, to summarize the performance of each method. Overall, our basis functions are significantly more effective than those proposed in prior work, and the approximation error is lower as well. A comparison of the resulting policies from both our value function approximation and the prior work value function approximation is shown in Fig. 3.3.

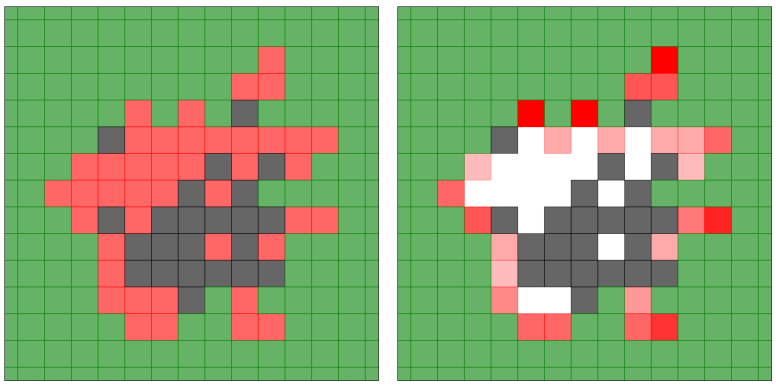

Figure 3.3: Comparison of policies from prior work and ACSAR. (both) Policies for a single time step of a wildfire simulation. Green cells are healthy trees and black are burnt trees. Fire color indicates control preference: white is low and dark red is high. (left) Policy using basis from prior work which treats all fires equally. (right) Our policy which prioritizes fires based on number of neighboring healthy trees.

We also apply our control framework to the 2014 West Africa Ebola outbreak, a graph-based model which we introduced in Chapter 2.5. We use the parameters $\alpha_i = 0.14\ \forall i \in \mathcal{V}$ and $\Delta\beta_i = 0.12\ \forall i \in \mathcal{V}$. We derived the α_i parameter from data [55] by fitting an exponential model to the cumulative number of cases per area, normalized by the area's population; see Fig. 3.4. We aggregate the exponential models across all areas to get a single parameter value to generate a more accurate parameter estimate. We generate an approximate Q_w function with the following reward and basis functions,

$$
\begin{aligned}
&r_i(x_i^t, x_i^{t+1}) = \mathbb{I}(x_i^t = S) - \mathbb{I}(x_i^{t+1} = E),\\
&w_i^\intercal b_i(x_i^t) = [w_i]_0 + [w_i]_1 \mathbb{I}(x_i^t = S) + [w_i]_2 \mathbb{I}(x_i^t = E),\\
&a_i^t w_i^\intercal c_i(x_{i\cup\mathcal{N}(i)}^t) = a_i^t [w_i]_3 \mathbb{I}(x_i^t = E) \sum_{j\in\mathcal{N}(i)} \mathbb{I}(x_j^t = S),
\end{aligned}
$$

with $w_i \in \mathbb{R}^4\ \forall i \in \mathcal{V}$. As a simplification, we assume that each community has the same number of neighbors, $|\mathcal{N}(i)| = 4\ \forall i \in \mathcal{V}$, so there is a single equivalence class. The derived policy is then applied to the original graph model.

A total of 100 simulations were run, with a capacity constraint of $C = 3$ and a discount factor of $\gamma = 0.9$. Each simulation is initialized with three communities being infected: Guieckedou (Guinea), Kailahun (Sierra Leone), and Lofa (Liberia); all other communities are initially healthy. Simulations terminate when no communities are infected and the performance metric z_{sim} for each simulation is the median number of weeks a community is infected, i.e., $z_{\text{sim}} = \text{median of } \{y_i = \sum_{\tau=1}^{T} \mathbb{I}(x_i^\tau = E) \mid i \in \mathcal{V}\}$. Solving the approximate Q_w ALP yields $\phi_i = 0.28$, and the median and maximum value of z_{sim} over 100 simulations were 32 weeks and 102 weeks, respectively.

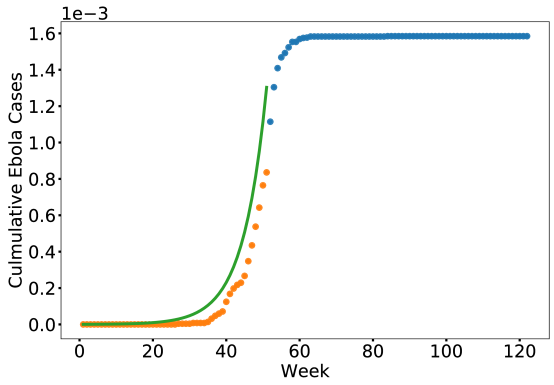

Figure 3.4: Example of fitting parameters for the 2014 West Africa Ebola outbreak model. Cumulative Ebola cases for Bomi in Liberia, normalized by population. Data (orange circles) are used to fit an exponential model (green line) from which we derive the model parameter α_i.

From the dataset, we also estimate the median time each community spent with active cases, by estimating the week when cases started increasing and the week when cases leveled off for each community. The difference between these quantities is the length of time each community was considered infected. Over all communities, the median infection time was 36 weeks and the maximum infection time was 112 weeks. Therefore, our approach has low approximation error and closely matches the dataset.

3.5.2 Percolation Framework Performance

We now evaluate our percolation-based framework on simulations of a forest wildfire near an urban, which we introduced a model for in Chapter 2.5. The lattice is a square of size 50×50 nodes, the right edge is composed of 500 urban areas, and the remaining nodes are trees. Each simulation is initialized with fires at the center as shown in Fig. 3.5. The initial set of fires was chosen so that at the first time step, there are a number of "interior" fires which will only spread for a few time steps before extinguishing. The parameters α_i and β_i were varied across the lattice as shown in Fig. 3.6 and the control effectiveness parameter was set to $\Delta\beta_i = 0.35\ \forall i \in \mathcal{V}$.

A total of $1,000$ total simulations were run with two different resource limits, $C = 6$ and $C = 10$, for each policy. For a given policy, the fraction of remaining healthy trees and remaining healthy urban areas were recorded at the end of each simulation run. Simulations ended when there were no more nodes on fire. For the UST policy, the fraction of removed urban areas was also recorded. For the RHT policy, two horizons were tested, $k = 1$ and $k = 3$, and for the UST policy the horizon was $k = 5$.

Tables 3.2 and 3.3 present the median of the results and Fig. 3.7 provides the full distribution of the results for each policy. Without control, the process is supercritical and Tables 3.2 and 3.3

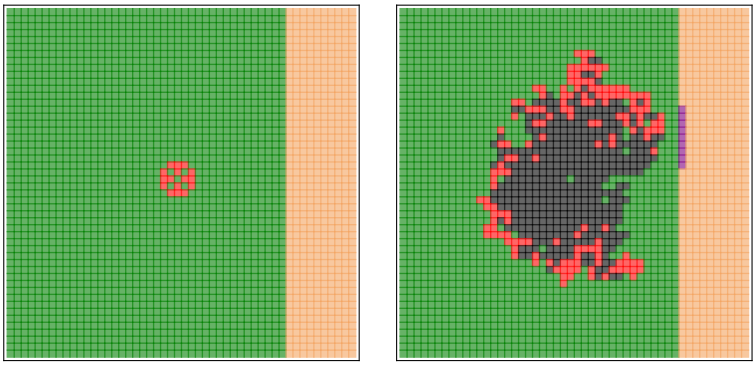

Figure 3.5: Details for simulation experiments with the percolation framework. (both) Green nodes are healthy trees, red are on fire, black are burnt, and light brown are healthy urban areas. (left) Initial condition for simulations. (right) Example snapshot of the UST policy where purple indicates removed urban areas. By removing urban areas, the UST policy prevents other urban areas from catching on fire.

show that the majority of the nodes are eventually burnt.

The UBT and DWT policies are not very effective with the lower resource limit $C = 6$ as they do not account for heterogeneous properties or for potential interior fires. However, with the increased capacity $C = 10$ these policies are more successful and are able to preserve more trees and urban areas. The RHT policy easily outperforms the UBT and DWT policies for both resource limits due to directly considering differences in the percolation parameter. However, this policy is only effective in preserving urban areas for the higher resource limit. In contrast, the UST policy is either equally or more effective than all other policies in preserving both trees and urban areas. Even for a low resource limit, UST preserves the majority of urban areas as desired. Fig. 3.5 shows an example snapshot of the lattice state for a single simulation while using the UST policy. The policy starts to remove urban areas once fire spreads too closely thus preventing urban areas from being burnt.

The distributions of the results, shown in Fig. 3.7, show that the median does not adequately capture the performance of each policy. All policies have large variance, although the RHT and UST policies significantly improve the likelihood that a majority of trees and urban areas are preserved. Only the UST policy reliably preserves urban areas for both resource limits as evidenced by the much lower variance, although this comes with the trade off of more trees being burnt and some urban areas being removed. Fig. 3.7 also provides a sense of how many times a given policy was stabilizing for all time steps. Policies with large variance in the results did not frequently meet the requirements of Theorem 6 over $1,000$ simulations.

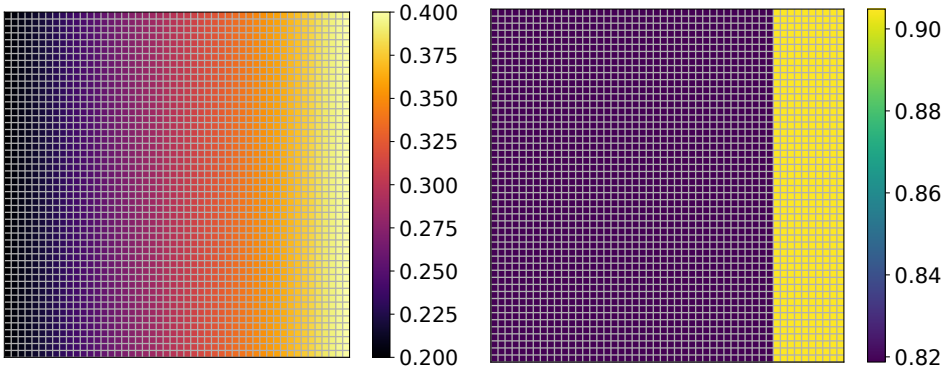

Figure 3.6: Parameter values for simulation experiments with the percolation framework. (left) Values of α_i for all nodes on the lattice. (right) Values of β_i for all nodes on the lattice. These parameter values correspond to a supercritical forest wildfire.

Table 3.2: Percolation results for resource limit $C = 6$ and $1,000$ simulations. The median fraction of removed urban areas for the UST policy is 3.10%.

Method	Median Remaining Healthy Trees (%)	Median Remaining Healthy Urban Areas (%)
No Control	3.80	0.00
UBT	13.25	0.02
DWT	16.32	0.20
RHT $k = 1$	22.78	1.00
$k = 3$	38.02	0.60
UST	38.78	96.90

3.6 Summary

In this chapter, we presented solution techniques for generating a control policy that strictly satisfies a capacity constraint, given that the underlying state of the graph-based Markov decision process is fully observable. Our first approach, based on approximate dynamic programming, has the benefit of an approximation error metric which directly provides feedback on the quality of the chosen basis function approximation. However, additional structure must be enforced on the basis functions in order to reduce the computational complexity of solving the linear programs and to ensure that a constrained policy can be tractably extracted from the approximate value or state-action function. In addition, it is non-trivial to deal with heterogeneous models where the model parameters or the discrete Markov models may be unique on a per-node basis. Our second approach, based on branching processes and bond percolation, allows us to easily define rule-based policies and address heterogeneous model properties. However, additional approximations are required and

Table 3.3: Percolation results for resource limit $C = 10$ and $1,000$ simulations. The median fraction of removed urban areas for the UST policy is 0.40%.

Method	Median Remaining Healthy Trees (%)	Median Remaining Healthy Urban Areas (%)
No Control	3.80	0.00
UBT	56.07	1.80
DWT	85.42	98.90
RHT $k = 1$	88.27	100.00
$k = 3$	90.70	100.00
UST	88.90	99.60

it is not straightforward to analyze the stability of the process under a given policy. We demonstrated through simulation results that both of our approaches are scalable to considerably large graph-based models, and that our methods are more effective than other proposed approaches in literature. For future directions, policy iteration methods [23] could be adapted so that ACSAR can handle heterogeneous model parameters while still providing an approximation error metric, and so that additional structure is not needed in order to produce a constrained control policy. For the percolation framework, it is difficult to analyze the process stability models without a lattice structure, and extending this approach to more general graph structures would be useful. Finally, we hope to apply these control techniques to additional application domains, especially with cooperative multi-agent teams, to provide an additional tool for team-based decision making strategies.

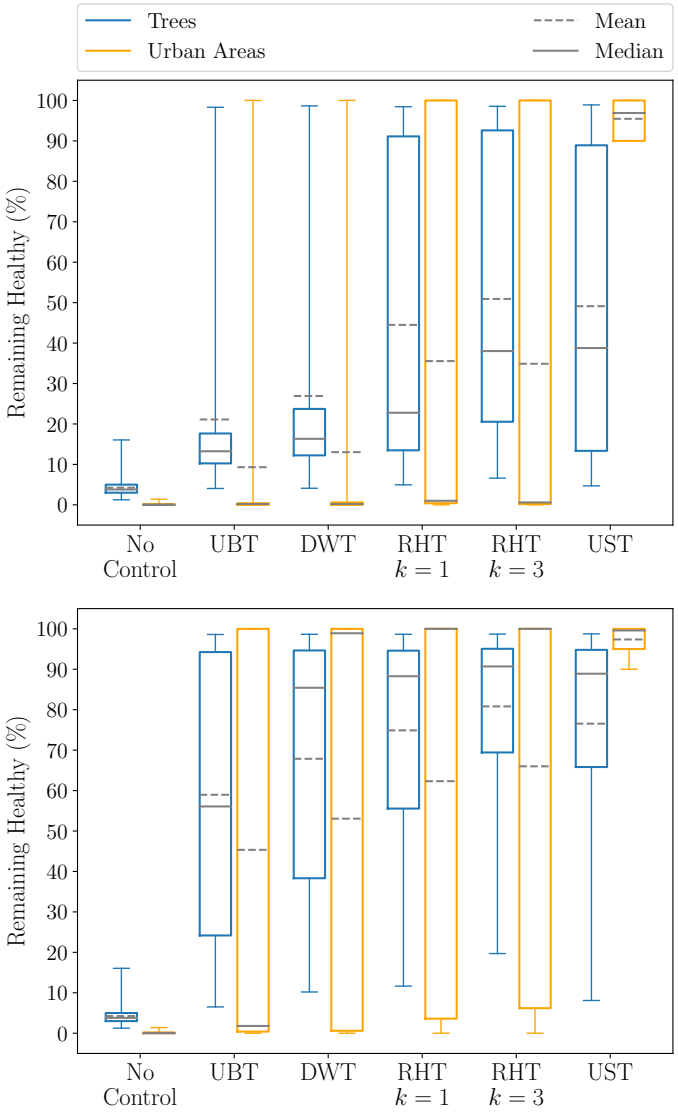

Figure 3.7: Percolation framework performance, presented as box and whisker plots over $1,000$ simulations with resource limit $C = 6$ (top) and $C = 10$ (bottom) for the different policies. The whiskers represent the minimum and maximum and the box shows the first quartile, mean, median, and third quartile. Only the UST policy is capable of reliably preserving the urban areas as other policies show a large variance in their performance.

Chapter 4

Fast Online Filtering

In this chapter, we introduce state uncertainty in the graph-based Markov decision process (GMDP) model and consider the problem of producing state estimates online as measurements are taken. We propose an approach based on variational inference, where a structured distribution is introduced to approximate the true posterior distribution at each time step. The result is a message-passing scheme that is tractable for the large state and observation spaces that typically arise in natural phenomena modeled by GMDPs, and we show that our approach is faster than and comparably accurate to loopy belief propagation. Furthermore, we also demonstrate the need for a fast and accurate online filtering scheme by developing a certainty-equivalence approach in conjunction with the control methods we discussed in Chapter 3. By using our filtering method, we show that our control policies perform comparably to the case where the state is fully observable. The material in this chapter appears in publications [31, 32].

4.1 Introduction

We begin this chapter by considering the problem of producing sequential state estimates online in the absence of control actions for the discrete time and discrete space graph-based Markov decision problem (GMDP) which we introduced in Chapter 2.4.1. Considering state uncertainty in the GMDP framework is necessary to address realistic natural phenomena and online sequential inference for large GMDPs requires approximate methods due to the large state and observation spaces. For this reason, we leverage variational inference to derive a suitable approach to address these challenges. The online aspect creates further challenges, as any candidate method must provide a belief at each time step in a reasonable amount of computation time. Prior work has presented many variations and improvements on variational inference methods, belief propagation methods, and many other filtering methods. However, other methods typically incur additional complexity or require model additional structure and as a result are not appropriate for the GMDP models we consider. We

review relevant literature in Section 4.2.

Our filtering approach is based on introducing an approximation to the evidence-based lower bound (ELBO), and we prove the approximation is itself a lower bound to the ELBO. We then leverage the mean-field approximation in variational inference to derive a message-passing scheme, which is similar in spirit to belief propagation methods. We show that our scheme is considerably faster than comparable methods, such as loopy belief propagation, while being at least as accurate on simulations of forest wildfires and the 2014 West Africa Ebola outbreak.

In this chapter, we also consider the problem of producing a control action, subject to a capacity constraint, given noisy measurements. In Chapter 3, we considered this same problem under the assumption of perfect state knowledge instead of noisy measurements. Considering uncertainty in the control problem is necessary to apply our method to more realistic problem descriptions and we develop a certainty-equivalence approach to provide a single framework capable of addressing realistic phenomena which naturally contain state uncertainty. To the best of our knowledge, our approach is the first framework to consider GMDPs with a control constraint and measurement uncertainty.

While the partially observable Markov decision process (POMDP) framework is appropriate for this problem, it is difficult to develop approximately-optimal methods that are suitable for the model sizes we consider using existing POMDP tools. In addition, any candidate method must also run in (near) real-time to be useful in the applications we consider, similar to the online filtering problem. Therefore, we separate the problem into using our filtering approach to produce accurate state estimates, and then using our control methods with the state estimate to determine a constrained control action. We show that this approach is performs comparably to the control results when the underlying state is observable.

The methods we present in this chapter are most appropriate for GMDPs with two properties common in large-scale spatial processes, called "Anonymous Influence" and "Symmetry." A GMDP has Anonymous Influence if the state dynamics of a given MDP relies on the number of influencing MDPs in particular states, and not the identity of these influencing MDPs. Symmetry refers to the insight that value approximations for a given MDP can frequently be reused for other MDPs in the GMDP, which greatly reduces the computational complexity of our control methods.

The remainder of this chapter is organized as follows. We review related work in Section 4.2. In Section 4.3, we derive our message-passing filtering scheme based on variational inference. In Section 4.4 we formulate our certainty-equivalence framework which combines our filter method with our approximate linear programming control approach from Chapter 3. Section 4.5 present simulation results demonstrating the performance of our filter and our certainty-equivalence approach in comparison with relevant methods from literature. We provide concluding remarks in Section 4.6.

4.2 Related Work

For the models we introduced in Chapter 2.5, the equivalent graphical model representation typically contains many cycles. Therefore, methods that rely on a tree structure (e.g., belief propagation) or assume few cycles cannot be directly applied. Furthermore, we are interested in producing a full posterior distribution over states to quantify certainty in the state estimate, in contrast to a maximum-likelihood estimate (e.g., Viterbi algorithm).

Particle filters can address some issues of online inference for GMDP models [74]. However, the number of particles required for a given accuracy increases with the state dimension [75] which is generally intractable for GMDPs. Proposed approaches that have addressed this issue [76, 77, 78] are appropriate for continuous dynamical system models and not the discrete state space models we consider. Other methods [26, 79, 27] have been applied to relatively large models but do not scale to the model sizes we consider.

Variational inference (VI) methods have been applied to relatively large discrete models for inference [80, 81, 82, 83]. Notably, semi-implicit VI [83] optimizes bounds of the evidence lower bound (ELBO), but these bounds are not suitable for our approach and thus we develop our own approximation. While some methods [84] perform approximate inference for large datasets, it is unclear how to adapt them for online use as applications have been limited to relatively small models [85]. Stochastic gradient methods typically require a differentiable distribution whereas we estimate arbitrary discrete distributions. Other methods [86] are based on exploiting distribution structure which we do not require.

Belief propagation (BP) methods can be derived using variational inference with energy approximations (e.g., Bethe or Kikuchi). Loopy belief propagation (LBP) has been shown to be effective in some discrete loopy graphical models [9] and we use LBP as a benchmark method. Generalized belief propagation (GBP) improves upon LBP [10] but incurs additional (worst-case exponential) complexity and is non-trivial to apply generally. We emphasize that our approach does not use these energy approximations.

In contrast, our filtering approach uses a logarithm approximation to develop a message-passing scheme that approximates the Kullback–Leibler divergence typically used in VI methods. The result of this approach is a combination of the computational efficiency of message-passing methods with the theoretical insight and effectiveness of variational inference approaches.

For the problem of computing constrained control policies under state uncertainty, POMDPs are the most appropriate framework and methods have been proposed for structured models, based on algebraic decision diagrams [20, 87], factored value functions [21, 88], and policy graphs [89]. Notably, Poupart et al. [87] are able to solve models with approximately 10^6 states, but this is still much smaller than the models we wish to address. While prior work suggests some appealing approximation methods, it is not clear how to adopt them for online use. Furthermore, while exploration actions help improve the state belief, they may sacrifice some performance in controlling

the process. We wish to maximize the effectiveness of the control since we aim to control large-scale natural disasters like forest wildfires and disease epidemics. Contrary to POMDP methods, which are dominated by the interleaving of filtering and control, we intentionally separate the two. This leads to our methods being able to scale up to problems with 10^{1192} possible discrete states, well beyond existing POMDP methods.

4.3 Variational Message-Passing Filter Scheme

We now derive our filtering scheme. The objective of a filter at a single time step is to produce the posterior distribution $p(x^t \mid y^{1:t})$ where $y^{1:t}$ is the history of measurements up to time t, $y^{1:t} = \{y^1, \dots, y^t\}$. We discussed the exact filter in Chapter 2.4.1, and we include it here for convenience,

$$p(x^t \mid y^{1:t}) \propto p(y^t \mid x^t) \sum_{x^{t-1}} p(x^t \mid x^{t-1}, a^{t-1}) p(x^{t-1} \mid y^{1:t-1}),$$

which is initialized by a prior at the initial time step, $p(x^1)$. Similarly, given the structure of the GMDP model discussed in Chapter 2.4.1, the recursive Bayesian filter (RBF) [90] is,

$$p(x^t \mid y^{1:t}) \propto \Big(\prod_{i=1}^{n} p_i(y_i^t \mid x_i^t) \Big) \Big(\sum_{x^{t-1}} p(x^{t-1} \mid y^{1:t-1}) \prod_{i=1}^{n} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) \Big). \tag{4.1}$$

The above expression does not simplify to a tractable form as we allow for arbitrary graph structure. In particular, the graphical model representation of the graph G may contain many undirected cycles (or loops), as is the case for the lattice-based graph in our wildfire model. See Fig. 4.1 for a graphical model representation of a simple GMDP as well as its equivalent Dynamic Bayesian Network representation.

Variational inference (VI) methods formulate an optimization problem to approximate an intractable posterior distribution [91]. The RBF (4.1) requires $(\prod_{i=1}^{n} |\mathcal{X}_i|) - 1$ values to specify $p(x^{t-1} \mid y^{t-1})$ and is intractable to compute for even a single time step despite the graph structure. For example, our wildfire model with 250 total trees has 10^{119} total states. Therefore, we use VI to introduce a family of distributions $q(x^t) \in \mathcal{Q}$ to approximate the posterior $p(x^t \mid y^{1:t})$ which ideally minimizes the Kullback-Leibler (KL) divergence between the two distributions. However, directly minimizing the KL divergence is infeasible due to requiring knowledge of the posterior. Instead, a tractable optimization is maximizing the evidence lower bound (ELBO) which indirectly minimizes the KL divergence. The ELBO is,

$$\text{ELBO} = \mathbb{E}_q \left[\log p(x^t, y^t \mid y^{1:t-1}) - \log q(x^t) \right], \tag{4.2}$$

where $\mathbb{E}_q$ indicates the expectation is taken with respect to the distribution $q(x^t)$. The ELBO is

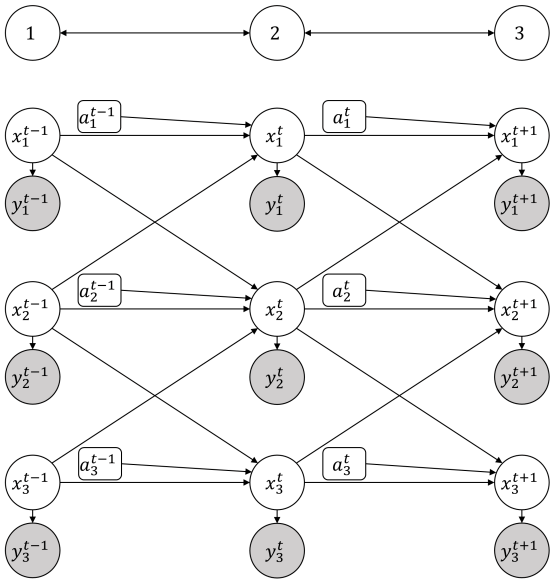

Figure 4.1: (top) An example GMDP consisting of three vertices, each of which represents an MDP, where arrows indicate the mutual influence between MDPs. (bottom) The underlying graphical model of the example GMDP, where arrows indicate influence between time steps.

derived by considering a distribution to approximate the intractable posterior. By using Jensen's inequality for the concave logarithm function, the following relationship holds,

$$\log p(y^{1:t}) = \log \mathbb{E}_q \left[\frac{p(x^t, y^t \mid y^{1:t-1})}{q(x^t)} \right] \geq \mathbb{E}_q \left[\log \frac{p(x^t, y^t \mid y^{1:t-1})}{q(x^t)} \right] = \text{ELBO}.$$

Therefore, optimizing the ELBO directly optimizes the probability of observing the history of measurements. Choosing an appropriate form for the approximating distribution $q(x^t)$ results in an optimization problem with a tractable solution, as we explain next.

We leverage the mean-field approximation where the approximating distribution is factored, $q(x^t) = \prod_{i=1}^{n} q_i(x_i^t)$, and a discrete distribution (or variational factor) is associated with each HMM in the GHMM. This approximation reduces the representation size of the posterior and leads to,

$$\text{ELBO} = \sum_{x^t} \Big(\prod_{i=1}^{n} q_i(x_i^t) \Big) \log p(x^t, y^t \mid y^{1:t-1}) - \sum_{i=1}^{n} \sum_{x_i^t} q_i(x_i^t) \log q_i(x_i^t),$$

after substitution of $q(x^t)$ and algebraic simplification. A common approach, known as coordinate ascent VI, finds a local optimum by iteratively optimizing each factor $q_i(x_i^t)$ while holding others fixed. The update expression for each factor is derived by collecting the terms in the ELBO which

involve factor $q_i(x_i^t)$,

$$\text{ELBO} = \sum_{x_i^t} q_i(x_i^t)\mathbb{E}_{-i}\left[\log p(x^t, y^t \mid y^{1:t-1})\right] - \sum_{x_i^t} q_i(x_i^t)\log q_i(x_i^t) + \text{other terms}, \tag{4.3}$$

where $\mathbb{E}_{-i}$ refers to the expectation taken with respect to the distribution $q(x^t)$ excluding factor $q_i(x_i^t)$, i.e., $\prod_{j=1,j\neq i}^{n} q_j(x_j^t)$. For the optimization of (4.3) over a single factor, the "other terms" are dropped as they are constant with respect to factor $q_i(x_i^t)$. As a result, the expression in (4.3) can be rewritten as the following objective function,

$$\mathcal{L}_i = -D_{KL}(q_i(x_i^t) \,||\, \exp\mathbb{E}_{-i}\left[\log p(x^t, y^t \mid y^{1:t-1})\right]), \tag{4.4}$$

where D_{KL} is the KL divergence. Since the KL divergence is non-negative and equals zero when the argument distributions are identical, maximizing $\mathcal{L}_i$ leads to the closed-form update expression,

$$q_i(x_i^t) \propto \exp\mathbb{E}_{-i}\left[\log p(x^t, y^t \mid y^{1:t-1})\right]. \tag{4.5}$$

We emphasize that arbitrary graph structure prevents simplification of the previous expression through structure in $\log p(x^t, y^t \mid y^{1:t-1})$.

4.3.1 Approximating the ELBO

For GHMMs with the mean-field assumption, the coordinate ascent update (4.5) for a single time step requires computing the joint probability,

$$\begin{aligned} p(x^t, y^t \mid y^{1:t-1}) &\propto p(y^t \mid x^t)\sum_{x^{t-1}} p(x^t \mid x^{t-1}, a^{t-1})p(x^{t-1} \mid y^{1:t-1}) \\ &\propto \Big(\prod_{i=1}^{n} p_i(y_i^t \mid x_i^t)\Big)\sum_{x^{t-1}}\prod_{i=1}^{n} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1})u_i(x_i^{t-1}), \end{aligned} \tag{4.6}$$

where $r(x^{t-1}) = \prod_{i=1}^{n} u_i(x_i^{t-1}) \approx p(x^{t-1} \mid y^{t-1})$ is the approximate factored prior distribution. Computing this quantity is typically intractable due to the required marginalization of all n HMMs.

Instead, a tractable computation of $\mathbb{E}_{-i}\left[p(y^t, x^t \mid y^{1:t-1})\right]$ is possible using a message-passing scheme. We first discuss necessary approximations to the ELBO (4.2) and then describe the scheme. We assume the joint probability satisfies a lower bound, $p(y^t, x^t \mid y^{1:t-1}) \geq \epsilon$ for some $0 < \epsilon < 1$. Given a bound ϵ, an under-approximation to the logarithm function over the interval $[\epsilon, 1]$ is the line,

$$g(\theta) = \frac{\log\epsilon}{1-\epsilon}(1-\theta), \tag{4.7}$$

and $g(\theta) \leq \log\theta$ for $\theta \in [\epsilon, 1]$. Using (4.7) to approximate $\log p(x^t, y^t \mid y^{1:t-1})$ in (4.2) results in a

surrogate ELBO,

$$\overline{\text{ELBO}} = \mathbb{E}_q \left[g\left(p(x^t, y^t \mid y^{1:t-1}) \right) - \log q(x^t) \right]. \tag{4.8}$$

Theorem 7. *Given $\epsilon > 0$ such that $\epsilon \leq p(x^t, y^t \mid y^{1:t-1}) \leq 1$, the surrogate ELBO (4.8) is a lower bound to the original ELBO (4.8).*

Proof. The difference between the surrogate ELBO (4.8) and the ELBO (4.2) is,

$$\begin{aligned}
\text{ELBO} - \overline{\text{ELBO}} &= \mathbb{E}_q \left[\log p(x^t, y^t \mid y^{1:t-1}) - \log q(x^t) \right] - \mathbb{E}_q \left[g\left(p(x^t, y^t \mid y^{1:t-1}) \right) - \log q(x^t) \right] \\
&= \mathbb{E}_q \left[\log p(x^t, y^t \mid y^{1:t-1}) \right] - \mathbb{E}_q \left[g\left(p(x^t, y^t \mid y^{1:t-1}) \right) \right] \\
&\geq 0, \ \forall x^t, y^t
\end{aligned}$$

The expectation operator is linear and thus preserves the lower bound relationship of the approximation (4.7) to the logarithm function. The lower bound is valid for any combination of states x^t and measurements y^t as the joint probability is bounded below by ϵ. □

Maximizing the surrogate ELBO (4.8) over the factors $q_i(x_i^t)$ therefore indirectly maximizes the ELBO (4.3). Following the same derivation for the ELBO, the factor objective (4.4) for the surrogate ELBO is,

$$\hat{\mathcal{L}}_i = -D_{KL}(q_i(x_i^t) \mid\mid \exp \mathbb{E}_{-i} \left[g\left(p(x^t, y^t \mid y^{1:t-1}) \right) \right]).$$

The coordinate update (4.5) changes to,

$$\begin{aligned}
q_i(x_i^t) &\propto \exp \mathbb{E}_{-i} \left[g\left(p(x^t, y^t \mid y^{1:t-1}) \right) \right] \\
&\propto \exp g\left(\mathbb{E}_{-i} \left[p(x^t, y^t \mid y^{1:t-1}) \right] \right),
\end{aligned} \tag{4.9}$$

and the factors are now a function of the expectation of the joint probability, as desired, due to the linear approximation.

Remark. Imposing a lower bound on the joint probability (4.6) precludes combinations of states and observations that have zero probability of occurring. In practice, we round estimates of (4.6) lower than ϵ up to ϵ, which has the effect of introducing noise into the joint probability. For large GHMM models, probabilities naturally tend to zero, e.g., the aggregate state distribution (2.9) and observation distribution (2.10), since the product of probabilities less than one will approach zero. This approximation can therefore be seen as preventing the expectation in (4.9) from being zero for all states x_i^t prior to updating the posterior factor $q_i(x_i^t)$. In addition, after updating a posterior factor with (4.9), state probabilities lower than ϵ are rounded to zero before normalizing the distribution. This is used to preserve the idea that some state transitions must be considered impossible, e.g., a healthy tree transitioning to on fire without any neighboring trees being on fire. We show through numerical simulations in Section 4.5 that this approach is effective. Finally, ϵ is

a tuning parameter and is chosen to be a small positive value to avoid excessively influencing the posterior factors.

4.3.2 Message-passing Scheme

We now build a tractable message-passing scheme to estimate the quantity $\mathbb{E}_{-i}\left[p(x^t, y^t \mid y^{1:t-1})\right]$ required in (4.9) for each coordinate update. Substituting (4.6) leads to,

$$\begin{aligned}\mathbb{E}_{-i}\left[p(x^t, y^t \mid y^{1:t-1})\right] \propto \sum_{\{x_j^t | j\in\mathcal{V}, j\neq i\}} \Big(\prod_{\substack{j=1\\ j\neq i}}^{n} q_j(x_j^t)\Big)\Big(\prod_{i=1}^{n} p_i(y_i^t \mid x_i^t)\Big) \\ \Big(\sum_{x^{t-1}} \prod_{i=1}^{n} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) u_i(x_i^{t-1})\Big).\end{aligned} \tag{4.10}$$

The message-passing scheme works as follows. Each HMM in the GHMM maintains an estimate of its posterior factor, $q_i^k(x_i^t)$, and an estimate of (4.10), $E_i^k(x_i^t)$; the superscript k on these quantities, and other quantities below, refers to the k^{th} estimate. For each iteration k of the scheme, each HMM i receives messages from the neighbor HMMs $j \in \mathcal{N}(i)$ and generates estimate $E_i^k(x_i^t)$. This estimate then updates the posterior factor using (4.9). Lastly, an updated message is calculated for the next iteration $k+1$.

The development of a message-passing scheme requires recognizing a recursive structure in the information that must be shared for each HMM to compute (4.6). We illustrate this structure and derive the required messages by describing the first two iterations of the scheme for a given HMM i.

The first iteration E_i^1 considers information from the neighbor HMMs $j \in \mathcal{N}(i)$,

$$E_i^1(x_i^t) \propto \Big[p_i(y_i^t \mid x_i^t)\Big] \sum_{x_i^{t-1}} u_i(x_i^{t-1}) \Bigg[\sum_{x_{\mathcal{N}(i)}^{t-1}} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) \prod_{j\in\mathcal{N}(i)} u_j(x_j^{t-1})\Bigg]. \tag{4.11}$$

The second iteration E_i^2 considers information from the neighbors $\mathcal{N}(i)$ and the neighbors of neighbors $\bigcup_{j\in\mathcal{N}(i)} \mathcal{N}(j)$,

$$\begin{aligned}E_i^2(x_i^t) \propto\ & p_i(y_i^t \mid x_i^t) \sum_{x_i^{t-1}} u_i(x_i^{t-1}) \sum_{x_{\mathcal{N}(i)}^{t-1}} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) \prod_{j\in\mathcal{N}(i)} u_j(x_j^{t-1}) \sum_{x_j^t} q_j^1(x_j^t) \\ & \Big[p_j(y_j^t \mid x_j^t) \sum_{x_{\mathcal{N}(j)}^{t-1}} p_j(x_j^t \mid x_j^{t-1}, x_{\mathcal{N}(j)}^{t-1}, a_j^{t-1}) \prod_{l\in\mathcal{N}(j)} u_l(x_l^{t-1})\Big].\end{aligned} \tag{4.12}$$

Note that the above expression assumes that HMM i is not a part of the neighbors of HMMs $j \in \mathcal{N}(i)$, i.e., that $i \notin \mathcal{N}(j)\ \forall j \in \mathcal{N}(i)$. This is a simplifying assumption to approximately include information from the neighbor set, as we do not assume any simplifying structure; see the Remark at the end of Section 4.3.3. There is a common structure in the first (4.11) and second (4.12) iterations,

as indicated by the large brackets in both expressions. Define d_i^1 as the following quantity,

$$d_i^1(x_i^{t-1}, x_i^t) = p_i(y_i^t \mid x_i^t) \sum_{x_{\mathcal{N}(i)}^{t-1}} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) \prod_{j \in \mathcal{N}(i)} u_j(x_j^{t-1}). \tag{4.13}$$

The second iteration (4.11) then simplifies to,

$$E_i^1(x_i^t) \propto \sum_{x_i^{t-1}} u_i(x_i^{t-1}) d_i^1(x_i^{t-1}, x_i^t). \tag{4.14}$$

In addition, the second iteration (4.12) simplifies to,

$$\begin{aligned} E_i^2(x_i^t) \propto p_i(y_i^t \mid x_i^t) \sum_{x_i^{t-1}} u_i(x_i^{t-1}) \sum_{x_{\mathcal{N}(i)}^{t-1}} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) \\ \prod_{j \in \mathcal{N}(i)} \left[u_j(x_j^{t-1}) \sum_{x_j^t} q_j^1(x_j^t) d_j^1(x_j^{t-1}, x_j^t) \right], \end{aligned} \tag{4.15}$$

by using information computed by the neighbor HMMs $j \in \mathcal{N}(i)$ during the first iteration of the scheme, d_j^1. Notably, the simplified second iteration (4.15) now only requires information from the neighbors $\mathcal{N}(i)$, as indicated by brackets. If the neighbors produce a message to share,

$$m_j^1(x_j^{t-1}) \propto u_j(x_j^{t-1}) \sum_{x_j^t} q_j^1(x_j^t) d_j^1(x_j^{t-1}, x_j^t),$$

then the second iteration (4.15) further simplifies,

$$E_i^2(x_i^t) \propto \left[p_i(y_i^t \mid x_i^t) \right] \sum_{x_i^{t-1}} u_i(x_i^{t-1}) \left[\sum_{x_{\mathcal{N}(i)}^{t-1}} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) \prod_{j \in \mathcal{N}(i)} m_j^1(x_j^{t-1}) \right]. \tag{4.16}$$

Finally, (4.16) shares a common structure with the first iteration (4.11), as indicated by brackets. If d_i^2 is defined as the following quantity for the second iteration,

$$d_i^2(x_i^{t-1}, x_i^t) = p_i(y_i^t \mid x_i^t) \sum_{x_{\mathcal{N}(i)}^{t-1}} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) \prod_{j \in \mathcal{N}(i)} m_j^1(x_j^{t-1}), \tag{4.17}$$

then the second iteration E_i^2 simplifies again to,

$$E_i^2(x_i^t) \propto \sum_{x_i^{t-1}} u_i(x_i^{t-1}) d_i^2(x_i^{t-1}, x_i^t),$$

which mirrors the form of the simplified first iteration (4.14). By initializing the messages as the

prior for all HMMs, $m_i^0(x_i^{t-1}) = u_i(x_i^{t-1})$, the quantity d_i^k can be written generally as,

$$d_i^k(x_i^{t-1}, x_i^t) = p_i(y_i^t \mid x_i^t) \sum_{x_{\mathcal{N}(i)}^{t-1}} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) \prod_{j \in \mathcal{N}(i)} m_j^{k-1}(x_j^{t-1}). \tag{4.18}$$

Subsequent iterations E_i^k, $k \geq 3$ continue to incrementally add influence from additional HMMs in the GHMM to improve the estimate of (4.10).

Using the general form of d_i^k, the estimate of the joint probability computed by HMM i at iteration k is,

$$\begin{aligned} E_i^k(x_i^t) &\propto \sum_{x_i^{t-1}} u_i(x_i^{t-1}) d_i^k(x_i^{t-1}, x_i^t) \\ &\approx \mathbb{E}_{-i}\left[p(x^t, y^t \mid y^{1:t-1})\right], \end{aligned} \tag{4.19}$$

and $q_i^k(x_i^t)$ is updated by using the above estimate with (4.9). Lastly, the updated message that is shared by each HMM at the next iteration is,

$$m_i^k(x_i^{t-1}) \propto u_i(x_i^{t-1}) \sum_{x_i^t} q_i^k(x_i^t) d_i^k(x_i^{t-1}, x_i^t). \tag{4.20}$$

Using (4.7) moves the expectation into the argument of (4.9) which allows the posterior factors to be used for marginalization of (4.10). This property is key for creating a tractable message-passing method. Otherwise, the quantity $\mathbb{E}_{-i}\left[\log p(x^t, y^t \mid y^{1:t-1})\right]$ is not tractable to compute as we do not allow for simplification based on the graph structure of the GHMM, on the properties of distributions for the posterior factors, or other properties. In addition, the estimates (4.19) are normalized to estimate the normalization constant of the joint probability (4.6), since the approximation (4.7) does not allow this constant to be factored out.

The previous derivation is based on a model where each MDP measurement is conditionally independent of other MDPs given its own state, i.e., measurement models of the form $p(y_i^t \mid x_i^t)$. In general, MDP measurements may be influenced by other MDPs as well, and we provide a derivation for the model $p(y_i^t \mid x_i^t, x_{\mathcal{N}(i)}^t)$ in the Appendix to consider a broader class of GMDP models with state uncertainty.

4.3.3 Simplifying with Anonymous Influence

Computing d_i^k (4.18) may be intractable as marginalizing out $x_{\mathcal{N}(i)}^{t-1}$ requires considering $\prod_{j \in \mathcal{N}(i)} |\mathcal{X}_j|$ values. If a HMM has many neighbors (large $|\mathcal{N}(i)|$) or if the neighbors have large state spaces $|\mathcal{X}_j|$ then the computational cost may be significant. Therefore, we now exploit Anonymous Influence to address this potential issue. If the dynamics (2.8) of a HMM rely on a count aggregator (CA) (as shown in (2.15)), then it is useful to create a mixed-mode function (MMF) $\tilde{m}_i^{k-1}(z_i^{t-1})$ to represent

Algorithm 4 Relaxed Anonymous Variational Inference (RAVI) for time step t

1: **Input:** prior factors $u_i(x_i^{t-1})$, graph G, actions a_i^{t-1}, dynamics $p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1})$, measurements y_i^t and measurement models $p_i(y_i^t \mid x_i^t)$
2: **Output:** posterior factors $q_i(x_i^t)$
3: **Parameters:** iteration limit $K_{\max}$, lower bound ϵ, convergence criteria
4: **for** each vertex i **do**
5: initialize message $m_i^0(x_i^{t-1}) = u_i(x_i^{t-1})$
6: initialize factor $q_i^0(x_i^t)$
7: **for** iteration $k = 1, \ldots, K_{\max}$ **do**
8: **for** each vertex $i \in \mathcal{V}$ **do**
9: Receive messages $\{m_j^{k-1}(x_j^{t-1}) \mid j \in \mathcal{N}(i)\}$
10: Compute $d_i^k(x_i^{t-1}, x_i^t)$ with (4.18) or (4.21)
11: Estimate $E_i^k(x_i^t)$ using (4.19)
12: Update $q_i^k(x_i^t)$ by (4.9)
13: Compute $m_i^k(x_i^{t-1})$ with (4.20)
14: **if** factors $q_i^k(x_i^t)$ converge **then** terminate early
15: **return** posterior factors $q_i(x_i^t) = q_i^k(x_i^t)$

the received neighbor messages. Using this MMF leads to the modified form of d_i^k,

$$d_i^k(x_i^{t-1}, x_i^t) = p_i(y_i^t \mid x_i^t) \sum_{z_i^{t-1}} p_i(x_i^t \mid x_i^{t-1}, z_i^{t-1}, a_i^{t-1}) \tilde{m}_i^{k-1}(z_i^{t-1}). \tag{4.21}$$

The marginalization for (4.21) is now with respect to z_i^{t-1} which has lower computational cost.

Algorithm 4, Relaxed Anonymous Variational Inference (RAVI), summarizes the approximate filter for a single time step. The factors $q_i(x_i^t)$ are then used as the priors $u_i(x_i^t)$ for the next time step. The main component is the message-passing scheme, which is relatively straightforward to implement. The posterior factors are initialized to any valid discrete distribution (line 6) and the algorithms runs for a fixed number of iterations $K_{\max}$ unless the factors converge (line 14).

Remark. Our filtering approach is based on two key approximations, an approximate lower bound on the joint probability (4.6) and a message-passing scheme to approximate the expectation of the joint probability (4.10). Variational Inference techniques commonly rely on approximations and simplifications, such as other bounds of the ELBO [83], the mean-field approximation [91], conjugate distributions [92], or the presence of tractable substructures [93]. These approximations are necessary to reduce the optimization of the ELBO to a tractable optimization. Furthermore, Loopy Belief Propagation (LBP) approximates computing marginals on a cyclic graph with a message-passing scheme, and has been shown to be accurate in a variety of applications [9]. Our focus in this work is to develop a probabilistic approach with a focus on scalability and performance, which we demonstrate with our simulation results in Section 4.5.

4.3.4 Additional Measurement Model Derivation

We now consider measurement models of the form $p_i(y_i^t \mid x_i^t, x_{\mathcal{N}(i)}^t)$. The form of (4.10) is then,

$$\mathbb{E}_{-i}\left[p(x^t, y^t \mid y^{1:t-1})\right] \propto \sum_{\{x_j^t | j\in\mathcal{V}, j\neq i\}} \Big(\prod_{\substack{j=1\\ j\neq i}}^{n} q_j(x_j^t)\Big)\Big(\prod_{i=1}^{n} p_i(y_i^t \mid x_i^t, x_{\mathcal{N}(i)}^t)\Big)$$
$$\Big(\sum_{x^{t-1}} \prod_{i=1}^{n} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) u_i(x_i^{t-1})\Big).$$

The first iteration is,

$$E_i^1(x_i^t) \propto \Bigg[\sum_{x_{\mathcal{N}(i)}^t} p_i(y_i^t \mid x_i^t, x_{\mathcal{N}(i)}^t) \prod_{j\in\mathcal{N}(i)} q_j^0(x_j^t)\Bigg]$$
$$\sum_{x_i^{t-1}} u_i(x_i^{t-1})\Bigg[p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) \prod_{j\in\mathcal{N}(i)} u_j(x_j^{t-1})\Bigg],$$

and the second iteration is,

$$E_i^2(x_i^t) \propto \sum_{x_{\mathcal{N}(i)}^t} p_i(y_i^t \mid x_i^t, x_{\mathcal{N}(i)}^t) \prod_{j\in\mathcal{N}(i)} q_j^1(x_j^t) \sum_{x_i^{t-1}} u_i(x_i^{t-1}) p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) \prod_{j\in\mathcal{N}(i)} u_j(x_j^{t-1})$$
$$\Bigg[\sum_{x_{\mathcal{N}(j)}^t} p_j(y_j^t \mid x_j^t, x_{\mathcal{N}(j)}^t) \prod_{l\in\mathcal{N}(j)} q_l^1(x_l^t) p_j(x_j^t \mid x_j^{t-1}, x_{\mathcal{N}(j)}^{t-1}, a_j^{t-1}) \prod_{l\in\mathcal{N}(j)} u_l(x_l^{t-1})\Bigg].$$

Following the same derivation as before and using the common structure in the previous two expressions indicated by brackets, the quantity d_i^k is,

$$d_i^k(x_i^{t-1}, x_i^t) = \sum_{x_{\mathcal{N}(i)}^t} p_i(y_i^t \mid x_i^t, x_{\mathcal{N}(i)}^t) \sum_{x_{\mathcal{N}(i)}^{t-1}} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}, a_i^{t-1}) \prod_{j\in\mathcal{N}(i)} m_j^{k-1}(x_j^{t-1}, x_j^t).$$

Messages are now initialized as,

$$m_i^0(x_i^{t-1}, x_i^t) = u_i(x_i^{t-1}) q_i^0(x_i^t).$$

The k^{th} estimate update (4.9) and message update (4.20) remain the same for this case. Similar to the derivation in Section 4.3.2, influence from additional HMMs is approximately added to estimates of (4.10), as we do not consider special graph structure. In addition, the Anonymous Influence property can be exploited if either the dynamics (2.8) or the measurement distribution $p_i(y_i^t \mid x_i^t, x_{\mathcal{N}(i)}^t)$ (or both) can be represented using mixed-mode functions (MMFs).

Algorithm 5 Certainty-Equivalence Framework

Offline
Solve linear program(s) for basis functions weights of approximate function $V_w(x^t)$ or $Q_w(x^t)$.

Online
for each time step t **do**
    Get measurement.
    Update state estimate using filter with measurement.
    Solve linear program to get constrained control action for the state estimate.
    Apply control action.

4.4 Constrained Control Under Measurement Uncertainty

Now that we have developed a process filter to produce accurate maximum-likelihood estimates of the underlying state of a GMDP, we now describe a certainty-equivalence framework to enable the use of our control methods. In particular, we combine our filter with our approximate linear programming control approach, called ACSAR, which we developed in Chapter 3. Algorithm 5 provides a high-level overview of our framework. First, a value or state-action function approximation is produced offline by solving linear programs to determine the weights of the chosen basis functions. Second, our fast filter produces online estimates of the process state, which is used in a linear program to determine a control action that satisfies a capacity constraint.

4.5 Simulation Experiments

For all simulation experiments, we use the forest wildfire model with the linear-type tree dynamics, which we introduced in Chapter 2.5. The model parameters $\alpha = 0.2 \ \forall i \in \mathcal{V}$ and $\beta_i = 0.9 \ \forall i \in \mathcal{V}$ were used, with a forest size of 50×50 trees that has 10^{1192} total states. At the initial time, all trees are healthy except for a 4×4 grid of fires in the center of the forest. Simulations terminate when there are no more trees on fire.

The measurement model for each tree $p_i(y_i^t \mid x_i^t)$ is parameterized by the probability p_c, which is the probability that the ground truth tree state is observed. The other two tree states are observed with probability $\frac{1}{2}(1 - p_c)$. The measurement model is thus,

$$p(y_i^t \mid x_i^t) = \begin{cases} p_c & \text{if } y_i^t = x_i^t, \\ \frac{1}{2}\,(1 - p_c) & \text{otherwise.} \end{cases} \tag{4.22}$$

4.5.1 Filter Performance

We compare our filter RAVI against loopy belief propagation (LBP) which we adapt for online sequential estimation by limiting the time history to a maximum of length $H = 3$. If adding a new

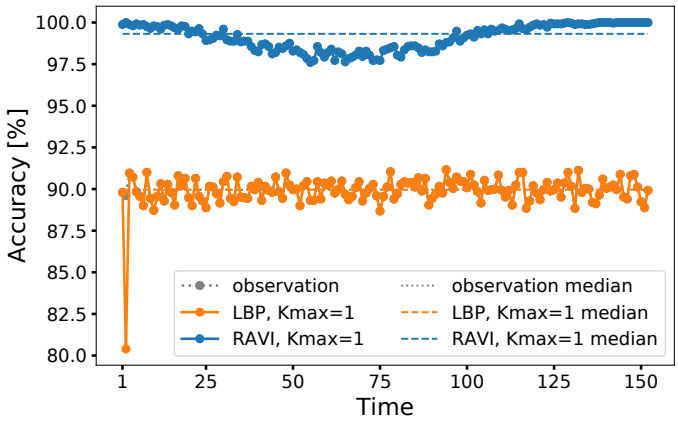

Figure 4.2: Example filter results for a single model simulation. The simulation accuracy for a filter is the median accuracy over the entire time series. Here, LBP is the same as taking as the measurement as the estimate, as it overlays the measurement accuracy in the plot. In contrast, RAVI is 9% better.

time step will exceed this limit, the model is reinitialized with only the latest time step and the prior belief is from the first time step of the previous model. For RAVI, the value of ϵ was chosen to be 10^{-10}. Both RAVI and LBP were terminated early if less than 1% of the posterior factors stop changing their maximum likelihood belief. We also compare against taking each measurement as the estimate for each time step, which may produce inconsistent estimates, e.g., a tree transitioning from healthy to burnt in one time step. All filters were initialized with the ground truth.

At each time step of the simulation, the belief produced by the filter is converted to a maximum likelihood estimate and compared to the true state. We compute the percentage of trees whose states are correctly estimated as the "accuracy" of the estimator at that time step. The result is a time history of accuracies for a single simulation, as shown in Fig. 4.2. We compute the median accuracy for the time history which we call the simulation accuracy. We run 10 total simulations and report the first quartile, the median, and the third quartile simulation accuracy. Table 4.1 shows the results for different message passing iteration limits and measurement model accuracies p_c in (4.22). While $p_c = 0.9$ may seem like a very accurate measurement model, there is a $0.9^{2500} \approx 10^{-115}$ probability that the true ground state is observed. For the lowest limit $K_{\max} = 1$, LBP is slow and is the same accuracy as simply taking the measurement as the estimate. While we showed in prior work that LBP can be effective when given enough iterations [31], it does not scale to the model size in this work and cannot be used online. For example, the 2018 Camp wildfire in Northern California at one point was spreading at a rate of 80 acres per minute [94]. At this rate, a wildfire burns 184 acres in 138 seconds, compared to 5.33 acres in 4 seconds. Therefore, it is critical to use a fast, accurate online filter to enable an effective response to natural disasters.

In contrast, RAVI is effective even for the low iteration limit, and improves slightly given more

Table 4.1: Filter results for two different measurement accuracies p_c in (4.22). Data are the median simulation accuracy for 10 simulations, and the subscript and superscript indicate the first and third quartiles, respectively. LBP improves with more iterations but is slow while RAVI is accurate and fast enough to be used online.

Method		Measurement Accuracy		Estimate Time (seconds)
		80%	90%	
Measurement		$80.0^{+0.1}_{-0.1}\%$	$90.0^{+0.2}_{-0.1}\%$	—
LBP	$K_{\max} = 1$	$80.0^{+0.0}_{-0.0}\%$	$90.0^{+0.0}_{-0.0}\%$	138.73
	$K_{\max} = 2$	$85.0^{+0.8}_{-1.1}\%$	$94.5^{+0.9}_{-0.8}\%$	251.43
RAVI	$K_{\max} = 1$	$98.0^{+0.4}_{-0.3}\%$	$99.4^{+0.1}_{-0.2}\%$	1.41
	$K_{\max} = 5$	$98.6^{+0.2}_{-0.2}\%$	$99.5^{+0.0}_{-0.1}\%$	4.25
	$K_{\max} = 10$	$98.6^{+0.2}_{-0.2}\%$	$99.5^{+0.0}_{-0.1}\%$	4.23

iterations. Coordinate ascent methods are known to find local optima and RAVI quickly finds a solution which does not significantly change with more iterations. In particular, the time to produce an estimate drops for $K_{\max} = 10$ which is likely due to slightly more accurate posterior factors at earlier time steps in each simulation.

4.5.2 Closed-loop Filter and Controller Performance

We now present simulation results to demonstrate the performance of our closed-loop filter and control approach, which we call RAVI ACSAR.

Given the filtering results in Table 4.1, we use the measurement and RAVI (with $K_{\max} = 5$) and $p_c = 0.9$ as filtering methods. We use ACSAR to generate both an approximate value function and an approximation state-action function, and we also compare with a method from prior work for approximate value functions. The control effectiveness parameter used was $\Delta\beta_i = 0.45 \ \forall i \in \mathcal{V}$, the discount factor was $\gamma = 0.95$, and the control capacity was $C = 5$.

We use the following reward and basis functions in conjunction with our approximate value function approach,

$$r_i(x^t_{i\cup\mathcal{N}(i)}) = \mathbb{I}(x^t_i = H) - \mathbb{I}(x^t_i = F) \sum_{j\in\mathcal{N}(i)} \mathbb{I}(x^t_j = H),$$
$$w_i^\intercal h_i(x^t_{i\cup\mathcal{N}(i)}) = [w_i]_0 + [w_i]_1 \mathbb{I}(x^t_i = H) + [w_i]_2 \mathbb{I}(x^t_i = F) \sum_{j\in\mathcal{N}(i)} \mathbb{I}(x^t_j = H),$$

and we use the following reward and basis functions with our approximate state-action function

Table 4.2: Results for two filter methods and three choices of basis approximations. Data are the median percent of remaining healthy trees over 100 simulations, with the subscript and superscript denoting the first and third quartile, respectively. Without control, the majority of the forest burns down. An accurate filter is required, as otherwise control effort is wasted on trees that are believed to be on fire but are actually healthy or burnt. Only our filtering method RAVI and our control approach ACSAR is successful in preserving the majority of trees in the forest.

Filter Method	Control Method	Remaining Healthy Trees
—	No Control	$1.0^{+0.0}_{-0.0}\%$
Measurement	Prior Work [22] $V_w(x^t)$	$1.3^{+0.3}_{-0.3}\%$
	ACSAR $V_w(x^t)$	$2.2^{+0.5}_{-0.3}\%$
RAVI $K_{\max} = 5$	Prior Work [22] $V_w(x^t)$	$1.9^{+1.7}_{-0.4}\%$
	ACSAR $V_w(x^t)$	$97.8^{+0.6}_{-1.0}\%$
	ACSAR $Q_w(x^t)$	$97.8^{+0.7}_{-1.0}\%$

approach,

$$
\begin{aligned}
r_i(x_i^t, x_i^{t+1}) &= \mathbb{I}(x_i^t = H) - (1 - a_i^t)\mathbb{I}(x_i^{t+1} = F), \\
w_i^\intercal b_i(x_i^t) &= [w_i]_0 + [w_i]_1 \, \mathbb{I}(x_i^t = H) + [w_i]_2 \, \mathbb{I}(x_i^t = F), \\
a_i^t w_i^\intercal c_i(x_{i \cup \mathcal{N}(i)}^t) &= a_i^t \, [w_i]_3 \, \mathbb{I}(x_i^t = F) \sum_{j \in \mathcal{N}(i)} \mathbb{I}(x_j^t = H).
\end{aligned}
$$

These are the same reward and basis functions we used in Chapter 3 to demonstrate the performance of ACSAR for the case when the underlying state is fully observable.

Table 4.2 summarizes the results for two different filtering methods in combination with the prior work basis functions and our basis functions. We present the median percent of surviving healthy trees, along with the first and third quartile, to summarize the performance of the overall framework. Without control, nearly the entire forest typically burns down. Although the measurement appears to be accurate enough for control, there are many cases where a tree is believed to be on fire but is actually healthy or burnt. As a result, control actions are wasted as treating a healthy or burnt tree has no effect. Only the combination of our filter and our value or state-action function basis approximation is successful in extinguishing the wildfire. Finally, we note that a higher capacity was used for the framework experiments ($C = 5$) compared to the ACSAR experiments in Chapter 3 ($C = 4$). More control is required to overcome the effects of occasionally wasting control effort on trees that are not on fire, but with an accurate filter, the increase is minimal.

4.6 Summary

In this chapter, we considered the case where the underlying state of a graph-based Markov decision process (GMDP) cannot be fully observed. Addressing uncertainty is critical for using GMDPs to model real-world processes and for applying decision-making algorithms. In particular, we leveraged approximations in the variational inference framework, such as a lower bound to the ELBO, to derive a message-passing scheme that is similar in spirit to belief propagation methods. We also exploited Anonymous Influence to further reduce the computational complexity, which is necessary in large graph models with many edges or large individual state spaces. We showed that our filter approach is around ten times faster than adopting loopy belief propagation for the online filtering problem while still maintaining high filter accuracy. Future work on filtering methods should focus on understanding the quality of the approximations we have made, and relaxing the structural assumptions to continue to move towards addressing general GMDP descriptions. In addition, we also constructed a certainty-equivalence framework to show that our filter is accurate enough to enable the use of our control methods without modifications. In particular, only a minimal increase in control effort is required to maintain a similar level of performance compared to the case where the underlying state is fully observable. For this framework, future work includes exploring factored POMDP methods to consider the full decision making problem without additional simplifying assumptions. While prior work has suggested some methods to achieve this, it is not clear how to best adopt these approaches for online use with models that have large state and observation spaces.

Chapter 5

Learning Model Parameters with Historical Data

So far in this thesis, we have considered graph-based Markov decision processes (GMDPs) where the model parameters have been specified a priori. Notably, we extracted parameters for our 2014 West Africa Ebola model using data from the World Health Organization (WHO) [55]. In this chapter, we formalize and consider the problem of learning the parameters of a GMDP model, specifically the state dynamics and measurement model parameters, using sequences of observations for each vertex in the graph. There exists many models and learning algorithms for graph-based process models, but almost all are either developed on a case-by-case basis or do not consider arbitrary influence models, as we do in this chapter. In addition, formulating a model learning algorithm completes our suite of algorithms for GMDPs; after learning the parameters of a model, we can directly consider the constrained control problem (Chapter 3) with or without state uncertainty (Chapter 4). Our approach is a coordinate-ascent style optimization approach, based on the Expectation-Maximization (EM) framework, which approximately optimizes the log-likelihood of the observed data. The result is a learning algorithm that is amenable to parallelization, and allows for efficient training of GMDPs with many vertices and long observation sequences. We demonstrate our algorithm on two real-world data sets, the daily count of Novel Coronavirus (COVID-19) cases by county in California, USA, and Twitter interactions on a topic over several days. Our experiments show that the resulting GMDP models explain the observed data better than a uncoupled model assumption. The material in this chapter appears in publication [33].

5.1 Introduction

In this chapter, we address the model learning problem for discrete time and discrete space graph-based Markov decision processes (GMDPs), which we introduced in Chapter 2.4.1, given a sequence of observations for each vertex in the graph. Many application domains have been modeled as a structured system of interacting Markov processes, such as recognition tasks [12, 95], biomedical data [13, 96, 97], disease epidemics [29, 26], forest wildfires [29], freeway traffic [98], music theory [11, 99], and social networks and user interactions [100, 48, 101, 102]. Structured Markov models have a number of benefits over standard Markov model formulations as well as other stochastic process modeling frameworks. First, structured representations can directly incorporate known structure in the process that generated the time series data. Second, with respect to other modeling frameworks, there is significant research on optimal control [16, 22, 25] and inference methods [26] for structured Markov models, which allow them to be deployed in real-world applications. In particular, by developing a model learning algorithm, we have a produced a complete algorithmic pipeline for considering real-world processes based on historical time series data by leveraging our prior work. We have considered the optimal control problem with capacity constraints [29, 30], the online state filtering problem [31], and the constrained control problem under measurement uncertainty [32]. In this chapter, we develop a model learning algorithm appropriate for large-scale GMDPs to allow us to apply our previous methods to models learned from publicly available data.

The Anonymous Influence property, which we discussed in Chapter 2.4.2, allows us to investigate different influence structures that may exist in real-world data. For example, to what extent does confirmation bias play a role in user activity and interactions in social networks? For a disease epidemic, does intra-community or inter-community transmission play a bigger role in daily case counts? We design simulation experiments to evaluate hypotheses like these by comparing learned GMDPs to learned models based on a completely independent process assumption.

Learning algorithms for structured Markov models, and in particular graph-based Markov models, have been previously developed in literature with early approaches based on mean-field networks [103]. For the most part, relevant prior methods have focused mainly on relatively simple coupled models, e.g., few number of individual processes, or different coupling assumptions, e.g., only the observations are coupled. In contrast, we consider arbitrarily coupling between the state dynamics of the individual MDPs when there is potentially a large number of MDPs in the GMDP. Our approach is based on the Expectation-Maximization optimization framework which leads to an approximately-optimal method with favorable complexity for large GMDPs.

The remainder of this chapter is organized as follows. We review relevant related work in Section 5.2. In Section 5.3, we formulate the model learning problem as an optimization statement, discuss the complexity of this statement, and provide background on the EM algorithm. We derive our approximate EM approach in Section 5.4 and discuss the resulting complexity of our approach. In Section 5.5, we introduce performance metrics for evaluating learned models, provide details on

two real-world data sets, and demonstrate the benefits of our approach over an uncoupled model assumption. We provide concluding remarks in Section 5.6.

5.2 Related Work

Model learning algorithms have been presented in literature for a variety of application domains and modeling assumptions, and we are particularly interested in discrete structured Markov models. An equivalent standard hidden Markov model (HMM) formulation can be created from a GMDP model, by taking the product space of the individual MDP state and measurement spaces. From there, standard approaches can be leveraged such as the Baum-Welch [104] algorithm, or for more general cases, the Expectation-Maximization (EM) framework [105]. However, these methods quickly become intractable for moderately sized GMDPs due to the curse of dimensionality, as there is an exponential dependence on the number of constituent MDPs. To address this challenge, the factorial HMM (FHMM) [11] and the coupled HMM (CHMM) [12] frameworks have been formulated, along with suitable methods to leverage model structure.

Several optimization frameworks have been applied to the learning problem for FHMMs and CHMMs, starting with approximate EM techniques [11] which were later expanded upon [106, 107]. Notably, the FHMM framework consists of coupling only between the measurements of the individual HMMs, and the state dynamics are independent. Other work has expanded on this limitation by considering state couplings and applying the EM framework. Specifically, in [99], the authors consider a hierarchical coupling structure and in [107], the authors consider a convex combination of simpler dynamical models. While these works are less general cases of the types of models we wish to consider, they nevertheless provide useful insight into parameter learning techniques.

The CHMM framework, in contrast, directly considers coupling interactions in the state dynamics, and initial learning methods were based on efficient exact techniques in the EM framework [12]. CHMMs quickly became popular for a number of applications and several techniques were developed, such as use sampling methods [96, 98], neural networks [95], and variable clustering [98]. Similar to the approach by Saul et al. [107], Zhong et al. [13] derived the EM-based learning algorithm for the case the joint transition probability in a CHMM is a linear combination of marginal probabilities. Notably, Raghavan et al. [48] proposed a simple CHMM consisting of two HMMs to model a social network, where one chain models a single user and the other chain models the aggregate influence exerted by other users. This idea is similar to Anonymous Influence, which we make use of in the GMDPs we consider in this chapter. The majority of these works also consider relatively simple CHMMs, with only a few interacting HMMs, whereas we wish to consider GMDPs with a significantly large number of MDPs.

Few works have directly considered the more general case of GMDPs, especially for the learning problem. Forsell et al. introduced the GMDP [22] and the authors proposed solution methods

for the optimal control problem. Dong et al. considered graph-based HMMs [26] to model the spread of an infection and the authors proposed a sampling-based inference method. To the best of our knowledge, our approach is the first to consider the model parameter learning problem for GMDPs. We specifically aim to address GMDPs with a large number of MDPs and arbitrary coupling structure. Finally, we note that other works that consider different models for coupled stochastic systems, such as other Markov model formulations [100, 101] and stochastic differential equations [102], are typically formulated for a specific application. In contrast, we wish to consider a more general class of models to draw connections between many different domains, and to provide additional insight and theoretical analysis.

5.3 Model Learning Problem

For the model learning problem, we wish to determine the parameters of a graph-based Markov decision process (GMDP), which we described in Chapter 2.4.1, with $\mathcal{V} = \{1, \ldots, n\}$ and n total MDPs. In particular, these parameters are the state dynamics $p_i(x_i^t \mid x_{i\cup\mathcal{N}}^{t-1})$ and the measurement model $p_i(y_i^t \mid x_i^t)$ for each vertex $i \in \mathcal{V}$. We represent the state dynamics for each vertex as a time-invariant matrix Λ_i and refer to specific elements by,

$$p_i(x_i^t \mid x_{i\cup\mathcal{N}(i)}^{t-1}) = [\Lambda_i]_{x_{i\cup\mathcal{N}(i)}^{t-1}, x_i^t}.$$

For the data sets we present in Section 5.5, we assume the measurements are real-valued, $y_i^t \in \mathbb{R}^d$ with dimension d, and we consider a generic measurement model in our algorithm derivation in the following section. Furthermore, we assume that the data consists of a sequence of measurements for each vertex, $\{y_i^2, \ldots, y_i^T \mid i \in \mathcal{V}\}$, over the set of uniformly-spaced time steps starting at $t = 2$ and ending at time $t = T$. We use the notation $y_i^{2:T}$ to refer to the measurements for a single vertex i and use $y^{2:T}$ to refer to the measurements for all vertices. Likewise, we use $x_i^{1:T}$ and $x^{1:T}$ to refer to state trajectories for a single vertex and for all vertices, respectively. Given a prior for each vertex at the initial time, $p_i(x_i^1)$, and a set of measurements for every vertex, $y^{2:T}$, the joint probability is,

$$p_\theta(y^{2:T}, x^{1:T}) = \prod_{i\in\mathcal{V}} p_i(x_i^1) \prod_{t=2}^{T} p_i(x_i^t \mid x_{i\cup\mathcal{N}(i)}^{t-1}) p_i(y_i^t \mid x_i^t), \tag{5.1}$$

where $\theta = \{\Lambda_i, p_i(y_i^t \mid x_i^t) \mid i \in \mathcal{V}\}$ collectively refers to the state dynamics and measurement model parameters for all vertices.

We consider optimizing the log-likelihood of the observed data $y^{2:T}$ over the set of state dynamics and measurement model parameters θ,

$$\underset{\theta}{\text{maximize}} \;\; \log p_\theta(y^{2:T}) = \underset{\theta}{\text{maximize}} \;\; \log \sum_{x^{1:T}} p_\theta(y^{2:T}, x^{1:T}), \tag{5.2}$$

which requires marginalizing all possible state sequences for every vertex in the graph from the joint probability distribution. As a result, exactly computing this maximization is intractable. Instead, a lower bound on this objective can be generated by using Jensen's inequality with a distribution over state trajectories $Q(x^{1:T})$,

$$\begin{aligned}
\log \sum_{x^{1:T}} p_\theta(y^{2:T}, x^{1:T}) &= \log \sum_{x^{1:T}} Q(x^{1:T}) \frac{p_\theta(y^{2:T}, x^{1:T})}{Q(x^{1:T})} \\
&\geq \sum_{x^{1:T}} Q(x^{1:T}) \log \frac{p_\theta(y^{2:T}, x^{1:T})}{Q(x^{1:T})} \\
&= \sum_{x^{1:T}} Q(x^{1:T}) \log p_\theta(y^{2:T}, x^{1:T}) - \sum_{x^{1:T}} Q(x^{1:T}) \log Q(x^{1:T}) \\
&= f(Q, \theta).
\end{aligned} \tag{5.3}$$

Given this lower bound, the Expectation-Maximization algorithm [105] performs coordinate ascent in $f(Q, \theta)$,

$$\begin{aligned}
&\text{Expectation (E-step): } Q^{k+1}(x^{1:T}) \leftarrow \arg\max_Q f(Q, \theta^k) \\
&\text{Maximization (M-step): } \theta^{k+1} \leftarrow \arg\max_\theta f(Q^{k+1}, \theta)
\end{aligned} \tag{5.4}$$

The E-step in this coordinate ascent approach has a closed-form solution. It is straightforward to show that the maximum is achieved when the approximating distribution is the conditional distribution of $x^{1:T}$, $Q(x^{1:T}) = p_{\theta^k}(x^{1:T} \mid y^{2:T})$, at which point the lower bound becomes an equality for a given θ^k. This yields the classic EM algorithm result,

$$\theta^{k+1} \leftarrow \arg\max_\theta \sum_{x^{1:T}} p_{\theta^k}(x^{1:T} \mid y^{2:T}) \log p_\theta(y^{2:T}, x^{1:T}). \tag{5.5}$$

In many cases, this approach is still computationally intractable as it requires computing the conditional distribution $p_{\theta^k}(x^{1:T} \mid y^{2:T})$ as well as marginalizations of this distribution. Therefore, we derive an approximate E-step to develop a tractable coordinate-ascent algorithm, which indirectly optimizes the log-likelihood of the observed data.

5.4 Approximate Expectation-Maximization Approach

Given an estimate of the model parameters θ^k, we approximate the true distribution over state trajectories $p(x^{1:T})$ by a distribution $Q(x^{1:T})$ which considers each MDP $i \in \mathcal{V}$ as an independent process given the observations,

$$Q(x^{1:T}) = \frac{1}{Z_Q} \prod_{i \in \mathcal{V}} p_i(x_i^1) \prod_{t=2}^{T} Q_i(x_i^t \mid x_i^{t-1}) p_i(y_i^t \mid x_i^t), \tag{5.6}$$

where Z_Q is a normalization constant,

$$Z_Q = \sum_{x^{1:T}} \prod_{i \in \mathcal{V}} p_i(x_i^1) \prod_{t=2}^{T} Q_i(x_i^t \mid x_i^{t-1}) p_i(y_i^t \mid x_i^t).$$

We parameterize Q_i by a time-varying bias $\phi_i^t \in \mathbb{R}^{|\mathcal{X}_i| \times |\mathcal{X}_i|}$ and,

$$Q_i(x_i^t \mid x_i^{t-1}) = \left[\phi_i^t\right]_{x_i^{t-1}, x^t}. \tag{5.7}$$

This bias term encapsulates the information from the Markov blanket of each node $i \in \mathcal{V}$ as required in the distribution $p(y^{2:T}, x^{1:T})$. Substituting this form of $Q(x^{1:T})$ into $f(Q, \theta^k)$ yields,

$$\begin{aligned} f(Q, \theta^k) &= \log Z_Q + \mathbb{E}_Q \left[\sum_{i \in \mathcal{V}} \log p_i(x_i^1) + \sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \log p_i(y_i^t \mid x_i^t) \right] \\ &\quad + \mathbb{E}_Q \left[\sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \log p_i(x_i^t \mid x_{i \cup \mathcal{N}(i)}^{t-1}) \right] - \mathbb{E}_Q \left[\sum_{i \in \mathcal{V}} \log p_i(x_i^1) \right] \\ &\quad - \mathbb{E}_Q \left[\sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \log Q_i(x_i^t \mid x_i^{t-1}) + \sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \log p_i(y_i^t \mid x_i^t) \right] \\ &= \log Z_Q + \sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \mathbb{E}_Q \left[\log p_i(x_i^t \mid x_{i \cup \mathcal{N}(i)}^{t-1}) - \log Q_i(x_i^t \mid x_i^{t-1}) \right] \end{aligned} \tag{5.8}$$

where several terms have canceled due to our choice of approximation. Working out the expectations leads to the following expressions,

$$\begin{aligned} &\mathbb{E}_Q \left[\log p_i(x_i^t \mid x_{i \cup \mathcal{N}(i)}^{t-1}) \right] \\ &\quad = \sum_{x_i^{t-1}} \sum_{x_i^t} p_Q(x_i^{t-1}, x_i^t \mid y_i^{2:T}) \sum_{x_{\mathcal{N}(i)}^{t-1}} \Big(\prod_{j \in \mathcal{N}(i)} p_Q(x_j^{t-1} \mid y_j^{2:T}) \Big) \log \left[\Lambda_i\right]_{x_{i \cup \mathcal{N}(i)}^{t-1}, x_i^t}, \\ &\mathbb{E}_Q \left[\log Q_i(x_i^t \mid x_i^{t-1}) \right] = \sum_{x_i^{t-1}} \sum_{x_i^t} p_Q(x_i^{t-1}, x_i^t \mid y_i^{2:T}) \log \left[\phi_i^t\right]_{x_i^{t-1}, x_i^t}. \end{aligned} \tag{5.9}$$

Substituting these expressions into the objective for the E-step results in,

$$\begin{aligned} f(Q, \theta^k) = \log Z_Q + \sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \sum_{x_i^{t-1}} \sum_{x_i^t} p_Q(x_i^{t-1}, x_i^t \mid y_i^{2:T}) \Big(&- \log \left[\phi_i^t\right]_{x_i^{t-1}, x_i^t} \\ &+ \sum_{x_{\mathcal{N}(i)}^{t-1}} \Big(\prod_{j \in \mathcal{N}(i)} p_Q(x_j^{t-1} \mid y_j^{2:T}) \Big) \log \left[\Lambda_i\right]_{x_{i \cup \mathcal{N}(i)}^{t-1}, x_i^t} \Big). \end{aligned} \tag{5.10}$$

To solve for the elements of the bias, we take the derivative of $f(Q, \theta^k)$ with respect to $\log\left[\phi_i^t\right]_{x_i^{t-1}, x_i^t}$ which yields,

$$\begin{aligned}
\frac{\partial f(Q,\theta^k)}{\partial \log\left[\phi^t\right]_{x_i^{t-1},x_i^t}} &= \frac{\partial \log Z_Q}{\partial \log\left[\phi_i^t\right]_{x_i^{t-1},x_i^t}} - p_Q(x_i^{t-1}, x_i^t \mid y_i^{2:T}) \\
&\quad + \frac{\partial p_Q(x_i^{t-1}, x_i^t \mid y_i^{2:T})}{\partial \log\left[\phi_i^t\right]_{x_i^{t-1},x_i^t}} \Big(- \log\left[\phi_i^t\right]_{x_i^{t-1},x_i^t} + \sum_{x_{\mathcal{N}(i)}^{t-1}} \Big(\prod_{j\in\mathcal{N}(i)} p_Q(x_j^{t-1} \mid y_j^{2:T}) \Big) \log\left[\Lambda_i\right]_{x_{i\cup\mathcal{N}(i)}^{t-1}, x_i^t} \Big) \\
&= \frac{\partial p_Q(x_i^{t-1}, x_i^t \mid y_i^{2:T})}{\partial \log\left[\phi_i^t\right]_{x_i^{t-1},x_i^t}} \left[- \log\left[\phi_i^t\right]_{x_i^{t-1},x_i^t} + \sum_{x_{\mathcal{N}(i)}^{t-1}} \Big(\prod_{j\in\mathcal{N}(i)} p_Q(x_j^{t-1} \mid y_j^{2:T}) \Big) \log\left[\Lambda_i\right]_{x_{i\cup\mathcal{N}(i)}^{t-1}, x_i^t} \right],
\end{aligned} \tag{5.11}$$

where we have used $\frac{\partial \log Z_Q}{\partial \log\left[\phi_i^t\right]_{x_i^{t-1},x_i^t}} = p_Q(x_i^{t-1}, x_i^t \mid y_i^{2:T})$ to simplify the expression. Setting the expression in brackets to zero leads to the following fixed-point equation,

$$\log\left[\phi_i^t\right]_{x_i^{t-1},x_i^t} = \sum_{x_{\mathcal{N}(i)}^{t-1}} \Big(\prod_{j\in\mathcal{N}(i)} p_Q(x_j^{t-1} \mid y_j^{2:T}) \Big) \log\left[\Lambda_i\right]_{x_{i\cup\mathcal{N}(i)}^{t-1}, x_i^t} . \tag{5.12}$$

We note here that using (5.12) to update the approximating distribution represented by ϕ_i^t requires enumerating $T|\mathcal{X}_i|^2 \prod_{j\in\mathcal{N}(i)}|\mathcal{X}_j|$ values, which may be intractable due to the number of observations T, the size of the neighbor set $|\mathcal{N}(i)|$, or the state space of the MDPs $|\mathcal{X}_i|$. Therefore, we now exploit Anonymous Influence to address this potential issue. By instead considering whether or not the neighboring MDPs $j \in \mathcal{N}(i)$ are in an influencing state, the fixed-point equation becomes,

$$\log\left[\phi_i^t\right]_{x_i^{t-1},x_i^t} = \sum_{z_i^{t-1}} p_Q\big(z_i^{t-1} \mid y_j^{2:T}, j \in \mathcal{N}(i)\big) \log\left[\Lambda_i\right]_{x_i^{t-1}, z_i^{t-1}, x_i^t} . \tag{5.13}$$

In the above expression, we make use of a count aggregator (CA) $z_i^{t-1} \in [0, 1, \ldots, |\mathcal{N}(i)|]$ to represent the influence of the MDPs $j \in \mathcal{N}(i)$ which has significantly lower computational cost. We also note that the distribution $p_Q\big(z_i^{t-1} \mid y_j^{2:T}, j \in \mathcal{N}(i)\big)$ is computed from the distributions $p_Q(x_j^t \mid y_j^{2:T}), j \in \mathcal{N}(i)$.

Solving the fixed-point equation (either (5.12) or (5.13)) requires iterating between the following two steps,

1. Compute the distributions $p_Q(x_i^{t-1}, x_i^t \mid y_i^{2:T})\ \forall i \in \mathcal{V}, t \in [2, T]$ given the estimates $\{\phi_i^t \mid i \in \mathcal{V}\}$;
2. Update the time-varying bias $\phi_i^t\ \forall i \in \mathcal{V}, t \in [2, T]$ given an estimate of the distributions.

The first step can be efficiently computed by using the Forward-Backward algorithm [104] with the model assumption in (5.6) which has time complexity $\mathcal{O}(T|\mathcal{X}_i|^2)$ for each vertex. The second step has time complexity $\mathcal{O}(T|\mathcal{X}_i|^2 \prod_{j\in\mathcal{N}(i)}|\mathcal{X}_j|)$, and with Anonymous Influence, this can be reduced to $\mathcal{O}(T|\mathcal{X}_i|^2|\mathcal{N}(i)|)$ for each vertex. We note that both steps can easily be parallelized and we take

advantage of this in our simulation experiments in Section 5.5. We alternate between these two steps until successive estimates of the time-varying biases converge, and the result is the approximating distribution $Q(x^{1:T}) = \prod_{i \in \mathcal{V}} Q_i(x_i^{1:T})$. With this distribution in hand, we can turn to the M-step to compute an update of the model parameters θ, as we discuss next.

The M-step objective is,

$$\begin{aligned} f(Q^{k+1}, \theta) &= \mathbb{E}_{Q^{k+1}}\left[\log p(y^{2:T}, x^{1:T})\right] \\ &= \underbrace{\sum_{i \in \mathcal{V}} \mathbb{E}_{Q^{k+1}}\left[\log p_i(x_i^1)\right]}_{\text{first term: prior}} + \underbrace{\sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \mathbb{E}_{Q^{k+1}}\left[\log p_i(x_i^t \mid x_{i \cup \mathcal{N}(i)}^{t-1})\right]}_{\text{second term: dynamics}} \\ &\quad + \underbrace{\sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \mathbb{E}_{Q^{k+1}}\left[\log p_i(y_i^t \mid x_i^t)\right]}_{\text{third term: measurement model}}, \end{aligned} \tag{5.14}$$

where we denote the estimate from the approximate E-step as Q^{k+1}; the entropy of Q^{k+1} is constant with respect to θ once the distribution is specified and therefore is not included in the objective. The first term of (5.14) is constant as the prior is assumed to be known and therefore can be dropped from the objective as well. The second term of (5.14) is,

$$\begin{aligned} &\sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \mathbb{E}_{Q^{k+1}}\left[\log p_i(x_i^t \mid x_{i \cup \mathcal{N}(i)}^{t-1})\right] \\ &= \sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \sum_{x_{i \cup \mathcal{N}(i)}^{t-1}} \sum_{x_i^t} \left(p_{Q^{k+1}}(x_i^{t-1}, x_i^t \mid y_i^{2:T}) \prod_{j \in \mathcal{N}(i)} p_{Q^{k+1}}(x_j^{t-1} \mid y_j^{2:T})\right) \log \left[\Lambda_i\right]_{x_{i \cup \mathcal{N}(i)}^{t-1}, x_i^t}. \end{aligned} \tag{5.15}$$

To derive an update expression for the state dynamics Λ_i, we introduce a Lagrange multiplier λ_i with the constraint $\sum_{x_i^t} \left[\Lambda_i\right]_{x_{i \cup \mathcal{N}(i)}^{t-1}, x_i^t} = 1 \ \forall x_{i \cup \mathcal{N}(i)}^{t-1}$, and solve the following equation,

$$\frac{\partial}{\partial \left[\Lambda_i\right]_{x_{i \cup \mathcal{N}(i)}^{t-1}, x_i^t}} \left[\text{Eq. (5.15)} + \lambda_i \Big(\sum_{x_i^t} \left[\Lambda_i\right]_{x_{i \cup \mathcal{N}(i)}^{t-1}, x_i^t} - 1 \Big)\right] = 0. \tag{5.16}$$

The result is the following update expression,

$$\left[\Lambda_i\right]_{x_{i \cup \mathcal{N}(i)}^{t-1}, x_i^t} = \frac{\sum_{t=2}^{T} p_{Q^{k+1}}(x_i^{t-1}, x_i^t \mid y_i^{2:T}) \prod_{j \in \mathcal{N}(i)} p_{Q^{k+1}}(x_j^{t-1} \mid y_j^{2:T})}{\sum_{t=2}^{T} p_{Q^{k+1}}(x_i^{t-1} \mid y_i^{2:T}) \prod_{j \in \mathcal{N}(i)} p_{Q^{k+1}}(x_j^{t-1} \mid y_j^{2:T})}. \tag{5.17}$$

We again note that we can exploit Anonymous Influence to reduce the computational complexity of updating the transition distribution parameters. In particular, we can take advantage of count

Algorithm 6 Model-fitting Algorithm

```
Initialize state dynamics and measurement model parameters
while dynamics and measurement model parameters have not converged do
    while time-varying bias has not converged do                ▷ Approximate E-step
        Compute distributions using Forward-Backward algorithm
        Update time-varying bias using (5.12)
    Update dynamics and measurement model parameters            ▷ M-step
```

aggregators (CAs), in which case the update expression is,

$$[\Lambda_i]_{x_i^{t-1},z_i^{t-1},x_i^t} = \frac{\sum_{t=2}^{T} p_{Q^{k+1}}(x_i^{t-1}, x_i^t \mid y_i^{2:T}) p_{Q^{k+1}}\left(z_i^{t-1} \mid y_j^{2:T}, j \in \mathcal{N}(i)\right)}{\sum_{t=2}^{T} p_{Q^{k+1}}(x_i^{t-1} \mid y_i^{2:T}) p_{Q^{k+1}}\left(z_i^{t-1} \mid y_j^{2:T}, j \in \mathcal{N}(i)\right)}, \tag{5.18}$$

where $z_i^{t-1} \in [0, 1, \ldots, |\mathcal{N}(i)|]$ summarizes the influence from the neighbors of MDP i.

Finally, the third term of (5.14) is,

$$\sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \mathbb{E}_{Q^{k+1}}\left[\log p_i(y_i^t \mid x_i^t)\right] = \sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \sum_{x_i^t} p_{Q^{k+1}}(x_i^t \mid y_i^{2:T}) \log p_i(y_i^t \mid x_i^t), \tag{5.19}$$

and the resulting update for the parameters of $p_i(y_i^t \mid x_i^t)$ depends on the choice of observation model. We derive the update expression for two different examples in Section 5.5 after we introduce two case studies demonstrating the use of our approach. Algorithm 6 summarizes our method and we provide details on two data sets in the next section, as well as performance metrics to evaluate our approach in comparison to a baseline method.

5.5 Datasets and Results

5.5.1 Performance Metrics

To develop a set of performance metrics, we also learn a model based on a complete independence assumption, where each MDP in the GMDP has a transition matrix $\Lambda_i^{\text{nMDP}} \in \mathbb{R}^{|\mathcal{X}_i| \times |\mathcal{X}_i|}$ with no coupling interactions with other MDPs. The same parameterization for the measurement model $p_i(y_i^t \mid x_i^t)$ is used, and we use the standard EM approach to learn the model parameters given the measurements $y_i^{2:T}$ for each MDP. We refer to this model as "nMDP," to emphasize the independence assumption, when presenting metrics to evaluate our approach. We start by comparing the objective value of the learning algorithms for each model assumption after reaching, which is a common metric when learning discrete Markov models. In particular, this corresponds to the log-likelihood of the data under the different model assumptions. We emphasize here that computing the log-likelihood of the data, $\log p_\theta(y^{2:T})$, is intractable for GMDPs as it requires enumerating all possible state

trajectories $x^{1:T} \in \prod_{i\in\mathcal{V}}|\mathcal{X}_i|$. Therefore, we use the log-likelihood computed from the approximate distribution ϕ_i^t instead, which provides an under-approximation to the log-likelihood under the true distribution based on the GMDP model.

We also introduce additional metrics based on the Kullback-Leibler (KL) divergence to provide additional analysis and insight,

$$f_i^{\mathrm{KL}} = \sum_{x_{i\cup\mathcal{N}(i)}^{t-1}} \sum_{x_i^t} [\Lambda_i]_{x_{i\cup\mathcal{N}(i)}^{t-1}, x_i^t} \left(\log [\Lambda_i]_{x_{i\cup\mathcal{N}(i)}^{t-1}, x_i^t} - \log g(x_{i\cup\mathcal{N}(i)}^{t-1}, x_i^t) \right), \tag{5.20}$$

where g represents a comparison or baseline distribution. We compute this metric for each MDP and then present the results as a box-and-whisker plot to draw conclusions and gain insight into the learned model. We consider two cases for the distribution g, starting with,

$$g(x_{i\cup\mathcal{N}(i)}^{t-1}, x_i^t) = \left[\Lambda_i^{\mathrm{nMDP}}\right]_{x_i^{t-1}, x_i^t}, \tag{5.21}$$

which uses the transition model from the nMDP model. If the metric f_i^{KL} has little spread in values for this case, then the learned GMDP does not represent significant coupling interactions between MDPs, and the independence assumption is more appropriate for the data. For this case, we refer to the metric as "coupling strength" and denote it f_i^{coupling}. We also consider,

$$g(x_{i\cup\mathcal{N}(i)}^{t-1}, x_i^t) = \frac{1}{\prod_{j\in\mathcal{N}(i)}|\mathcal{X}_j|} \sum_{x_{\mathcal{N}(i)}^{t-1}} [\Lambda_i]_{x_{i\cup\mathcal{N}(i)}^{t-1}, x_i^t}, \tag{5.22}$$

which averages the learned dynamics model over configurations of neighbor states for a given MDP. We use this average distribution as a comparison to determine if the influence of neighboring MDPs is not significant, i.e., the dynamics of an MDP do not change based on the neighbor MDP states. We again use a box-and-whisker plot to present this metric for a learned GMDP, we refer to the metric as "influence strength" and denote it $f_i^{\mathrm{influence}}$.

5.5.2 Novel Coronavirus 2019 (COVID) in California

We use the GMDP model we introduced in Chapter 2.5 to model the 2020 COVID pandemic in California, USA. The data consists of a daily case count for all 58 counties in California, normalized by county population, from March 18$^{\mathrm{th}}$, 2020 to October 28$^{\mathrm{th}}$, 2020 [108]. The initial belief of each county is based on the case count at the first time step. If there are a non-zero number of initial cases, $y_i^2 > 0$, then the prior belief is,

$$p_i(x_i^1) = \begin{cases} 0.4 & \text{if } x_i^1 = \text{Alert Level 2}, \\ 0.2 & \text{otherwise.} \end{cases} \tag{5.23}$$

If there are no initial cases, $y_i^2 = 0$, then the prior belief is,

$$p_i(x_i^1) = \begin{cases} 0.4 & \text{if } x_i^1 = \text{Alert Level 0}, \\ 0.2 & \text{otherwise.} \end{cases} \tag{5.24}$$

We use the following observation model based on the Normal distribution,

$$\begin{aligned} p_i(y_i^t \mid x_i^t = s) &= \text{Normal}(y_i^t; s, \sigma_{is}) \\ &= \sigma_{is}^{-1}(2\pi)^{-1/2} \exp\Big(-\frac{1}{2}(y_i^t - s)^2 \sigma_{is}^{-2} \Big), \end{aligned} \tag{5.25}$$

where the mean of the Normal distribution is the state value, $x_i^t \in \mathcal{X}_i = \{0, 1, 2, 3\}$, and the standard deviation associated with each state value are the unknown parameters. Furthermore, we also scale the mean values by a constant $c = 4000$, to correlate the Alert Level of a county with the expected number of daily cases scaled by population. The third term in (5.14) is then,

$$\begin{aligned} \sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \mathbb{E}_{Q^{k+1}} \left[\log p(y_i^t \mid x_i^t) \right] = \sum_{i \in \mathcal{V}} \sum_{t=2}^{T} \sum_{s \in \mathcal{X}_i} p_{Q^{k+1}}(x_i^t = s \mid y_i^{2:T}) \Big(\log \sigma_{is}^{-1} + \log(2\pi)^{-1/2} \\ - \frac{1}{2}(y_i^t - s)^2 \sigma_{is}^{-2} \Big). \end{aligned} \tag{5.26}$$

Taking the derivative of this expression with respect to σ_{is}^{-1} and setting it equal to zero yields,

$$\sum_{t=2}^{T} p_{Q^{k+1}}(x_i^t = s \mid y_i^{2:T}) \Big(\sigma_{is} - (y_i^t - s)^2 \sigma_{is}^{-1} \Big) = 0, \tag{5.27}$$

and the resulting update is,

$$\sigma_{is}^2 = \frac{\sum_{t=2}^{T} p_{Q^{k+1}}(x_i^t = s \mid y_i^{2:T})(y_i^t - s)^2}{\sum_{t=2}^{T} p_{Q^{k+1}}(x_i^t = s \mid y_i^{2:T})}. \tag{5.28}$$

We present the objective value of the learning algorithm for the GMDP and nMDP models, along with the coupling and influence metrics, in Fig. 5.1. First, the negative log-likelihood plots show that the GMDP model assumption has a significantly lower value at convergence, which corresponds to a higher likelihood of observing the data. Furthermore, the coupling strength f_i^{coupling} and the influence strength $f_i^{\text{influence}}$ are non-zero for a significant fraction of counties in California. The spread of values in both metric shows that our GMDP model is able to learn a combination of coupled and uncoupled models for different counties, in order to best fit the observed data. An uncoupled model may be more descriptive based on the measures taken by a county, such as mask mandates, limiting gathering sizes, and requiring quarantines for incoming travelers, which limits the influence of neighboring counties on the daily case count. At this point, we can investigate the

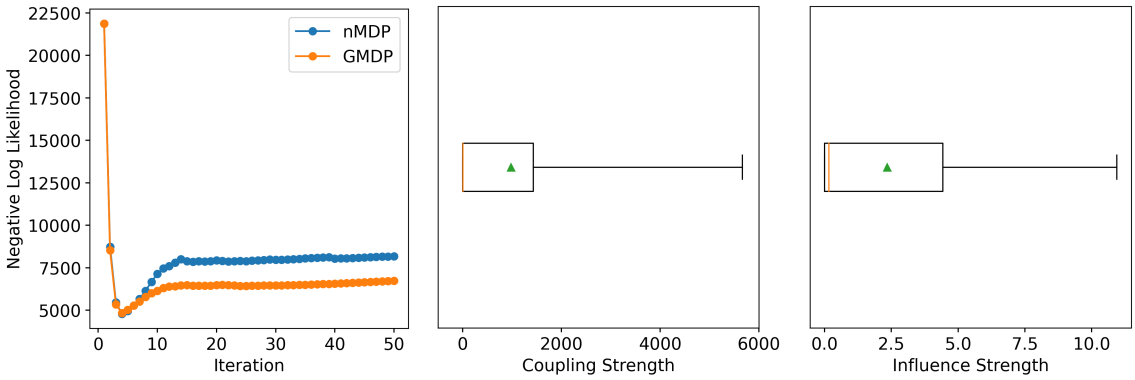

Figure 5.1: (left to right) Comparison of the objective values for the GMDP and nMDP model learning algorithms, and the plots of the f_i^{coupling} and $f_i^{\text{influence}}$ metrics, for the COVID-19 data set. In the box plots, the orange line is the median, the green triangle is the mean, and the caps refer to the minimum and maximum values. For the coupling strength, the minimum is 0.00, the mean is 980.72, and the maximum is 5666.00. For the influence strength, the minimum is 0.00, the mean is 2.35, and the maximum is 10.94. The GMDP model assumption better explains the data by explicitly including coupling interactions, as indicated by all of the metrics.

learned models further to understand what measures are particularly effect in reducing the infection rate.

5.5.3 Tweets on a Topic

We use the GMDP model we introduced in Chapter 2.5 to model user opinions on a single topic based on user activity. We discretize time into eight hour intervals, and for each interval, the data consists of the number of tweets posted and the sentiment of the tweets. We use an off-the-shelf text classifier [109] to generate the sentiment for tweets and we consider an observation model based on the Geometric and Normal distributions,

$$\begin{aligned} p_i(y_i^t \mid x_i^t = s) &= \text{Geometric}(k_i^t; h_i)\text{Normal}(d_i^t; s\mathbf{1}_{k_i^t}, \sigma_{is}\mathbf{I}_{k_i^t}) \\ &= (1-h_i)h_i^{k_i^t}(2\pi)^{-k_i^t/2}|\sigma_{is}^2\mathbf{I}_{k_i^t}|^{-1/2}\exp\Big(-\frac{1}{2}(d_i^t - s\mathbf{1}_{k_i^t})^\intercal(\sigma_{is}^2\mathbf{I}_{k_i^t})^{-1}(d_i^t - s\mathbf{1}_{k_i^t})\Big), \end{aligned} \tag{5.29}$$

where k_i^t is the number of tweets posted, h_i is the user posting rate, and $d_i^t \in \mathbb{R}^{k_i^t}$ represents the tweet sentiments arranged as a vector. For intervals where a user does not post, $k_i^t = 0$, the observation model reduces to $p_i(y_i^t \mid x_i^t = s) = 1 - h_i$. We translate the user states into discrete values in order to use them as the means of the Normal distribution, $x_i^t \in \mathcal{X}_i = \{--, -, \circ, +, ++\} = \{-0.5, -0.25, 0, 0.25, 0.5\}$, as text sentiment is restricted to the interval $[-1, 1]$. In the measurement model, the only unknown parameters are the posting rate and the standard deviation associated

with each user state. The third term in (5.14) works out to,

$$\begin{aligned}\sum_{i\in\mathcal{V}}\sum_{t=2}^{T}\mathbb{E}_{Q^{k+1}}\left[\log p_i(y_i^t \mid x_i^t)\right] &= \sum_{i\in\mathcal{V}}\sum_{t=2}^{T}\sum_{s\in\mathcal{X}_i} p_{Q^{k+1}}(x_i^t = s \mid y_i^{2:T})\Big(\log(1-h_i) + k_i^t \log h_i \\ &\qquad + \log(2\pi)^{-k_i^t/2} + \log|\sigma_{is}^2 \mathbf{I}_{k_i^t}|^{-1/2} \\ &\qquad - \frac{1}{2}(d_i^t - s\mathbf{1}_{k_i^t})^\intercal(\sigma_{is}^2\mathbf{I}_{k_i^t})^{-1}(d_i^t - s\mathbf{1}_{k_i^t})\Big) \\ &= \sum_{i\in\mathcal{V}}\Big(\log(1-h_i)(T-1) + (\log h_i)\sum_{t=2}^{T} k_i^t \\ &\qquad + \sum_{t=2}^{T}\sum_{s\in\mathcal{X}_i} p_{Q^{k+1}}(x_i^t = s \mid y_i^{2:T})\Big(\log(2\pi)^{-k_i^t/2} \\ &\qquad + k_i^t \log\sigma_{is}^{-1} - \frac{1}{2}(d_i^t - s\mathbf{1}_{k_i^t})^\intercal(d_i^t - s\mathbf{1}_{k_i^t})\sigma_{is}^{-2}\Big)\Big).\end{aligned} \tag{5.30}$$

Taking the derivative of (5.30) with respect to the posting rate h_i and setting it equal to zero gives,

$$-\frac{T-1}{1-h_i} + \frac{1}{h_i}\sum_{t=2}^{T} k_i^t = 0, \tag{5.31}$$

which results in the closed-form solution,

$$h_i = \frac{\frac{1}{T-1}\sum_{t=2}^{T} k_i^t}{1 + \frac{1}{T-1}\sum_{t=2}^{T} k_i^t}. \tag{5.32}$$

Taking the derivative of (5.30) with respect to σ_{is}^{-1} and setting it equal to zero gives,

$$\sum_{t=2}^{T} p_{Q^{k+1}}(x_i^t = s \mid y_k^{2:T})\Big(k_i^t\sigma_{is} - (d_i^t - s\mathbf{1}_{k_i^t})^\intercal(d_i^t - s\mathbf{1}_{k_i^t})\sigma_{is}^{-1}\Big) = 0, \tag{5.33}$$

and the resulting update is,

$$\sigma_{is}^2 = \frac{\sum_{t=2}^{T} p_{Q^{k+1}}(x_i^t = s \mid y_i^{2:T})(d_i^t - s\mathbf{1}_{k_i^t})^\intercal(d_i^t - s\mathbf{1}_{k_i^t})}{\sum_{t=2}^{T} p_{Q^{k+1}}(x_i^t = s \mid y_i^{2:T})k_i^t}. \tag{5.34}$$

Our data set consists of tweets on the topic of fake news from February 9$^{\text{th}}$, 2017 to March 18$^{\text{th}}$, 2017 [110]. The initial belief for each user is uniform over user's possible states, and we remove users with less than 50 tweets to ensure each user has enough data to learn an accurate model of their posting behavior. We also remove tweets with a sentiment less than ± 0.05 to eliminate tweets with an undetermined sentiment, which would incorrectly indicate that a user is neutral on a topic. Finally, we limit the number of neighbors for each user to at most five, and then add edges to the

three users with the most tweets. We use this process to consider the effect of users seeing other users post about the topic without explicitly responding or mentioning the top posters. We present the objective value of the learning algorithm for the GMDP and nMDP models, along with the coupling and influence metrics, in Fig. 5.2. For the log-likelihood, both the GMDP and the nMDP models approximately achieve the same value at convergence. However, the objective value can be misleading, as we are using an approximation to the true log-likelihood of the data under the GMDP model assumption, which is intractable to directly compute. Therefore, we have introduced additional metrics to provide further insight and analysis for the learned models. In particular, the coupling and influence metrics show that the coupled interaction model provided by the GMDP better explain the observed data than the independent model assumption for a number of users. On the other hand, some users are simply posting due to the aggregate activity on the topic, and not because of particular interactions with other users.

At this point, it is possible to investigate which users are more responsible for driving posting activity and user interactions on a given topic, compared to others which are simply expressing their opinion. For this data set, we note that it is difficult to determine sentiment from tweets [111], due to the 140 character limit and the frequent use of abbreviations, slang, and emojis. However, there are no easily available off-the-shelf text sentiment classifiers specifically for tweets, which limits the quality of the data given to the learning algorithms, and accurately determining tweet sentiment remains an open research question. We believe that improving the sentiment analysis will directly benefit our framework, and we intend to investigate this aspect further in future work.

5.5.4 Data Considerations

We emphasize that using real-world data to learn models contains a number of challenges, which remain open research questions in literature. Data typically contains various types of bias, such as: skew towards certain demographics, control actions influence the observations and cannot be separated, and high-order interactions effects are difficult to accurately describe in discrete Markov models. Furthermore, models learned from historical data may have limited insight for long-term future predictions, since influence and coupling assumptions that fit prior data may not hold in the future. Nevertheless, our learned models provide insight into which effects or details should be further investigated. In particular, we have proposed a modeling framework where it is straightforward to specify higher-order coupling effects than typically considered in prior work. By building upon the Markov model foundation, which has been thoroughly studied in prior work, we aim to provide additional modeling tools and to also leverage previous validation and verification analysis techniques in future work, as we discuss next in our concluding remarks.

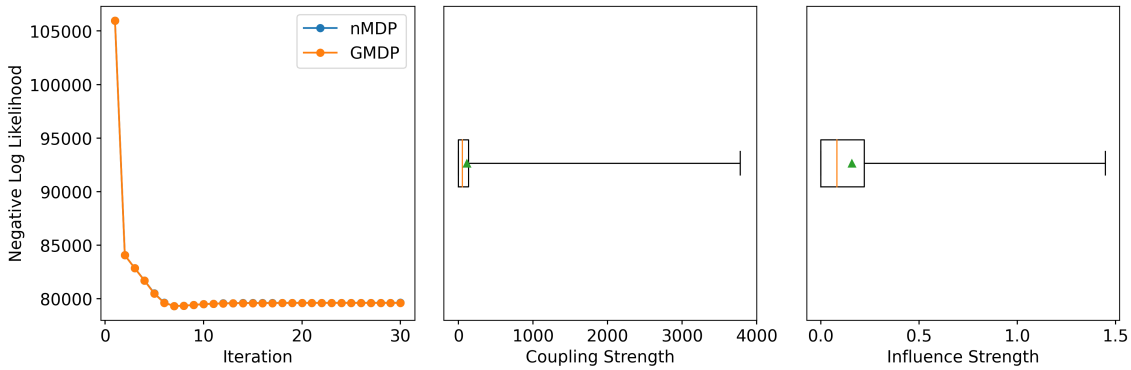

Figure 5.2: (left to right) Comparison of the objective values for the GMDP and nMDP model learning algorithms, and the plots of the f_i^{coupling} and $f_i^{\text{influence}}$ metrics, for the fake news Twitter data set. While both methods achieve the same objective value (the trajectories are overlaid in the plot), the coupling strength and influence strength metrics indicate that the GMDP model better explains the correlations between different users' behavior based on their posting activity. In the box plots, the orange line is the median, the green triangle is the mean, and the caps refer to the minimum and maximum values. For the coupling strength, the minimum is 0.00, the mean is 111.79, and the maximum is 3779.15. For the influence strength, the minimum is 0.00, the mean is 0.16, and the maximum is 1.45. We note that it is difficult to extract sentiment from tweets which limits the insights from the learned models.

5.6 Summary

In this chapter, we developed a model learning algorithm appropriate for large GMDPs with arbitrary coupling structure. We leverage the Expectation-Maximization framework to derive our approach which approximately optimizes the log-likelihood of the observed data. Our approach enables the use of GMDPs with real historical data and in conjunction with our other algorithms, provides a complete set of methods to use GMDPs in a variety of problem settings. In comparison to current methods in literature, we directly consider the case of arbitrary coupling structure. Our experiments with two data sets show that our learned models better explain the observed data compared to a independent model assumption. In addition, we are able to learn non-trivial influence structures which provide insight into real-world processes. In future work, we plan to consider hybrid models, where identity is important for a subset of MDPs in the GMDP, to further improve the representational power of GMDPs. Additional case studies in different application domains would also help validate the use of GMDPs. While prior work has considered relatively small coupled models, this work provides insight into other learning approaches which may have improved theoretical analysis. Given the popularity of Markov models, adapting prior methods for verification and validation is critical for understanding the accuracy and limitations of our learned models.

Chapter 6

Interacting with GMDPs using Teams of Robots

In previous chapters, we presented algorithms for controlling GMDPs with capacity constraints, estimating the state online of a GMDP, and learning the parameters of a GMDP. In this chapter, we develop frameworks which coordinate a cooperative team of autonomous agents that interact with a process modeled by a GMDP in order to consider more practical implementations of our algorithms. We present three frameworks that utilize a cooperative team of autonomous for (i) containing and extinguishing a forest wildfire, (ii) monitoring an aggressive forest wildfire, and (iii) solving a more general statement of team problems. Each of our frameworks is validated through extensive simulation experiments and the performance is compared baseline heuristics and other methods in prior work, when relevant.

We begin by presenting in Section 6.1 a reinforcement learning (RL) framework to generate a decentralized policy for a cooperative team of unmanned aerial vehicles (UAVs) to control a forest wildfire [34]. Each agent has motion constraints, can only view a limited area of the forest, passively communicate with other UAVs, and carries a limited amount of fire retardant. We leverage recent advances in model-free RL to develop a centralized training procedure that produces a network that compactly represents a decentralized control policy for the UAVs. Furthermore, the resulting solution scales independently of forest size and total number of UAVs. We show through extensive simulation experiments that the policy enables effective cooperation and outperforms a baseline heuristic.

Next, we present in Section 6.2 a novel schedule scheme to enable a team of UAVs to perform persistent surveillance of an aggressive wildfire [35]. By aggressive, we mean a wildfire which spreads at a comparable rate to the speed of the UAVs, which means travel time to areas of interest cannot be neglected. Furthermore, we assume that UAVs can only communicate when physically very close

together, as robust long-range communication networks are not viable in disaster response scenarios. Our framework is based on information-theoretic optimization for teams, such as the team orienteering problem, and the result is a scheme which coordinates UAVs meeting to share information and effectively coordinate. Our scheme can be executed in a decentralized manner, and we show that our approach is effect through simulation experiments. In particular, our framework outperforms the case of zero communication and can be comparable to the case of unlimited communication.

Finally, we present in Section 6.3 a general optimization statement that describes cooperative team problems [36]. We use this problem statement to motivate general distributed optimization methods that a team of autonomous agents can use to recover the optimal centralized solution. In particular, we assume there is a communication graph which describes communication links between agents and that a limited amount of information can be shared through communication. We develop solution methods based on the alternating direction method of multipliers framework combined with consensus ideas. We show through simulation experiments on synthetic data and on a persistent surveillance problem that our methods recover the optimal centralized solution and outperform comparable methods in literature.

The material in this chapter appears in publications [34, 35, 36].

6.1 Distributed Deep Reinforcement Learning for Persistent Control

Forest fires are responsible for a significant amount of economic, property, and environmental damage all over the world. Studies have shown that the economic impact alone can reach over a billion US dollars per incident for suppression efforts, environment rehabilitation, and public assistance [112]. Monitoring and controlling forest fires is therefore an appealing application for aerial robotics as terrain, weather, and other factors can pose serious challenges for firefighters – including the potential for loss of life. An attractive solution is to use autonomous multi-agent systems, such as a team of Unmanned Aerial Vehicles (UAVs), in this domain. UAVs are agile, potentially disposable, can be deployed in large numbers, and can form a distributed system, eliminating the need for a central decision maker. Information from a multi-agent system can be provided to firefighters to assist their planning, monitoring, and control efforts.

In this section we first introduce an agent model that includes motion, communication, and action constraints. We discuss task decomposition to address problem complexity and describe a hand-tuned heuristic to generate control inputs for the UAV agents. We then develop a novel extension of Q-learning for multi-agent systems to create a decentralized and scalable solution by which a network of aerial agents can effectively contain the spread of a forest wildfire. Specifically, the solution scales independent of the forest size and the number of agents. Simulation and hardware experiments validate its efficacy.

6.1.1 Related Work

Prior work on the application of robotics to assisting with forest fires has focused on the modeling and monitoring aspects. A number of models have been introduced to describe the spread of a wildfire, including elliptical PDE models [113], vector propagation models using spatial data [114] (used by the US Department of Agriculture Forest Service), and stochastic lattice-based models [66, 115].

Several methods have been published on using autonomous agents to surveil and monitor a forest fire. Methods include using computer vision techniques to detect a fire [116, 117] or to monitor a fire and relevant data of the fire (e.g. temperature and flame height) [118, 119, 120, 121]. Decentralized monitoring systems with UAVs have also been investigated [122]. In general, persistent surveillance is a broad research topic with methods that could be applied to forest fire models.

Surprisingly, there are relatively few published methods for autonomous agents to control a forest fire. One method [123] constructs a fully decentralized method using potential fields and the agents are able to surround and suppress a wildfire. Another method [124] uses a centralized formulation with one agent collecting observations that are sent to a base station which then determines actions for the other agents. In [66], the authors also develop two centralized policies that are idealized and provide conditions necessary for agents to stabilize a fire under their model. Other work has posed forest fire fighting as a centralized resource allocation problem without autonomous agents [125, 126].

Reinforcement learning (RL) approaches also mainly focus on persistent monitoring of forest fires. Ure et al. [127] developed an online decentralized cooperative method for agents to build an estimated model of a forest fire process. Julian et al. [128] proposed two deep reinforcement learning methods for fixed-wing aircraft with imperfect sensors to maximize sensor coverage of a forest fire. The application of RL in prior work for modeling and sensor coverage [127, 128] provided inspiration for using RL methods to produce a decentralized policy for agents to suppress a forest fire. Bertsimas et al. [129] applied Monte Carlo tree search and rolling horizon optimization to forest fire management for a centralized decision maker and assume suppression resources can instantaneously be assigned to any location.

Model-based RL is the appropriate framework for solving the described problem, but exact methods and many approximate methods either cannot address the domain complexity or require recomputing solutions when changing problem parameters. To address domain complexity, we leverage task abstraction which has been shown to be effective for improving long-term planning [130, 131]. For scalability, we propose a novel extension of deep $\mathcal{Q}$-learning.

6.1.2 Agent Model

We use the lattice-based forest wildfire model we introduced in Chapter 2.5, where there are a total of n trees arranged on a lattice. The graph vertex set is $\mathcal{V} = \{1, \ldots, n\}$ and edges for each tree exist

between itself and its lattice neighbors.

The UAV agent model includes motion constraints, communication limits, and simple sensors. We use the index k to refer to the k^{th} agent, as in the wildfire model we use i to refer to the i^{th} tree. There is a total of C agents operating in the domain and the UAVs are considered agile in the sense that u_{limit} actions can be executed before the forest wildfire model is updated.

Due to the disparity of time steps for model updates and time steps for agent actions, the time index τ is used when referring to time-dependent agent quantities.

1. **Motion.** Each agent moves on the same plain square two-dimensional lattice $\mathbb{Z}^2$ as the forest. The agent is also not constrained to stay within any boundaries — it can move infinitely away from the lattice in any direction. The position of an agent k is represented by $p_k^\tau \in \mathbb{Z}^2$. The agent action space contains nine possible actions which modify the position, $p_k^{\tau+1} = p_k^\tau + u_k^\tau$ with $u_k^\tau \in \mathcal{U}$ and,

$$\mathcal{U} = \left\{ \begin{bmatrix} i \\ j \end{bmatrix} \,\middle|\, (i,j) \in \{-1,0,1\} \times \{-1,0,1\} \right\}.$$

   Note that action set $\mathcal{A}$ refers to actions that influence the transitions of the forest and action set $\mathcal{U}$ refers to actions that modify the position of agents.

2. **Sensors.** Each agent is equipped with two sensors.

   i. *Camera.* A downward facing Infrared (IR) camera captures the states of an $h \times w$ sized grid of trees. The agent is located at the center of the image and if images are taken at an edge of the forest, the image is padded with healthy trees.

   ii. *Radio.* The agents are notified of the initial average position q_{fire} of the trees on fire over the entire forest, $q_{\text{fire}} = \left(\sum_{i=1}^n \mathbb{I}(x_i^0 = F) q_i\right) / \sum_{i=1}^n \mathbb{I}(x_i^0 = F)$.

   Example sensor data for an agent is shown in Figure 6.1.

3. **Communication.** Agents communicate with the nearest agent. For example, agent k communicates with

$$j_{\text{comm}} = \underset{j=1,\ldots,C \wedge j \neq k}{\arg\min} \|p_k^\tau - p_j^\tau\|_2.$$

   Agent j_{comm} transmits the position $p_{j_{\text{comm}}}^\tau$ and the label j_{comm} to agent k; no other information is shared. Agent k does not transmit information to agent j_{comm} unless agent k is also the nearest agent. The nearest agent may change at each time step τ as the agents move on the lattice. Figure 6.1 provides an example of the communication model for three agents.

4. **Control action.** Fire retardant is applied to a tree when any agent's position coincides with

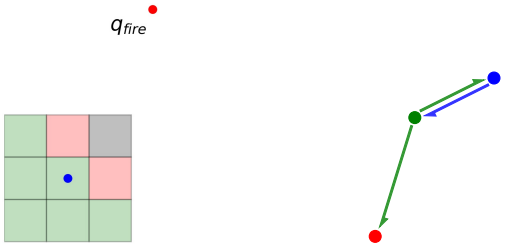

Figure 6.1: (left) Example sensor data for an agent (blue circle): an image (here, $h = w = 3$) of tree states and the initial fire location q_{fire} (red circle). In the image, color indicates tree state: green is healthy, red is on fire, black is burnt. (right) Example communication based on distance for three agents (red, green, and blue circles). Arrow directions show the flow of information and line color indicates the broadcasting agent.

a tree on fire,

$$a_i^t = \begin{cases} 1 & \text{if } x_i^t = F \wedge p_k^\tau = q_i \wedge \tau \in [t, t + u_{\text{limit}}], \\ 0 & \text{otherwise.} \end{cases}$$

The effect of retardant does not compound if multiple agents move to the same tree within the interval $[t, t + u_{\text{limit}}]$ and therefore agents will need to cooperate to avoid wasting control effort. A retardant capacity can be specified, $d_k \in \mathbb{Z}_{>0}$, to limit the number of trees on fire an agent can dump retardant on. If an agent runs out of retardant, the agent returns to a base station near the edge of the forest, its retardant capacity is refilled, and it is re-deployed into the forest at the station.

We assume that agents without retardant have a greater maximum speed and that refilling takes a negligible amount of time. Therefore, a depleted agent is refilled and re-deployed at the station at the next time step τ.

5. **Memory.** Agents track if their position has coincided with a tree that is on fire or burnt, $c_k \in \{\text{True}, \text{False}\}$, which is determined by the image data. Initially, c_k is False and the value is changed only once,

$$c_k = \text{True if } x_i^t \in \{F, B\} \wedge p_k^\tau = q_i \wedge \tau \in [t, t + u_{\text{limit}}]\,,$$

for any time t during the forest fire model simulation.

The agent modeling assumptions can reasonably be implemented on a sub-class of UAVs. Mellinger et al. [132] describe a 3D trajectory and altitude controller for quadrotor vehicles. The quadrotors can be commanded to maintain a constant altitude and move laterally to achieve the discrete motion described in the agent motion model.

Cameras and High Frequency (HF) radios are compact and relatively inexpensive sensors and many quadrotor platforms incorporate both for various applications. We assume the agents are

distributed throughout a forest (e.g. uniformly or randomly) for a task, such as persistent monitoring, prior to the fire ignition. The fire ignition location is assumed to be known and is sent to the agents through HF broadcasting.

Dense communication networks (e.g. all-to-all communication) with high bandwidth are possible but are difficult and costly to implement. Instead, a passive method is described where agents can continually broadcast their position and index label and receive the same information via HF radios. The communication model is relatively low bandwidth and does not require reciprocal data transfer or acknowledgment of transmission or reception, resulting in less susceptibility to errors. Lastly, we assume a practical amount of fire retardant can be stored on-board the agents; enough to dump retardant on 10 trees before returning to the base station for a refill. Although the forest fire and agent models are highly abstracted, we believe they retain the key attributes for designing a fire fighting policy for a team of UAVs.

6.1.3 Heuristic Approach

We now describe a hand-tuned method to generate actions for agents that does not depend on the forest size or the number of agents.

Task Decomposition. In Chapter 2.5.4 we discussed the difficulty in finding solutions to the GMDP model with an agent model, which includes agent motion, sensing, and communication constraints. Therefore, we decompose the problem at the agent level. Each agent is tasked with completing two objectives, described as

1. approach the initial forest fire location,

2. move to suppress the forest fire.

The first task is equivalent to finding the shortest path between an agent's position p_k^τ and the location q_{fire}. Agent communication and cooperation are ignored to further simplify the task. The solution is described in closed-form as,

$$u_k^\tau = \arg\min_{u \in \mathcal{U}} \|p_k^\tau + u - q_{\text{fire}}\|_2. \tag{6.1}$$

Once an agent moves "close enough" to the initial fire location, the agent switches to the second task. The memory c_k determines when to switch tasks as c_k describes when an agent first encounters a tree that is not healthy, an indication that the agent is near a fire.

Algorithm Description. Algorithm 7 describes the heuristic that generates actions for both tasks. The method uses information derived from an agent's sensors which is encapsulated in a feature set denoted s_k^τ for notational convenience. The agent feature set s_k^τ consists of the following quantities.

1. **Memory c_k^τ.**

2. **Image I_k^τ.** The camera image is vectorized to $I_k^\tau \in \mathbb{R}^{hw}$.

3. **Avoidance e_k^τ.** Agents use right-of-way priority by comparing their index with the nearest agent's index. The value is e_k^τ = True if $k > j_{\text{comm}}$ and False otherwise. Agents with higher indices are responsible for yielding to agents with lower indices.

4. **Rotation vector v_k^τ.** The vector $v_{k,\text{cen}}^\tau = p_k^\tau - q_{\text{fire}}$ informs the agent of its relative position with respect to the initial fire location. The rotation vector is $v_k^\tau = R_\perp v_{k,\text{cen}}^\tau$ where $R_\perp$ is the 2D rotation matrix that rotates a vector through -90 degrees. The vector v_k^τ is constructed so that always taking the action $u_k^\tau = \arg\min_{u \in \mathcal{U}} \|u/\|u\|_2 - v_k^\tau\|_2$ will continuously move the agent clockwise about the location q_{fire}.

5. **Nearest agent vector w_k^τ.** The vector $w_k^\tau = p_k^\tau - p_{j_{\text{comm}}}^\tau$ informs the agent of its relative position with respect to the closest agent. If agent k is responsible for avoiding agent j_{comm} (determined by e_k^τ) then it should not move within a safety radius of one of agent j_{comm}.

The feature set s_k^τ is formed by the concatenation of this information into a vector,

$$s_k^\tau = \begin{bmatrix} c_k & I_k^\tau & v_k^\tau & e_k^\tau & w_k^\tau \end{bmatrix}^\intercal. \tag{6.2}$$

Figure 6.2 provides an example of the direction vectors. Both the heuristic and the deep RL method proposed in Section 6.1.4 use this feature set as the only input for each agent to determine its control action.

Once c_k = True, the heuristic first proposes to simply rotate clockwise about the fire center (line 4). The purpose of this action is to make each agent "patrol" and apply retardant along the set of "boundary" trees of the forest fire.

Definition 7 (Boundary tree)**.** A tree i is considered a "boundary tree" if $x_i^t \in \{F, B\}$ and $\sum_{j \in \mathcal{N}(i)} \mathbb{I}(x_j^t = H) > 0$.

The rotational action may sometimes skip applying retardant to trees on fire since the fire grows stochastically. Therefore, the heuristic also considers two "left" actions defined relative to the proposed action so agents can move appropriately as the fire grows (lines 5 and 7). These actions are defined relative to an action $u_k^\tau = \begin{bmatrix} i & j \end{bmatrix}^\intercal$ as,

$$u_{\text{left},1} = \begin{bmatrix} -j \\ i \end{bmatrix},$$

$$u_{\text{left},2} = \frac{1}{\max\{|i+j|, |i-j|, 1\}} \begin{bmatrix} i-j \\ i+j \end{bmatrix}.$$

On the other hand, the agent may move far away from boundary trees if the fire does not grow as quickly as anticipated. A "right" action is taken when the agent believes the proposed action

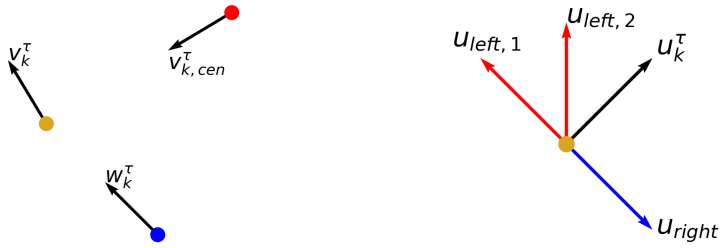

Figure 6.2: (left) Example direction vectors $v^\tau_{k,\text{cen}}$, v^τ_k, and w^τ_k for a given agent (gold circle). The red circle denotes the fire ignition location q_{fire} and the blue circle denotes the nearest agent to the given agent. (right) In the heuristic, the agent considers left and right actions (red and blue vectors) relative to an action u^τ_k.

would move it to a healthy tree that is far from a boundary tree. Specifically, if the new position $p_k^{\tau+1} = p_k^\tau + u_k^\tau$ coincides with a healthy tree x_i^t that has many nearby healthy trees, then the proposed action is switched to a "right" action (line 9), defined relative to u_k^τ as,

$$u_{\text{right}} = \begin{bmatrix} j \\ -i \end{bmatrix}.$$

We define "many nearby healthy trees" as the condition,

$$\mathbb{I}\left(\left|\left\{j \in \mathcal{V} \mid 0 < \|q_j - p_k^{\tau+1}\|_2 \leq 2 \wedge \mathbb{I}(x_j^t = H)\right\}\right| \geq 6\right),$$

which represents the agent moving to a healthy tree which has at least six healthy neighboring trees in its Moore neighborhood. Figure 6.2 shows an example of left and right actions for a proposed action. Lastly, the agent chooses to "stop" if $e_k^\tau =$ True and it will move within the safety radius of the nearest agent (line 11), where the stop action is $u = \begin{bmatrix} 0 & 0 \end{bmatrix}^\intercal$.

We stress that this complex hand-tuned heuristic encapsulates significant expertise from the authors. It is then noteworthy that the deep RL approach we describe next produces a policy that significantly outperforms this heuristic.

A reward function for the second task is described in Algorithm 8 and is used for the deep RL approach. Additional performance metrics are presented in Section 6.1.5 to evaluate both methods. Agents are rewarded for applying control to boundary trees on fire (line 4) as well as moving near the boundary (line 6). Agents are heavily penalized if they fail to avoid collisions (line 9) and rewarded if they move further away from other agents (line 11). Agents are rewarded for moving clockwise about q_{fire} (line 13), which is the condition,

$$\begin{bmatrix} 0 & 0 & -1 \end{bmatrix} \cdot \left(v^\tau_{k,\text{cen}} \times u^\tau_k / \|u^\tau_k\|_2\right) \geq 0.$$

Algorithm 7 Heuristic

Input: agent feature set s_k^τ
Output: agent action u_k^τ
if c_k = True **then**
    $u_k^\tau = \arg\min_{u \in \mathcal{U}} \|u/\|u\|_2 - v_k^\tau\|_2$
    if tree on fire to "left" of agent **then**
        u_k^τ = move to tree on fire
    else if burnt tree to "left" of agent **then**
        u_k^τ = move to burnt tree
    else if agent moves and sees many healthy trees **then**
        u_k^τ = move "right"
    if $e_k^\tau \wedge \|w_k^\tau + u_k^\tau\|_2 \leq 1$ **then**
        u_k^τ = "stop"
else u_k^τ = use Equation (6.1)

Algorithm 8 Agent Reward

Input: agent feature set s_k^τ and action u_k^τ
Output: agent reward r_k^τ
Initialize $r_k^\tau = 0$
if agent moved to a boundary tree on fire **then**
    $r_k^\tau = r_k^\tau + 1$ **else** $r_k^\tau = r_k^\tau - 2$
else if agent moved to a healthy tree **then**
    if tree has at least one on fire or burnt neighbor **then**
        $r_k^\tau = r_k^\tau + 0.5$ **else** $r_k^\tau = r_k^\tau - 1$
if $e_k^\tau \wedge \|w_k^\tau + u_k^\tau\|_2 \leq 1$ **then**
    $r_k^\tau = r_k^\tau - 10$
if $e_k^\tau \wedge \|w_k^\tau\|_2 \leq 1 \wedge \|w_k^\tau + u_k^\tau\|_2 > 1$
    then $r_k^\tau = r_k^\tau + 1$
if agent moved "clockwise" **then** $r_k^\tau = r_k^\tau + 1$

6.1.4 Multi-Agent Deep Q Network

Our deep RL approach generates a decentralized solution that outperforms the heuristic and scales independently of the forest size and the number of agents. A neural network is used to represent the state-action utility function Q following [133]. The network outputs values for all agent actions $u_k^\tau \in \mathcal{U}$ given an agent feature vector s_k^τ and the architecture consists of three fully-connected layers with ReLU activations, represented schematically in Figure 6.3.

The training algorithm for the Multi-Agent Deep Q Network (MADQN) is given in Algorithm 9. Every episode consists of specifying an initial set of trees on fire and agent locations with an example shown in Figure 6.4. Each agent contributes experiences $\left(s_k^\tau, u_k^\tau, r_k^\tau, s_k^{\tau+1}\right)$ to shared memory once c_k = True (centralized training). Experience replay is used to break up correlations between experiences (line 18). After training, agents use the same copy of the final trained network to generate actions online (decentralized execution).

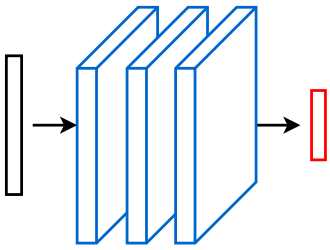

Figure 6.3: MADQN network architecture. The input is an agent's feature set s_k^τ (black rectangle). Hidden layers are fully-connected with ReLU activations (blue rectangles). The output is a vector containing a value for each action u_k^τ (red rectangle).

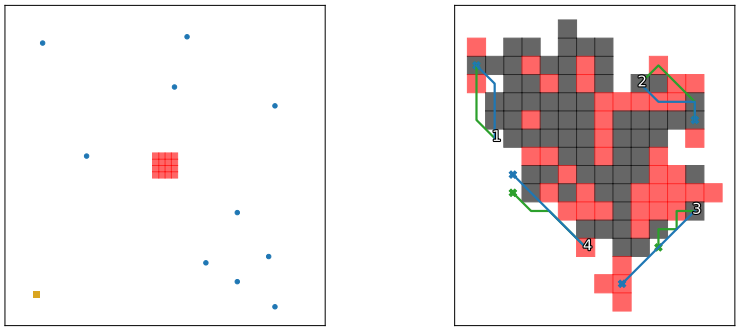

Figure 6.4: Details for MADQN simulation experiments. (both) Red cells are trees on fire and black cells are burnt trees. (left) Sample initial condition with random initial agent positions (blue circles) and a 4×4 grid of trees on fire. Base station is shown as a gold square. (right) Comparison of heuristic (blue) and MADQN (green) paths. Agent number indicates initial position and $\times$'s are final positions.

Target networks [133] are used for stable training of the neural network (line 19). In addition, simulation episodes are capped at a fixed number of model updates, set by t_{limit} (line 5), to bound the maximum cumulative reward.

The training algorithm linearly anneals the exploration rate ϵ from ϵ_{init} to ϵ_{final} over z_{learn} updates of the network. The heuristic (Algorithm 7) is used to populate the replay memory with experiences before network training starts. The heuristic also provides guided exploration of the state space by occasionally providing agent actions (line 10)

6.1.5 Simulation and Hardware Experiments

Following [66], the spatial spreading parameter is $\alpha = 0.7237$ and the fire persistence parameter is $\beta = 0.9048$. The effect of retardant is $\Delta\beta = 0.54$. The MADQN architecture uses three hidden layers, each of which has a size of 2,048. The replay memory was initialized by the heuristic with 5,000 experiences, the memory limit was $D_{\text{limit}} = 1{,}000{,}000$, and the discount factor was $\gamma = 0.95$.

Algorithm 9 Multi-Agent Deep Q Network (MADQN)

Initialize network Q with random θ
Initialize target Q with $\theta^- = \theta$, initialize replay D
for episode $1, \ldots, N$ **do**
    Initialize simulator, agent positions and memory
    while less than t_{limit} simulator updates **do**
        for each agent k **do**
            Generate s_k^τ and update c_k with image data
            if c_k = False **then** u_k^τ = Equation (6.1)
            else sample b uniformly from $[0, 1]$
                if $b \leq \epsilon$ **then** u_k^τ = query heuristic
                else u_k^τ = query network
            Generate reward r_k^τ (Algorithm 8)
        if u_{limit} actions taken **then** update simulator
        if episode ends **then** continue to next episode
        Update agent positions using current actions
        for each agent k **do**
            if c_k = True **then** generate $s_k^{\tau+1}$ and add $\left(s_k^\tau, u_k^\tau, r_k^\tau, s_k^{\tau+1}\right)$ to D
        Sample minibatch from D and update network
        if z_{limit} updates **then** update $\theta^- = \theta$
        Drop experiences if $|D| > D_{\max}$, anneal rate ϵ

The exploration rate ϵ started at one and was linearly annealed to 0.15 over 20,000 updates. For training, 110 episodes were run on a 50×50 sized forest with 10 agents and 16 initial trees on fire. Episodes were terminated after $t_{\text{limit}} = 100$ model updates. The camera image dimensions were $h = w = 3$ and agents performed $u_{\text{limit}} = 6$ actions per model update. Agents were initialized to a random set of tree positions within the forest at the beginning of each simulation. When used, the capacity constraint was $d_k = 10$ for all agents.

The model parameters α and β were chosen to simulate a fire that does not self extinguish but instead will burn down the majority of the forest if control is not applied. In addition, random agent positions do not let the network assume that the fire will always be surrounded by agents.

The evaluation metric for both methods is the fraction of remaining healthy trees to total trees $f_{\text{episode}} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(x_i^T = H)$ at the end of a simulation. Given that the model is stochastic, $N_{\text{episodes}} = 1{,}000$ total simulations were run for each method. The number of agents was varied with two different initial sets of trees on fire: a 4×4 and a 10×10 sized square grid, both at the center of the forest.

Three numerical experiments were used to evaluate the control methods. The first assumes unlimited control capacity to determine if it is possible to contain an aggressive forest fire. The second adds the capacity constraint for more realistic results. The third increases the model parameters to $\alpha = 0.3$ and $\beta = 0.92$ to demonstrate robustness of MADQN to model variations. Larger model parameter values results in a fire that spreads faster and lasts longer.

Table 6.1: MADQN simulation results for three experiments. Data are presented as "loss / limited / win" percentages for 1,000 episodes. The forest is a square grid of 50×50 trees and fires are initialized in a square grid at the center. The shaded cell indicates the configuration used for network training. When overwhelmed with trees on fire (e.g. 10 agents, 10×10 fires) both methods fail to consistently suppress the fire. Otherwise, MADQN scales better with more agents and preserves more healthy trees compared to the heuristic.

Agents	1. Unlimited Capacity			
	4×4 Fires		10×10 Fires	
	Heuristic	MADQN	Heuristic	MADQN
10	61.9 / 0.2 / 37.9	45.0 / 0.0 / 55.0	97.1 / 0.2 / 2.7	95.3 / 0.2 / 4.5
50	24.3 / 22.0 / 53.7	12.6 / 4.3 / 83.1	40.9 / 32.0 / 27.1	22.3 / 10.5 / 67.2
100	17.5 / 23.6 / 58.9	7.5 / 4.9 / 87.6	30.9 / 38.9 / 30.2	15.8 / 11.3 / 72.9

Agents	2. Limited Capacity		3. Limited Capacity, Larger α and β	
	4×4 Fires		4×4 Fires	
	Heuristic	MADQN	Heuristic	MADQN
10	83.8 / 0.0 / 16.2	78.5 / 0.3 / 21.2	94.0 / 0.0 / 6.0	93.1 / 0.0 / 6.9
50	48.0 / 3.4 / 48.6	12.0 / 5.3 / 82.7	77.5 / 0.0 / 22.5	35.5 / 5.6 / 58.9
100	41.0 / 4.3 / 54.7	9.5 / 4.7 / 85.8	70.4 / 1.1 / 28.5	26.9 / 8.6 / 64.5

Figure 6.5 shows sample distributions of the metric f_{episode} using the heuristic and MADQN. For a batch of episodes, results are divided into three categories,

1. "losses": $f_{\text{episode}} \leq 0.2$,
2. "limited": $0.2 < f_{\text{episode}} \leq 0.8$,
3. "wins": $f_{\text{episode}} > 0.8$,

in order to separate the modes of the distribution and summarize the performance. A loss corresponds to almost all trees burning down, limited refers to a limited number of trees burning down, and a win is when a large majority of healthy trees remain at the end of a simulation.

Table 6.1 compares performance of the heuristic and MADQN. For 16 initial fires, the maximum achievable f_{episode} is 99.36% (all trees on fire immediately transition to burnt) and for 100 initial fires the maximum is 96%. The heuristic does well, but does not scale as well as MADQN. With more initial fires and low agents, both methods are less effective, but MADQN rapidly improves with additional agents. Most notably, MADQN was only trained on the case of 10 agents with 16 initial fires and unlimited control capacity. Nevertheless, it adapts to more agents and a different initial configuration, as well as the capacity constraint and model variations. Figure 6.4 shows a comparison of the actions that the heuristic would make compared to the network. The heuristic typically more strictly follows the action to rotate about the fire center q_{fire}.

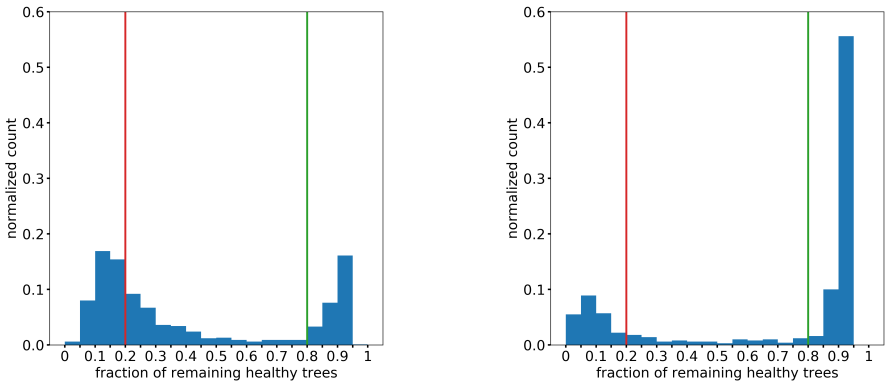

Figure 6.5: Sample distributions of f_{episode} using (left) heuristic and (right) MADQN for 100 initial fires, 50 agents, and unlimited control capacity. Values used for delineating the three performance categories are shown by red and green lines.

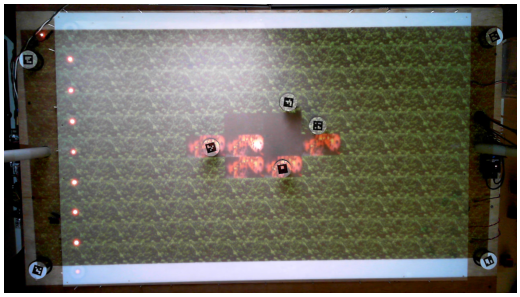

Figure 6.6: Still frame taken from hardware experiments using MADQN which were conducted in the robotarium. An overhead projector displays images to represent healthy, on fire, and burnt trees. Mobile robots represent firefighting UAVs.

The Robotarium [134] at the Georgia Institute of Technology was used to demonstrate actions made by MADQN on mobile robots, as shown in Figure 6.6. The experiment used four agents performing $u_{\text{limit}} = 2$ actions per model update in a 10×10 forest. We show that the agents are able to move to successfully surround and extinguish the wildfire.

6.2 Spatial Scheduling of Informative Meetings for Persistent Coverage

We now consider the task of using a cooperative team of autonomous aerial vehicles (UAVs) to continuously surveil an aggressive forest wildfire. We call a wildfire "aggressive" when its rate of spread is comparable to the movement speed of the UAVs. In such cases, the agents cannot rely on neglecting the travel time to points of interest in the environment. There is significant

interest in using cooperative multi-agent teams to mitigate natural disasters. In California, 10 of the 20 most destructive wildfires in state history occurred within the last four years, including the current most destructive wildfire in November 2018 [135]. Furthermore, the number of wildfires and their intensity will increase in the United States [136]. For natural disaster response, multi-agent systems also cannot rely on communication networks that operate over unlimited range [137]. Issues such as high transmitter power requirements, limited on-board power, and weather conditions typically preclude robust, long-range communication networks. Therefore, we assume that agents communicate only when they are physically close. We model this constraint as requiring agents to be at the same location in the forest in order to exchange information. Furthermore, since wildfires naturally occur over large areas, agents must communicate to maintain an accurate belief of the wildfire as traversing the entire domain individually is ineffective.

In this section, we use a realistic sensing model and build a scalable coordination algorithm to enable the use of our control framework with state uncertainty. Many approaches have been proposed in literature for the persistent coverage problem and some directly consider communication constraints. However, the majority of these methods still allow for communication over arbitrary distances. In addition, prior work also does not scale to processes modeled by large discrete Markov models. Therefore, we propose a novel decentralized coordination algorithm that is scalable to large process models and large multi-agent teams. Agents coordinate by scheduling a future time and location in the forest to meet to ensure information is continuously shared. By using this approach, agents avoid duplicating efforts when they are unable to communicate. Our framework approximately solves the problem of multi-agent exploration with limited communication, and we show through simulations that our approach is effective.

6.2.1 Related Work

Many methods have been proposed for persistent coverage and we review the approaches most relevant to our work here. Coverage frameworks have been developed to account for communication limitations [138, 139], but are typically applied to static or slowly changing environmental processes whereas we aim to monitor a fast process. In addition, many methods require a connected communication graph in order to prove stability properties. In contrast, we consider a problem setup where communication connectivity cannot be guaranteed.

Information-based metrics have been used to create multi-agent exploration frameworks, such as sequential allocation [140, 141], sampling [142, 143, 144], and mixed-integer optimization [145]. Similar to coverage control, many of these methods assume a connected or fully-connected communication graph. While some methods include communication limitations, such as link failures [143] or lossy channels [144], these frameworks still assume communication can occur over arbitrarily large distances.

There is a significant amount of literature on surveillance specifically for wildfires, some of which

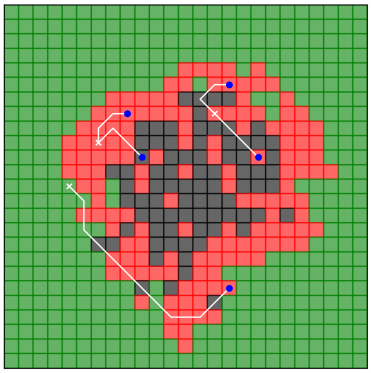

Figure 6.7: Illustration of persistent coverage of a forest wildfire using autonomous aerial vehicles. The forest lattice is visualized as a grid of cells. Multiple agents (blue circles) are tasked with monitoring an aggressive wildfire (red are fires, black are burnt, and green are healthy trees). Agents schedule meetings (white ×'s) to periodically communicate and coordinate their efforts. Communication only occurs in meetings.

propose centralized frameworks [146, 117, 147]. Decentralized methods are based on wildfire boundaries or perimeters [148, 149], image-based feature processing [150, 151], potential field controllers [152, 153], and team coordination or assignment [154, 155]. Almost all of these methods assume a large or unlimited communication radius, e.g. Ure et al. [151] only consider bandwidth limitations, whereas we enforce limited range communication.

Our framework can be viewed as a synchronization (or rendezvous) strategy [156, 157, 158], but we do not assume periodic synchronization of all agents (which may occur over large distances). In addition, some methods [158] require precise boundary information that cannot be known in advance, but must be estimated as the wildfire spreads, and is non-trivial for multiple wildfire sources. In contrast, our approach handles multiple sources which may merge or split over time.

Prior work also does not address large discrete space and discrete time models, as we do in this work. Many methods use discrete process filters or enumerate (or sample) paths through the domain, and do not scale to large state spaces. Likewise, the multi-agent exploration problem is naturally described by partially observable Markov decision processes (POMDPs) [159], but exact methods and many approximate methods are intractable for our problem. In the next section, we introduce the wildfire model along with an agent model.

6.2.2 Agent Model

We use the lattice-based forest wildfire model we introduced in Chapter 2.5, where there are a total of n trees arranged on a lattice. The graph vertex set is $\mathcal{V} = \{1, \ldots, n\}$ and edges for tree exist between itself and its lattice neighbors. The UAV agent model includes dynamics, communication constraints, simple sensors, and schedule-related information.

Dynamics. Each agent moves on the same lattice as the forest. The position of an agent k is represented by $z_k^t \in \mathbb{Z}^2$. The agent action space contains nine possible actions which modify the position, $z_k^{t+1} = z_k^t + u_k^t$ with $u_k^t \in \mathcal{U}$ and,

$$\mathcal{U} = \left\{ \begin{bmatrix} i \\ j \end{bmatrix} \middle| \, (i,j) \in \{-1,0,1\} \times \{-1,0,1\} \right\}.$$

Sensors. Each agent has a downward facing camera which produces a noisy estimate of the states of an $h \times w$ sized grid of trees. The agent is located at the center of the image.

Definition 8 (Camera Function $I(q)$)**.** The function $I(q)$ returns a set of size hw containing the trees $i \in \mathcal{V}$ observed by a camera centered at location q.

The accuracy of each tree state in the image is parameterized by $0 \leq p_c \leq 1$, which is the probability that the observation is the ground truth tree state x_i^t. Each tree in the image is observed independently of all other trees. Equation (6.3) summarizes the sensor model,

$$p(y_i^t \mid x_i^t) = \begin{cases} \frac{1}{2}(1-p_c) + \mathbb{I}(y_i^t = x_i^t)\left(\frac{3}{2}p_c - \frac{1}{2}\right) & \text{if } i \in I(z_k^t), \\ \frac{1}{3} & \text{otherwise.} \end{cases} \tag{6.3}$$

In this model, observations of trees in the agent's camera ($i \in I(z_k^t)$) have probability p_c of matching the ground truth ($y_i^t = x_i^t$) and have probability $\frac{1}{2}(1-p_c)$ of not matching ($y_i^t \neq x_i^t$). Observations of trees outside the camera view have uniform uncertainty (probability $\frac{1}{3}$) for each of the three tree states.

Wildfire Belief. Agents maintain their own belief over the state of the wildfire that is updated every time step using their observations with an approximate Bayesian filter. Agents merge beliefs when in a meeting to improve coordination and wildfire tracking. Details on the belief update and merge are provided in Section 6.2.4.

Communication. Agents only exchange information at prearranged meetings and information is shared by using a Bayesian information fusion strategy so that all agents in the meeting have the same belief. Agents then use their belief to schedule their next meeting and compute nominal paths to the next meeting.

Schedule Information. Agents maintain information related to the meetings they participate in, described by schedules $\mathcal{S}$ and $\mathcal{S}'$. Details on these quantities are provided in Section 6.2.5.

1. Time budget, d_k. The amount of time remaining until the next meeting F_k will occur.
2. Next meeting location, F_k. A lattice location that the agent must be at in d_k time steps to satisfy the schedule.
3. Last meeting location, L_k. A lattice location that represents the meeting after the meeting at location F_k.

4. Stored paths, $\mathcal{R}_k$. A set representing the nominal paths of other agents, which is used to improve the individual path planning of an agent between meetings, when no communication occurs.

The agent modeling assumptions can be reasonably implemented for a sub-class of UAVs. For the discrete motion model, a 3D trajectory and altitude controller [132] can be used to have quadrotors maintain a constant altitude and move laterally. Agents also use a positioning system (e.g., GPS) to maintain a common coordinate frame.

6.2.3 Decentralized Information Gathering Framework

The main idea behind our algorithm is to have pairs of agents schedule their next meeting during their current meeting, subject to all previously scheduled meetings. The agents achieve this by jointly solving a path optimization problem in which they are constrained to end their paths after a prescribed time at the same location (i.e. the next meeting). The last meeting for each agent appear as constraints in this optimization. The objective is to jointly maximize an information gathering metric subject to these constraints. This algorithm guarantees that the future meetings are always kept, even though they are planned in a pairwise distributed and asynchronous fashion. In other words, pairs of agents plan meetings for themselves without knowledge of what all other agents are doing. Between scheduled meetings, an agent can change its path based on new sensed information, as long as the first and last meeting constraints are still satisfied. Our framework results in a distributed, reactive information gathering system.

We only consider pairwise agent meetings in this work, with the aim of maximizing the team's tracking ability of the wildfire. Agents must move to a common location to share beliefs, which leads to many observations of the same area. At the same time, the wildfire is spreading in all directions and therefore the agents sacrifice tracking the wildfire in order to improve their collective belief. In the case of aggressive wildfires, this trade-off may significantly impact the team's performance. We plan to investigate different meeting sizes, along with dynamic scheduling ideas, in future work.

Algorithm 10 provides an overview of the decentralized framework. When agents are meeting, they fuse their individual beliefs, schedule a future meeting, and perform joint path planning to generate nominal paths. Note that these steps can be performed individually (decentralized) or by having one agent perform all steps (centralized) and sharing the results, since each agent has the exact same set of information after fusing beliefs. Outside of meetings, agents do not communicate and individually re-plan their nominal paths based on their own belief of the wildfire. Each agent also updates their own belief at each time step using observations from their camera.

6.2.4 Process Filter and Merging Beliefs

We now describe a scalable approximate Bayesian filter that each agent uses to produce a belief over the state of the wildfire at each time step. We note that the tree dynamics need to be modified

Algorithm 10 Decentralized Information Gathering Framework

```
1: Schedule initial meetings (Algorithm 11)
2: Deploy agents
3: for each time step t do
4:     if a meeting s ∈ S ∪ S' occurs then
5:         Merge agents' k ∈ s beliefs using Eq. (6.5)
6:         Schedule next meeting for s (Algorithm 12)
7:         Perform team path planning (Algorithm 14)
8:     for each agent k do
9:         Plan path to next meeting F_k (Algorithm 15)
10:        Move to first location of planned path
11:        Reduce time budget, d_k ← d_k − 1
12:        Take image and update individual belief
13:    Update wildfire process every ρ time steps
```

to account for the relative speed of the agents moving and the wildfire spreading. We model this relative speed by updating the wildfire every ρ time steps; between updates, the wildfire does not change (Alg. 10, line 13). The dynamics of a tree are then,

$$p_i(x_i^{t+1} \mid x_i^t, x_{\mathcal{N}(i)}^t) = \begin{cases} \text{Table 2.2} & \text{every } \rho \text{ time steps,} \\ \mathbb{I}(x_i^{t+1} = x_i^t) & \text{otherwise.} \end{cases} \tag{6.4}$$

Let $b(x^t) = p(x^t \mid \{y^1, \dots, y^t\})$ represent the belief over the states of all trees given a history of measurements $\{y^1, \dots, y^t\}$; here, y^t represents a measurement of all trees at time t. Given the dynamics (6.4) and sensor model (6.3), the exact recursive Bayesian filter [31] to update the belief is,

$$b(x^t) \propto \Big(\prod_{i=1}^n p_i(y_i^t \mid x_i^t)\Big) \sum_{x^{t-1}} b(x^{t-1}) \prod_{i=1}^n p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}),$$

which is initialized by the prior $b(x^1)$. However, this filter requires the marginalization of all trees which involves enumerating 3^n values. This is not tractable for large forest sizes and thus we use an approximate filter instead. The belief is a product of individual beliefs for each tree, $b(x^t) \approx \prod_{i=1}^n b(x_i^t)$, where $b(x_i^t) = p(x_i^t \mid \{y_i^1, \dots, y_i^t\})$. The belief update for each tree is,

$$b(x_i^t) \propto p_i(y_i^t \mid x_i^t) \sum_{x_i^{t-1}} \sum_{x_{\mathcal{N}(i)}^{t-1}} p_i(x_i^t \mid x_i^{t-1}, x_{\mathcal{N}(i)}^{t-1}) b(x_i^{t-1}) \prod_{x_{\mathcal{N}(i)}^{t-1}} b(x_j^{t-1}).$$

The cost of updating the forest belief per agent is now $\sum_{i=1}^n 3^{|\mathcal{N}(i)|+1}$ operations, and storing the belief requires $\mathcal{O}(n)$ space. The approximate filter can be interpreted as a first order approximation to the true Bayesian filter [31].

We adopt a straightforward approach for fusing beliefs of multiple agents. Let $b_k(x_i^t)$ represent

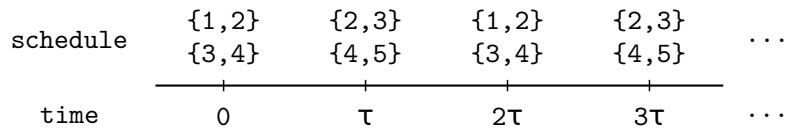

Figure 6.8: For $C = 5$ agents, the schedule is $\mathcal{S} = \{\{1, 2\}, \{3, 4\}\}$ and $\mathcal{S}' = \{\{2, 3\}, \{4, 5\}\}$. The set $\{i, j\}$ indicates agents i and j will meet at the same lattice location and meetings between the same pair of agents re-occur every 2τ time steps.

the belief of agent k about tree i at time t. For agents in a meeting s, a merged belief for a single tree is the average of individual beliefs,

$$\bar{b}(x_i^t) = \frac{1}{|s|} \sum_{k \in s} b_k(x_i^t). \tag{6.5}$$

The merged belief for all trees is then the application of (6.5) for each tree to produce $\{\bar{b}(x_i^t)\ \forall i \in \mathcal{V}\}$. Each agent in the meeting then replaces its own belief with the merged belief. We use averaging to avoid assigning high confidence to poor beliefs, since agents observe a small fraction of the forest at each time step and must rely on open-loop predictions of un-observed areas for planning. We stress that belief fusion has been studied extensively [160], and is not the focus of our contribution in this framework. Our focus is on the distributed, online scheduling of spatial meetings and informative planning between meetings, and any particular belief fusion algorithm can be used within this scheduling framework. We plan to investigate the use of other belief fusion methods in future work. In the next section, we describe our novel scheduling framework.

6.2.5 Schedule Framework

We now propose a strategy to create a feasible schedule that can be executed in a decentralized manner. A meeting consists of a location on the lattice and a time interval to describe when the meeting should occur.

Meeting Intervals. We specify the timing of meetings with the parameter τ. Every τ time steps, the schedule alternates between two sets of assigned meetings $\mathcal{S}$ and $\mathcal{S}'$,

$$\mathcal{S} = \left\{ \{1, 2\}, \ldots, \{2i - 1, 2i\} \mid i \in \left\{ 1, \ldots, \left\lfloor \frac{C}{2} \right\rfloor \right\} \right\}$$
$$\mathcal{S}' = \left\{ \{2, 3\}, \ldots, \{2i, 2i + 1\} \mid i \in \left\{ 1, \ldots, \left\lfloor \frac{C - 1}{2} \right\rfloor \right\} \right\}$$

where set $\{i, j\}$ indicates agents i and j will meet and C is the total number of agents. This construction specifies which agents will meet. In addition, agents 1 and C each only have one meeting in $\mathcal{S} \cup \mathcal{S}'$, while all other agents each have two. Fig. 6.8 shows an example schedule for $C = 5$.

Algorithm 11 Schedule Initial Meetings

1: **Input:** schedule $\mathcal{S}'$, meeting interval τ, agent initial positions z_k^1
2: **Output:** next meeting locations $\{L_k \mid k \in \mathcal{S}'\}$
3: Initialize observed locations, $\mathcal{B} \leftarrow \emptyset$
4: **for** each meeting $s \in \mathcal{S}'$ **do**
5: Find reachable meeting locations $\mathcal{M}_1$,

$$\mathcal{M}_1 = \left\{ i \in \{1, \ldots, d_1 d_2\} \mid \tau = \max_{k \in s} \|q_i - z_k^1\|_\infty \right\}.$$

6: Choose location q_m where $m = \arg\max_{i \in \mathcal{M}_1} \lambda_i$ (6.6)
7: Set $L_k = q_m$ and $d_k = \tau \ \forall k \in s$
8: Update observed locations, $\mathcal{B} \leftarrow \mathcal{B} \cup I(q_m)$

Meeting Locations. Given the interval τ, agents must schedule a future meeting location which is reachable, accounting for other meetings that occur in between as well. This is achieved by finding the set of forest locations that can be reached in τ time steps. An information metric is computed for all reachable locations and a "high value" location is chosen.

Initial Meetings. Prior to the deployment of agents, Algorithm 11 is used in a centralized manner to assign meeting locations for all meetings in $\mathcal{S}'$. This procedure sets L_k as long as agent k has a meeting in $\mathcal{S}'$ which ensures that schedule $\mathcal{S}'$ will be satisfied in τ time steps. Reachable meeting locations (Alg. 11, line 5) are tree locations that are at most τ distance from any agent in the meeting to ensure that all agents will be able to reach the location. The chosen meeting location (Alg. 11, line 6) is based on the residual information λ_i at each location, which accounts for the previously assigned locations $\mathcal{B}$. Details on this metric are provided in Section 6.2.6.

Next, agents are deployed at locations in the forest according to the schedule $\mathcal{S}$ and individually update their schedule information F_k and L_k. For example, for meeting $\{1, 2\}$, agents 1 and 2 are deployed at the same location in the forest, both agents set F_k equal to their deployment location, and also set $L_k = \emptyset$. Agents with no meeting in $\mathcal{S}$ are deployed by themselves and set $F_k = L_k = \emptyset$. This construction ensures that agents will meet according to schedule $\mathcal{S}$ at the first time step. Furthermore, agents are deployed such that any pair of agents with a meeting $k \in \mathcal{S}'$ are initially within τ distance of each other.

After Algorithm 11, agents that have a meeting in $\mathcal{S}'$ and do not have a meeting in $\mathcal{S}$ will have $F_k = \emptyset$. These agents set $F_k \leftarrow L_k$ and $d_k = \tau$ to have a reachable next meeting. Agents that have a meeting in $\mathcal{S}$ and do not have a meeting in $\mathcal{S}'$ will have $L_k = \emptyset$. These agents set $L_k \leftarrow F_k$ and $d_k = 2\tau$ to have the correct time budget for the next meeting.

Subsequent Meetings. For meetings that occur after the initial time step, agents use Algorithm 12 as a team to schedule future meetings after merging their belief. First, a predicted belief is created by predicting forward τ time steps without any measurements; this corresponds to an open-loop prediction of the wildfire. Next, each agent adds their first and last meeting locations, as well as

Algorithm 12 Schedule Next Meeting

1: **Input:** current meeting s, merged belief, meeting interval τ, agent data $\{F_k, L_k \mid k \in s\}$
2: **Output:** new meeting locations $\{F_k, L_k \mid k \in s\}$
3: Predict future belief $\left\{b(x_i^{t+\tau}) \ \forall i \in \mathcal{V}\right\}$
4: Initialize observed locations, $\mathcal{B} \leftarrow \bigcup_{k \in s} I(F_k) \cup I(L_k)$
5: **for** each agent's set of stored paths $\mathcal{R}_k$ **do**
6: **for** each path $\mathcal{P}_j \in \mathcal{R}_k$ **do**
7: $\mathcal{B} \leftarrow \mathcal{B} \cup \{I(q) \mid q \in \mathcal{P}_j\}$
8: Find reachable meeting locations $\mathcal{M}_\tau$,

$$\mathcal{M}_\tau = \left\{i \in \mathcal{V} \mid \tau = \max_{k \in s \setminus \{1,C\}} \|q_i - L_k\|_\infty\right\}.$$

9: **for** each location $q_i, i \in \mathcal{M}_\tau$ **do**
10: **for** each agent $k \in s$ **do**
11: **if** $j \in \{1, C\}$ **then** use $d_k = 2\tau$ **else** use $d_k = \tau$
12: $\mathcal{P}, w_{ik} \leftarrow \texttt{Search}(L_k, q_i, d_k, \{b(x_i^{t+\tau}) \ \forall i \in \mathcal{V}\}, \mathcal{B})$
13: $v_i = \frac{1}{|s|} \sum_{k \in s} w_{ik}$
14: Choose meeting location q_m with m chosen randomly from the set $\{i \mid v_i \geq \gamma \max_{j \in \mathcal{M}_\tau} v_j\}$
15: **for** each agent $k \in s$ **do**
16: **if** $j \in \{1, C\}$ **then** Set $F_k = L_k = q_m$ and $d_k = 2\tau$
17: **else** Set $F_k \leftarrow L_k$ then $L_k \leftarrow q_m$ and $d_k = \tau$

their stored nominal paths, to the set of observed locations $\mathcal{B}$ (Alg. 12, lines 4 and 5), to account for locations that will be observed by other agents not participating in the agents' meeting.

Reachable meeting locations (Alg. 12, line 8) are lattice locations that are τ time steps away from the agents' last meetings L_k to ensure that agents are also able to satisfy their other meetings. Note that agents with only one meeting in $\mathcal{S} \cup \mathcal{S}'$ (i.e., agents $j \in \{1, C\}$) are excluded, as these agents do not have another meeting in τ time steps. A weight for each meeting location (Alg. 12, line 13) is computed by averaging the maximum weight paths each agent would take to the location. A location is randomly chosen from a set of "high" weights (Alg. 12, line 14), where $0 \leq \gamma \leq 1$, to prevent multiple separate meetings from inadvertently choosing the same location. Once a location is chosen, agents update their first and last meeting values and their time budget (Alg. 12, line 15). An example of Algorithm 12 is provided in Fig. 6.9. Next, we provide details on the path planning methods.

6.2.6 Individual and Joint Path Planning

Information Metric. We first describe an information metric for the value of observing different areas of the wildfire. The entropy of the belief of a tree is,

$$H_i(x_i^t) = \sum_{x_i^t} b(x_i^t) \log b(x_i^t),$$

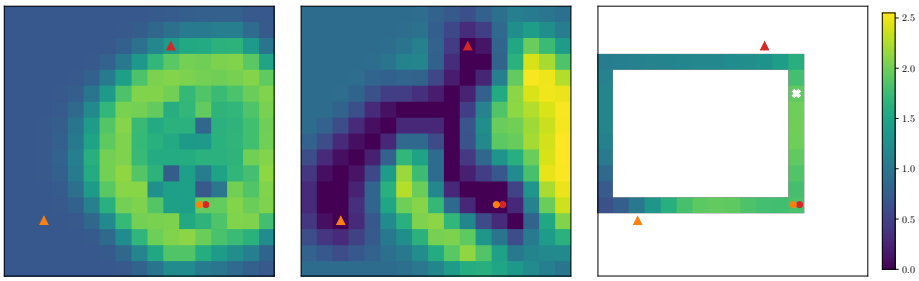

Figure 6.9: Example of scheduling a next meeting. (left) The expected conditional entropy is visualized as a heatmap. The orange and red agents are meeting (circles) and each have another meeting to satisfy (orange and red triangles, respectively). (center) The residual information after each agent adds their stored paths and meeting locations. (right) The reachable meeting locations and their weights, computed by averaging the highest weight paths by each agent. The chosen meeting location is denoted by the white $\times$.

and the expected conditional entropy is,

$$\overline{H}_i(x_i^t \mid y_i^t) = -\sum_{x_i^t} \sum_{y_i^t} p(y_i^t) b(x_i^t) \log b(x_i^t) \geq 0,$$

where $p(y_i^t) = \sum_{x_i^t} p(y_i^t \mid x_i^t) b(x_i^t)$. This quantity describes the expected decrease in entropy in the belief $b(x_i^t)$ after measuring tree i using the camera sensor model (6.3). The mutual information gain for a path $\mathcal{P}_k$ [140] is,

$$\mathcal{T}(\mathcal{P}_k) = \sum_{q_i \in \mathcal{P}_k} \sum_{j \in I(q_i)} H_j(x_j^t) - \overline{H}_j(x_j^t \mid y_j^t).$$

For the multi-agent informative planning problem, we use residual information [140, 141], $\mathcal{T}(\mathcal{B} \cup \mathcal{P}_k) - \mathcal{T}(\mathcal{B})$, where the set $\mathcal{B}$ accounts for other paths taken prior to planning path $\mathcal{P}_k$. Intuitively, the residual information motivates agents to observe different areas of the forest as multiple observations of the same locations have diminishing returns. Each agent needs to compute a path which maximizes the metric, starts at their current position, and passes through their meetings F_k and L_k at the correct times. This problem can be re-framed as finding a maximum path weight given a length constraint, as we discuss next.

Maximizing Path Weight Given Length Constraint. The problem of finding a maximum weight path on a graph given a path length constraint and start and end locations is known as the orienteering problem. As the orienteering problem is NP-hard, prior work has focused on solutions with sub-optimality bounds [161]. Since our framework relies on solving this problem many times to set meeting locations and to plan paths to meetings, we use a fast heuristic. Agents then re-plan their paths between meetings to improve their performance.

Algorithm 13 `Search`

Input: start e, end f, length l, belief $\{b(x_i^t)\ \forall i \in \mathcal{V}\}$, observed locations $\mathcal{B}$
Output: path $\mathcal{P}$, path weight w
Compute weights for lattice locations (6.6)
Build directed acyclic graph with (e, l) as the root vertex
Use Bellman-Ford to find longest path $\mathcal{P}$ from vertex (e, l) to vertex $(f, 0)$ and its associated weight w

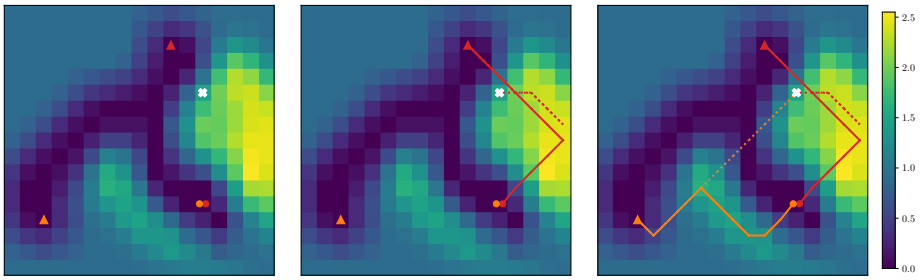

Figure 6.10: Example of joint path planning, based on Fig. 6.9. (left) Residual information heatmap and next meeting location (white ×). (center) The red agent first plans a path from its position to its next meeting (solid line), then from its next meeting to its last meeting (dashed line). (right) Given the red agent's path, the orange agent plans its path. Each agent then stores the other agent's nominal path. Note that the `Search` heuristic produces backtracking paths. Paths are improved by re-planning between meetings.

We now describe our heuristic approach. The value of moving to a tree is based on the residual information,

$$\lambda_i = \mathcal{T}(\mathcal{B} \cup \{q_i\}) - \mathcal{T}(\mathcal{B}). \tag{6.6}$$

We also add a small positive bias ϵ to the expected conditional entropy to account for situations where the conditional entropy is zero for all trees, e.g. when the initial belief of all agents is the ground truth. Given the weights $\{\lambda_i\ \forall i \in \mathcal{V}\}$, a start and end location, and a total path length, we create a directed, acyclic graph (DAG) representation by augmenting each location with a distance. For an agent to plan a path from location e to location h with a maximum length of l, the DAG root vertex is (e, l). The children of this vertex are the locations f that satisfy $\|f - h\|_\infty \leq l - 1$ (i.e. the agent moves closer to h) and $\|e - f\|_\infty \leq 1$ (i.e. the new location satisfies the agent dynamics). The location f is then added to the DAG by adding the vertex $(f, l-1)$ with edges from (e, l) to $(f, l-1)$ with weight λ_f. Likewise, the children of f are the locations g that satisfy $\|g-h\|_\infty \leq l-2$ and $\|f - g\|_\infty \leq 1$. This process continues until there are no more locations to add as decreasing the distance after each new location is added enforces that the agent ends at h after l actions. The Bellman-Ford algorithm is then used to find a maximum weight path (Alg. 13). The `Search` function therefore has $\mathcal{O}(\tau^4)$ time complexity, which is also the dominating complexity in our framework, as this function is used for both scheduling meetings and planning paths.

Algorithm 14 Team Path Planning

1: **Input:** current meeting s, merged belief $\{\bar{b}(x_i^t)\ \forall i \in \mathcal{V}\}$,
meeting interval τ, agent data $\{z_k^t, F_k, L_k, \mathcal{R}_k \mid k \in s\}$
2: **Output:** updated paths $\{\mathcal{P}_k \mid k \in s\}$
3: Initialize observed locations, $\mathcal{B} \leftarrow \emptyset$
4: **for** each agent's set of stored paths $\mathcal{R}_k$ **do**
5: **for** each path $\mathcal{P}_j \in \mathcal{R}_k$ **do**
6: $\mathcal{B} \leftarrow \mathcal{B} \cup \{I(q) \mid q \in \mathcal{P}_j\}$
7: **for** each agent $k \in s$ **do**
8: **if** $j \in \{1, C\}$ **then**
9: $\mathcal{P}_k, w \leftarrow \texttt{Search}(z_k^t, L_k, 2\tau, \{\bar{b}(x_i^t)\ \forall i \in \mathcal{V}\}, \mathcal{B})$
10: **else**
11: $\mathcal{P}_F, w \leftarrow \texttt{Search}(z_k^t, F_k, \tau, \{\bar{b}(x_i^t)\ \forall i \in \mathcal{V}\}, \mathcal{B})$
12: $\mathcal{P}_L, w \leftarrow \texttt{Search}(F_k, L_k, \tau, \{\bar{b}(x_i^t)\ \forall i \in \mathcal{V}\}, \mathcal{B})$
13: $\mathcal{P}_k \leftarrow \mathcal{P}_F \cup \mathcal{P}_L$
14: Update observed locations, $\mathcal{B} \leftarrow \mathcal{B} \cup \{I(q) \mid q \in \mathcal{P}_k\}$
15: **for** each agent $k \in s$ **do**
16: Update paths, $\mathcal{R}_k \leftarrow \mathcal{R}_k \cup \{\mathcal{P}_j \mid j \in s, j \neq k\}$

Individual and Team Path Planning. Agents perform team path planning (Alg. 14) when meeting. The team orienteering problem is challenging and we adopt the multi-agent strategy from Singh et al. [140], which is a sequential allocation planning method. Sequential allocation is an appealing approach as it is a relatively straightforward method to implement for a variety of multi-agent planning problems, and there is significant research on sub-optimality bounds. Furthermore, as our main contribution is a scheduling framework, this method can be changed to any other appropriate planning method. First, one agent plans a path ignoring other agents and updates the set of observed locations $\mathcal{B}$. Each subsequent agent then plans a path, accounting for the previous paths through the set $\mathcal{B}$. To plan paths for two separate meetings, agents first plan from their current position to their next meeting, and add a path from their next meeting to their last meeting (Alg. 14, lines 11 and 12). If agents only have one meeting, a single path is planned with a longer length constraint (Alg. 14, line 9). Agents store the nominal paths computed by the joint path planning to use when there is no communication between meetings (Alg. 14, line 16), which requires $\mathcal{O}(\tau)$ space. Agents remove elements from the stored paths during individual path planning, as detailed next, and therefore $\mathcal{R}_k$ does not grow unbounded. An example of Algorithm 14 is provided in Fig. 6.10.

Agents individually perform receding horizon style planning to account for the wildfire process updating as they move (Alg. 15). Agents use and remove the first location in their stored nominal paths to account for actions of other agents. The weights (6.6) are then calculated from the belief and the set $\mathcal{B}$. In the next section, we present simulation experiments to demonstrate the framework performance.

Algorithm 15 Individual Path Planning

Input: agent belief $\{b(x_i^t)\ \forall i\}$, time budget d_k,
stored paths $\mathcal{R}_k$, position z_k^t
Output: planned agent path $\mathcal{P}_k$
Initialize observed locations, $\mathcal{B} \leftarrow \emptyset$
for each stored path $\mathcal{P}_j \in \mathcal{R}_k$ **do**
    Remove first location q from path, $\mathcal{P}_j \setminus q$
    Update observed locations, $\mathcal{B} \leftarrow \mathcal{B} \cup I(q)$
$\mathcal{P}_k, w \leftarrow \texttt{Search}(z_k^t, F_k, d_k, \{b(x_i^t)\ \forall i \in \mathcal{V}\}, \mathcal{B})$

6.2.7 Simulation Experiments

For the simulations, the forest is a lattice of size 25×25 for a total of $n = 625$ trees and a state space of size 10^{298}. The camera sensing area is $h = w = 3$ and the accuracy is $p_c = 0.95$. While it may seem that the camera is extremely accurate, there is a $0.95^9 = 0.63$ probability that the image returns the true state for all trees in the image. Furthermore, the relatively small field of view also reduces the filter performance due to partial observability. All agents use the ground truth as the initial belief. For scheduling meetings, $\gamma = 0.9$ was used (Alg. 12, line 14).

Performance Metric. Performance for information-based frameworks is typically based on the remaining entropy after running the framework [140, 141]. In addition, the underlying model is typically static (e.g. visual reconstruction) or slowly changing (e.g. on the scale of weeks or months), and does not have an absorbing state. However, the wildfire process quickly reaches an absorbing state which corresponds to no fires in the forest. If the agents never take measurements or move, their belief will converge to 100% accuracy and zero entropy as the initial fire quickly burns down the forest. As a result, poor coordination appears to be successful. Therefore, we introduce a metric to evaluate the transient performance,

$$f_{\text{metric}} = \frac{1}{T}\sum_{t=1}^{T} \frac{\left|\bigcup_{k\in\{1,\dots,C\}} \left\{i \in I(z_k^t) \mid x_i^t = F\right\}\right|}{\left|\left\{i \in \mathcal{V} \mid x_i^t = F\right\}\right|},$$

where T is the total number of simulation time steps. This metric represents the average fraction of ground truth fires covered by the agents over a full simulation. For aggressive wildfires, this metric is typically less than one due to the comparable rate of agents moving and the wildfire spreading. When there are no fires at a given time step, the corresponding term in the summation is zero.

Comparison Methods. We compare with two baseline algorithms. First, there is no communication and at each time step, agents predict their belief 8 steps in the future and choose the highest entropy location that is 8 actions away to move towards. Second, the agents are considered a single team and always communicate; we again stress this is infeasible in practice as it requires long-range communication and only serves as an ideal baseline. The agents have a single belief that is updated

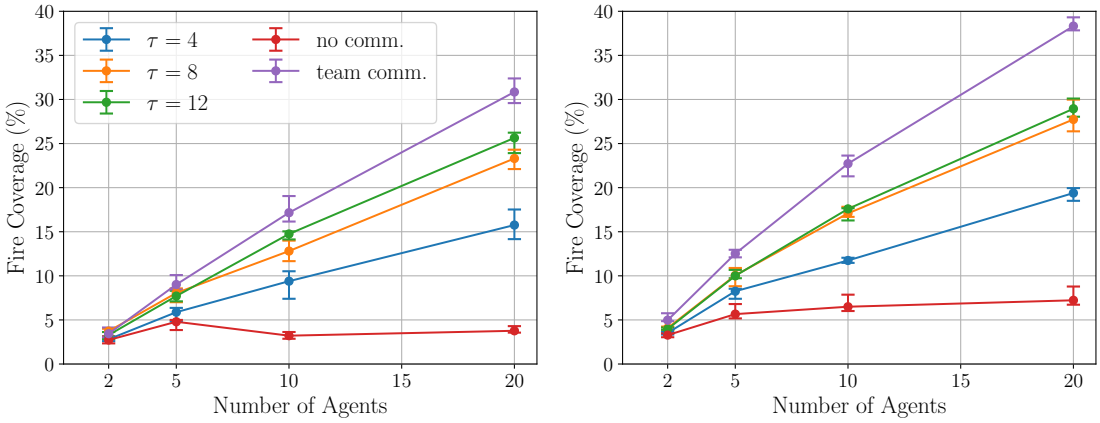

Figure 6.11: Simulation results for different wildfire scenarios, (left) $\rho = 1$ with $T = 60$ and (right) $\rho = 2$ with $T = 120$. The "no communication" baseline is ineffective for all cases, whereas the "team communication" baseline benefits from additional communication. Our framework outperforms the no communication baseline and is comparable to the team communication baseline for many cases. In general, more agents and longer meeting intervals τ improve the framework performance, with diminishing returns as τ increases.

in a centralized manner using all observations. Every 8 time steps, the agents use a sequential allocation strategy, similar to Algorithm 12, to determine a unique F_k and a nominal path for each agent. For other time steps, agents simply execute their planned path. Both baselines are heuristics, as the multi-agent persistent exploration problem with and without unlimited communication are open problems.

Results. Fig. 6.11 shows simulation results for two cases, (left) process update rate $\rho = 1$ with $T = 60$ total time steps and (right) $\rho = 2$ with $T = 120$. For both cases, 10 simulations were run, and the first quartile, median, and third quartile are shown for each configuration of parameters. For all cases, the "no communication" baseline is ineffective. The "team communication" baseline has the best performance, due to the benefit of communication between all agents at every time step. Furthermore, the team communication baseline highlights the benefit of maintaining a single belief using all agent observations without requiring agents to meet to communicate. Overall, our framework is better than the no communication baseline given enough agents, and is comparable to the team communication baseline for many cases. Finally, it is clear that the $\rho = 1$ case is a challenging scenario which mainly requires more agents to be effective.

6.3 Consensus-based ADMM for Task Assignment

In this section, we move away from a specific application domain and instead consider a more general class of optimization-based problems for cooperative autonomous teams. Cooperative team-based

problems in robotics frequently involve solving a constrained optimization problem in a decentralized fashion, as in multi-robot formation control, task assignment, and target tracking. While the field of distributed optimization has recently made significant strides, much of this work has yet to be translated to the field of robotics. Typically, decentralized optimization methods for multi-robot systems are tailored for specific problem aspects, such as robot dynamics, sensor models, and communication architecture. In this work, we describe a more general framework for describing cooperative robotics problems, propose decentralized algorithms, and provide conditions for robots to determine the globally optimal solution. In particular, we discuss our approach in the context of multi-robot task assignment where robots are required to cooperatively complete a set of tasks. We consider two problem statements for the task assignment problem. First, there are equal number of tasks and robots and each task is assigned a unique robot. Second, there are less tasks than robots and each task must be assigned at least one robot. We consider a framework that naturally captures both of these problem statements and allows for a single algorithmic approach for simplicity and flexibility.

We leverage recent advances of the alternating direction method of multipliers (ADMM) framework for our decentralized algorithms. ADMM has been studied extensively and applied to problems mainly in machine learning and signal processing but few works have explored ADMM algorithms in robotics. Notably, the abstractions in distributed optimization do not directly translate to the abstractions typically considered in robotics. For example, agents in distributed optimization are information processors (e.g. CPU and GPU cores or remote servers) whereas agents in robotics are physical vehicles (e.g. mobile robots or drones) with limited processing and communication resources. Therefore, we aim to provide a guide for other practitioners in robotics to take advantage of the benefits of ADMM. We specifically discuss consensus-style variants of ADMM which are appealing for multi-robot systems due to the robustness to robot failure and the relatively limited processing required by any one robot. Ultimately, we provide an iterative algorithm in which each robot performs a local optimization and some update steps, communicates locally with neighbors, and then repeats until convergence. This structure is similar to classic consensus methods in multi-robot systems.

6.3.1 Related Work

The ADMM framework is well established and studied within the optimization community; see [162] for a review. There has been a recent resurgence of ADMM approaches as many applications in statistics and machine learning with large datasets can be formulated as structured convex optimization problems amenable to parallel processing. Notably, the typical ADMM formulation with a separable objective requires a centralized update step: a central node gathers information from "worker" nodes, a global parameter is updated, and this parameter is communicated to the worker

nodes. The worker nodes then perform individual computations and the process repeats. This abstraction is fundamentally at odds with the abstractions in multi-robot applications where no single robot has access to (or gathers) all information. Therefore, we leverage recent work on ADMM [163, 164] to develop decentralized algorithms suitable for robotics.

Variations of ADMM have been developed that do not require a central node [165, 166]. However, these approaches either require stricter conditions or consider a more specific optimization statement. In contrast, we leverage recent work [163, 164] to formulate algorithms for a more general class of problems in robotics.

There are few existing works in robotics that leverage ADMM. Park et al. [167] use the typical ADMM approach for multi-target tracking which requires a central node. Choudhary et al. [168] leverage ADMM as a parallelized optimization solver for distributed map storage and updates in SLAM. Notably, Van Parys and Pipeleers [169] tailor an ADMM approach without a central node to a specific motion planning problem for a team of vehicles and note that their approach has no convergence guarantees.

For task assignment, the Hungarian algorithm [170] is the fastest known centralized algorithm; in contrast, we propose ADMM as a decentralized optimization method for task assignment problems. We specifically consider linear sum assignment problems (LSAPs). Branch and bound methods have been used in the centralized case [171] however there are few decentralized versions. Falsone et al. [172] propose an algorithm to solve a multi-robot mixed-integer linear program (MILP) but require a centralized update step. Testa et al. [173] propose a decentralized cutting plane method but require the objective function to be integer-valued. Lastly, Bürger et al. [174] propose a distributed simplex method for MILPs but do not consider inequality constraints, as we do in this work.

Distributed variants of the Hungarian algorithm have been proposed [175, 176]. Both of these methods recover the centralized performance and we use [176] as a benchmark method. We note that these methods are more complicated to implement and analyze than the methods we develop. Lastly, auction (or market) methods are also popular [177, 178, 179, 180] although some of these methods require shared memory or a centralized coordinator. We also use [179] and [180] as benchmark methods and note that [179] is known to provide sub-optimal assignments.

Our main objective is to develop decentralized ADMM algorithms for task assignment and to demonstrate its competitive performance with other methods. More broadly, we aim to provide an additional tool for solving multi-robot optimization problems.

6.3.2 Multi-robot Task Assignment

We begin by defining the task assignment problem and then discuss a more general optimization statement that represents a range of cooperative multi-robot problems.

The multi-robot team consists of n robots that are required to cooperatively complete m tasks. The vector $x_i \in \{0, 1\}^m$ describes the tasks that robot i has been assigned. The elements of x_i are

binary and $[x_i]_j = 1$ indicates robot i is assigned to task j. Conversely, $[x_i]_j = 0$ means robot i is not assigned to task j. The cost of the assignment of a robot to different tasks is described by the vector $\alpha_i \in \mathbb{R}^m$. We also enforce robots to only be assigned to one task. We consider two variations of an additional constraint,

(Case 1) there are an equal number of tasks and robots ($n = m$) and each task is assigned to only one robot (and vice versa);

(Case 2) there are more robots than tasks ($n > m$) and each task must be assigned to at least one robot.

The communication network for the team is represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. Each vertex corresponds to a robot thus $\mathcal{V} = \{1, \ldots, n\}$ and an edge $e = (i, j)$ exists if robot i and robot j are in communication with each other. The *neighbor set* $\mathcal{N}(i) = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$ describes the set of robots in communication with robot i.

Assumption 1 (Connected Network)**.** *We assume the communication network $\mathcal{G}$ is* connected. *In other words, there exists a path between every pair of vertices $i, j \in \mathcal{V}$ which may consist of more than one edge.*

We note that some prior work assumes a fully-connected graph, in which each robot is connected to every other robot, whereas we assume a less strict condition. The following optimization summarizes the task assignment problem.

Problem 1 (Multi-robot Task Assignment)**.**

$$\underset{x_1,\ldots,x_n \in \mathbb{R}^m}{\text{minimize}} \quad \sum_{i=1}^{n} \alpha_i^\intercal x_i \tag{6.7a}$$

$$\text{subject to} \quad [x_i]_j \in \{0, 1\} \ \forall j \in \{1, \ldots, m\}, i \in \mathcal{V} \tag{6.7b}$$

$$\mathbf{1}_m^\intercal x_i = 1 \ \forall i \in \mathcal{V} \tag{6.7c}$$

$$\underbrace{\sum_{i=1}^{n} x_i = \mathbf{1}_m}_{\text{Case 1}} \ \text{or} \ \underbrace{\sum_{i=1}^{n} x_i \geq \mathbf{1}_m}_{\text{Case 2}} \tag{6.7d}$$

Problem 1 is an integer linear program (ILP). The most appropriate solution methods are those that directly consider integer constraints (e.g., branch and bound or cutting plane methods), but it is difficult to produce decentralized versions of these methods. Therefore, we instead consider a linear programming relaxation where the integer constraint is replaced with a linear constraint. After solving the relaxed program, the resulting solution is rounded to an integer solution. In Problem 1, the constraint (6.7b) is replaced by,

$$[x_i]_j \in \{0, 1\} \rightarrow 0 \leq [x_i]_j \leq 1 \quad \forall j \in \{1, \ldots, m\}, i \in \mathcal{V}. \tag{6.8}$$

For Case 1, solving the relaxed problem results in an integer-valued solution with no need for rounding. To see this, note that for each vector x_i in Problem 1, the basis of the feasible set is the standard basis $\{e_i \in \mathbb{R}^m \mid 1 \leq i \leq m\}$. By the fundamental theorem of linear programming, the optimal solution $x_i^\star$ either lies at a vertex or on a face of the convex polytope defined by the feasible set. The solution only lies on the polytope face if robots have equal cost of assignment for some (or all) of the tasks which results in multiple optimal assignments with equal objective value. Handling multiple optimal solutions is non-trivial [174, 176] and therefore we only consider problems with a unique optimal solution.

For Case 2, there is no general characterization of when the relaxed solution will be integer-valued. We adopt a simple rounding strategy and show through simulations that this strategy is effective (see Section 6.3.6). The goal is to solve Problem 1 in a decentralized fashion. We first discuss a class of problems that include the relaxed formulation of Problem 1.

6.3.3 General Cooperative Multi-robot Problems

We consider the following statement as a generalized optimization for multi-robot cooperative problems,

Problem 2 (Multi-robot Optimization Problem)**.**

$$\begin{aligned} \underset{x_1,\ldots,x_n \in \mathbb{R}^m}{\text{minimize}} \quad & \sum_{i=1}^{n} f_i(x_i) + g_i(x_i) \\ \text{subject to} \quad & \sum_{i=1}^{n} A_i x_i = b \text{ and } \sum_{i=1}^{n} C_i x_i \leq d, \end{aligned}$$

with $A_i \in \mathbb{R}^{l_1 \times m}$, $b \in \mathbb{R}^{l_1}$, $C_i \in \mathbb{R}^{l_2 \times m}$, $d \in \mathbb{R}^{l_2}$, $f_i : \mathbb{R}^m \to \mathbb{R}$, *and* $g_i : \mathbb{R}^m \to \mathbb{R}$.

In the objective function of Problem 2, the function f_i encodes the cost function associated with robot i. The function g_i captures regularization or non-smooth components, such as constraint sets on x_i. Let $M_i \in \mathbb{R}^{n \times m}$ be a matrix with row i containing all ones and the remaining entries are zero. Then, Problem 2 is equivalent to the relaxation of Problem 1 when,

$$\text{(Case 1)} \quad A_i = \begin{bmatrix} M_i \\ I_m \end{bmatrix} \in \mathbb{R}^{(n+m)\times m},\ b = \mathbf{1}_{n+m},\ \text{and } C_i = d = 0; \tag{6.9}$$

$$\text{(Case 2)} \quad A_i = M_i,\ b = \mathbf{1}_m,\ C_i = -I_m, \text{and } d = -\mathbf{1}_m; \tag{6.10}$$

and the functions $f_i(x_i)$ and $g_i(x_i)$ are,

$$f_i(x_i) = \alpha_i^\intercal x_i, \qquad g_i(x_i) = \begin{cases} 0 & \text{if } 0 \leq [x_i]_j \leq 1\ \forall j \in \{1,\ldots,m\}, \\ \infty & \text{otherwise.} \end{cases} \tag{6.11}$$

In the next section, we develop ADMM-based algorithms for Problem 2 and thus the relaxation of Problem 1 as well. We first list the assumptions required for the guarantees of ADMM methods.

Assumption 2. *The functions f_i and g_i are closed, proper, and strictly convex. For Problem 2, the Lagrangian $\mathcal{L}$ has a saddle point, a unique minimum is obtained, and strong duality holds.*

For the task assignment problem, f_i and g_i satisfy Assumption 2. The relaxation of the binary constraint (6.8) allows for a convex formulation of the task assignment problem. Otherwise, this constraint produces a non-convex problem which precludes the standard convergence guarantees of ADMM.

6.3.4 Distributed Primal Problem Approach

We now develop distributed algorithms to solve Problem 2, starting with an approach for the primal problem statement. Problem 2 can equivalently be expressed with the variables $X_i \in \mathbb{R}^{m\times n}$ with one such variable associated with each robot,

$$\begin{aligned} \underset{\substack{X_1,\dots,X_n\in\mathbb{R}^{m\times n}\\ Y_{ij}\in\mathbb{R}^{m\times n}}}{\text{minimize}} \quad & \sum_{i=1}^{n} \phi_i(X_i) + \psi_i(X_i) \\ \text{subject to} \quad & \sum_{j=1}^{n} A_j X_i e_j = b \text{ and } \sum_{j=1}^{n} C_j X_i e_j \leq d,\ \forall i \in \mathcal{V} \\ & X_i = Y_{ij} \text{ and } X_j = Y_{ij}\ \forall j \in \mathcal{N}(i), i \in \mathcal{V}, \end{aligned} \tag{6.12}$$

where $e_j \in \mathbb{R}^n$ is the standard basis vector, $\phi_i(X_i) = f_i(X_i e_i)$, and $\psi_i(X_i) = g_i(X_i e_i)$. The quantities A_j, b, C_j, and d are defined in (6.9) (Case 1) and (6.10) (Case 2) for task assignment. The variables Y_{ij} enforce agreement of solutions between each robot and its neighbors. We have introduced the functions ϕ_i and ψ_i to maintain the same abstraction as Problem 2: the objective is a sum of functions where each function may only be known by a single robot i. For the primal approach applied to task assignment, each robot reasons about the assignments of all robots in the team and uses only its own assignments in its contribution to the objective value, since $x_i = X_i e_i$ if $X_i = [x_1 \cdots x_n]$. In addition, $\psi_i(X_i)$ can encode more problem structure. For example, in the relaxed task assignment problem, all robot assignments must be in the $[0, 1]$ interval. Therefore, each robot i can constrain all entries of its solution X_i to be within this interval, rather than only column i, to improve convergence.

It suffices to solve (6.12) in a decentralized fashion and the result will be optimal for Problem 2, as we discuss next, due to the consensus variables Y_{ij}.

Proposition 4 ([163])**.** *Optimization (6.12) is equivalent to Problem 2 with the same optimal objective value. If $X_i^\star\ \forall i \in \mathcal{V}$ is optimal for (6.12) and $x_i^\star\ \forall i \in \mathcal{V}$ is optimal for Problem 2 then $X_i^\star = [x_1^\star \cdots x_n^\star]\ \forall i \in \mathcal{V}$.*

Proof. By Assumption 1, the communication network is connected and all robots have the same solution $X_i = X\ \forall i \in V$ due to the consensus constraints Y_{ij}. Then, (6.12) simplifies to,

$$\begin{aligned} \underset{X\in\mathbb{R}^{m\times n}}{\text{minimize}} \quad & \sum_{i=1}^{n} \phi_i(X) + \psi_i(X) \\ \text{subject to} \quad & \sum_{j=1}^{n} A_j X e_j = b \text{ and } \sum_{j=1}^{n} C_j X e_j \leq d. \end{aligned}$$

Let $X = [x_1 \cdots x_n]$ with $x_i \in \mathbb{R}^m$ so $Xe_j = x_j$. By definition of ϕ_i and ψ_i, Problem 2 and (6.12) are equivalent statements. We also assume there is a unique optimal solution and thus both statements have the same optimal solution. □

The formulation in (2) allows us to apply the ADMM method. First, we form an *augmented Lagrangian* $\mathcal{L}_a$ for (6.12),

$$\begin{aligned} \mathcal{L}_a = & \sum_{i=1}^{n} \phi_i(X_i) + \psi_i(X_i) + \frac{\rho}{2} \sum_{i=1}^{n} \sum_{j\in\mathcal{N}(i)} \left\|X_i - Y_{ij}\right\|_F^2 + \left\|X_j - Y_{ij}\right\|_F^2 \\ & + \sum_{i=1}^{n} \sum_{j\in\mathcal{N}(i)} \mathbf{1}_m^\intercal \Big(V_{ij} \circ (X_i - Y_{ij}) + W_{ij} \circ (X_j - Y_{ij}) \Big) \mathbf{1}_n. \end{aligned}$$

We use the dual variables $V_{ij}, W_{ij} \in \mathbb{R}^{m\times n}$ for each local agreement constraint but have omitted the other constraints in (6.12) (i.e., they have not been dualized). In addition, $\rho > 0$ is a penalty parameter on violation of the robot agreement constraint. ADMM is an iterative process that traditionally consists of three steps per iteration. First, $\mathcal{L}_a$ is minimized with respect to (w.r.t.) the variables $X_i\ \forall i \in V$ (subject to the remaining constraints in (6.12) that were not dualized) considering the other variables Y_{ij}, V_{ij}, W_{ij} to be fixed. A unique set of minimizers is guaranteed to exist by the strict convexity of $\mathcal{L}_a$ and the assumption that Problem 2 is well-defined. Next, $\mathcal{L}_a$ is minimized w.r.t. Y_{ij} with all other variables held constant. Finally, the multipliers V_{ij}, W_{ij} are updated via gradient scent. This process then repeats for subsequent iterations until convergence.

In the following discussion, we use superscript k to indicate iterations of ADMM. Note that $\mathcal{L}_a$ is separable w.r.t. the robot solutions X_i and the variables Y_{ij}. With this decomposition, the update of each solution X_i is,

$$\begin{aligned} X_i^{k+1} = \underset{X_i\in\mathbb{R}^{m\times n}}{\arg\min} \quad & \phi_i(X_i) + \psi_i(X_i) + \mathbf{1}_m^\intercal \left((V_{ij}^k + W_{ji}^k) \circ X_i \right) \mathbf{1}_n \\ & + \rho \sum_{j\in\mathcal{N}(i)} \left\| X_i - \frac{X_i^k + X_j^k}{2} \right\|_F^2 \\ \text{subject to} \quad & \sum_{j=1}^{n} A_j X_i e_j = b \text{ and } \sum_{i=1}^{n} C_j X_i e_j \leq d. \end{aligned}$$

Algorithm 16 Primal Consensus ADMM

1: **Given:** initial $X_i^0 \in \mathbb{R}^{n\times m}$ and $Q_i^0 = 0$ for each robot i; penalty parameter $\rho > 0$
2: **repeat**
3: **for** each $i \in \mathcal{V}$ **do**
4: Q_i^{k+1} = Equation (6.13) and X_i^{k+1} = Equation (6.14)
5: **until** iteration limit reached or robot solutions converge

The variables Y_{ij} have a simple closed-form update, $Y_{ij}^{k+1} = \frac{1}{2\rho}\left(V_{ij}^k + W_{ij}^k\right) + \frac{1}{2}\left(X_i^{k+1} + X_j^{k+1}\right)$. The dual variable updates are,

$$V_{ij}^{k+1} = V_{ij}^k + \frac{\rho}{2}\left(X_i^{k+1} - Y_{ij}^{k+1}\right), \qquad W_{ij}^{k+1} = W_{ij}^k + \frac{\rho}{2}\left(X_j^{k+1} - Y_{ij}^{k+1}\right).$$

If the dual variables are initialized to zero, then the dual update simplifies [163],

$$Q_i^{k+1} = Q_i^k + \rho \sum_{j\in\mathcal{N}(i)} \left(X_i^k - X_j^k\right), \tag{6.13}$$

where $Q_i^k = \sum_{j\in\mathcal{N}(i)} V_{ij}^k + W_{ji}^k$. Further, the X_i update for the task assignment problem can be written as,

$$\begin{aligned} X_i^{k+1} = \underset{X_i\in\mathbb{R}^{m\times n}}{\arg\min} \quad & \alpha_i^\intercal X_i e_i + \mathbf{1}_m^\intercal \left(Q_i^k \circ X_i\right)\mathbf{1}_n + \rho|\mathcal{N}(i)|\left\|X_i\right\|_F^2 \\ & - \rho\mathbf{1}_m^\intercal\Big(\sum_{j\in\mathcal{N}(i)}\left(X_i^k + X_j^k\right)\circ X_i\Big)\mathbf{1}_n \\ \text{subject to} \quad & \sum_{j=1}^n A_j X_i e_j = b \text{ and } \sum_{j=1}^n C_j X_i e_j \le d \\ & 0 \le X_i e_i \le 1 \end{aligned} \tag{6.14}$$

after substituting (6.11) and simplifying. Equation (6.14) is a constrained quadratic program (QP) which can be solved efficiently with a general-purpose QP solver. Equations (6.13) and (6.14) consist of local updates: each robot stores and updates two variables $Q_i^k, X_i^k \in \mathbb{R}^{m\times n}$ and communicates its solution X_i^k with its neighbors $\mathcal{N}(i)$ at each iteration k. Algorithm 16 summarizes the decentralized algorithm. Each robot is guaranteed to reach the optimal solution.

Proposition 5 ([162, §3.2-3.3], [163])**.** *Given Assumptions 1 and 2, each robot solution converges to the optimal solution: $X_i^k \to X^\star\ \forall i \in \mathcal{V}$ as $k \to \infty$ where $\{x_i^\star \mid i \in \mathcal{V}\}$ is the optimal solution of Problem 2 and $X^\star = [x_1^\star \cdots x_n^\star]$.*

Our problem formulation (6.12) differs from prior work due to the constraints in (6.14). These constraints do not modify the convergence argument since (6.14) is well-defined with a non-empty feasible set (Assumption 2). Therefore, we refer to the arguments in Proposition 2 by Mateos et al. [163] and omit the proof here. Next, we develop an algorithm based on duality.

6.3.5 Distributed Dual Problem Approach

The dual formulation of Problem 2 is,

$$\begin{aligned} \underset{\nu\in\mathbb{R}^{l_1},\lambda\in\mathbb{R}^{l_2}}{\text{minimize}} \quad & \sum_{i=1}^{n}\left(h_i(\nu,\lambda)+\frac{1}{n}\nu^\intercal b+\frac{1}{n}\lambda^\intercal d\right) \\ \text{subject to} \quad & \lambda\geq 0 \end{aligned} \tag{6.15}$$

where $h_i(\nu,\lambda)=\underset{x_i\in\mathbb{R}^m}{\text{maximize}} -f_i(x_i)-g_i(x_i)-\nu^\intercal A_i x_i-\lambda^\intercal C_i x_i$, after simplification of the standard dual problem derivation. An equivalent problem, amenable to decentralization due to a separable structure, is the following statement,

$$\begin{aligned} \underset{\substack{\nu_1,\ldots,\nu_n\in\mathbb{R}^{l_1}\\ \lambda_1,\ldots,\lambda_n\in\mathbb{R}^{l_2}\\ t_{ij}\in\mathbb{R}^{l_1},u_{ij}\in\mathbb{R}^{l_2}}}{\text{minimize}} \quad & \sum_{i=1}^{n}\left(h_i(\nu_i,\lambda_i)+\frac{1}{n}\nu_i^\intercal b+\frac{1}{n}\lambda_i^\intercal d\right) \\ \text{subject to} \quad & \nu_i=t_{ij},\nu_j=t_{ij},\lambda_i=u_{ij},\text{ and }\lambda_j=u_{ij}\ \forall j\in\mathcal{N}(i),i\in\mathcal{V} \\ & \lambda_i\geq 0\ \forall i\in\mathcal{V} \end{aligned} \tag{6.16}$$

Each robot reasons about the variables ν_i and λ_i, from which its decision x_i is recovered using h_i. Similar to Proposition 4, (6.15) and (6.16) are equivalent.

Corollary 1. *Statements (6.15) and (6.16) have the same optimal solution.*

We now follow the same derivation as the primal approach to develop the individual update steps for each robot. An augmented Lagrangian is formed for (6.16) without the constraint $\lambda_i\geq 0$ and this function is again separable. Each robot individually solves the following optimization to update ν_i and λ_i,

$$\begin{aligned} \underset{\nu_i\in\mathbb{R}^{l_1},\lambda_i\in\mathbb{R}^{l_2}}{\text{minimize}} \quad & h_i(\nu_i,\lambda_i)+\frac{1}{n}\nu_i^\intercal b+\frac{1}{n}\lambda_i^\intercal d+\nu_i^\intercal(v_{1,ij}^k+v_{2,ji}^k)+\lambda_i^\intercal(u_{1,ij}^k+u_{2,ji}^k) \\ & +\rho\sum_{j\in\mathcal{N}(i)}\left\|\nu_i-\frac{\nu_i^{k-1}+\nu_j^{k-1}}{2}\right\|_2^2+\left\|\lambda_i-\frac{\lambda_i^{k-1}+\lambda_j^{k-1}}{2}\right\|_2^2 \\ \text{subject to } & \lambda_i\geq 0, \end{aligned} \tag{6.17}$$

where the quantities $v_{1,ij},v_{2,ij}\in\mathbb{R}^{l_1}$ and $w_{1,ij},w_{2,ij}\in\mathbb{R}^{l_1}$ are dual variables for the variables t_{ij} and u_{ij}, respectively. The dual variable gradient ascent yields,

$$\begin{aligned} v_{1,ij}^{k+1}&=v_{1,ij}^k+\frac{\rho}{2}\left(\nu_i^{k-1}-\nu_j^{k-1}\right), & v_{2,ij}^{k+1}&=v_{2,ij}^k+\frac{\rho}{2}\left(\nu_j^{k-1}-\nu_i^{k-1}\right), \\ w_{1,ij}^{k+1}&=w_{1,ij}^k+\frac{\rho}{2}\left(\lambda_i^{k-1}-\lambda_j^{k-1}\right), & w_{2,ij}^{k+1}&=w_{2,ij}^k+\frac{\rho}{2}\left(\lambda_j^{k-1}-\lambda_i^{k-1}\right). \end{aligned}$$

Let $q_i^k=\sum_{j\in\mathcal{N}(i)}v_{1,ij}^k+v_{2,ji}^k$ and $r_i^k=\sum_{j\in\mathcal{N}(i)}w_{1,ij}^k+w_{2,ji}^k$. Then the previous updates are replaced

by,

$$q_i^k = q_i^{k-1} + \rho \sum_{j \in \mathcal{N}(i)} \nu_i^{k-1} - \nu_j^{k-1}, \qquad r_i^k = r_i^{k-1} + \rho \sum_{j \in \mathcal{N}(i)} \lambda_i^{k-1} - \lambda_j^{k-1}, \tag{6.18}$$

if all variables $v_{1,ij}, v_{2,ij}, w_{1,ij}, w_{2,ij}$ are initialized to zero [163]. We again omit the variables t_{ij}, u_{ij} as they are redundant given the other variables ν_i, λ_i, x_i.

We note that solving (6.17) is difficult for task assignment as each robot must also solve the optimization,

$$\begin{aligned} h_i(\nu_i, \lambda_i) = \underset{x_i \in \mathbb{R}^m}{\text{maximize}} \quad & -\alpha_i^\intercal x_i - \nu_i^\intercal A_i x_i - \lambda_i^\intercal C_i x_i \\ \text{subject to} \quad & 0 \leq x_i \leq 1, \end{aligned}$$

using (6.11) and either (6.9) (Case 1) or (6.10) (Case 2). Therefore, we leverage problem structure to produce tractable updates for ν_i, λ_i, and x_i. The objective in (6.17) is concave in x_i given values of ν_i, λ_i and is convex in the augmented variable $\begin{bmatrix} \nu_i & \lambda_i \end{bmatrix}^\intercal$ given a value of x_i. Therefore, the minimax theorem [181, §2.6] can be applied to solve (6.17) by considering the maximization first along with the existence of a saddle point. For clarity in the following discussion, we define,

$$\theta(w, z, s, \epsilon_i) = \frac{1}{\rho} w - \frac{1}{\rho n} z - \frac{1}{\rho} s + \sum_{j \in \mathcal{N}(i)} \epsilon_i + \epsilon_j.$$

Applying the minimax theorem, substituting the definition of h_i, and completing the square for both ν_i and λ_i modifies (6.17) to,

$$\begin{aligned} &\underset{x_i \in \mathbb{R}^m}{\text{maximize}} \ \underset{\nu_i \in \mathbb{R}^{l_1}, \lambda_i \in \mathbb{R}^{l_2}}{\text{minimize}} \ -\alpha_i^\intercal x_i \\ &+ \rho |\mathcal{N}(i)| \left\| \nu_i - \frac{1}{2|\mathcal{N}(i)|} \theta(A_i x_i, b, q_i^k, \nu_i^{k-1}) \right\|_2^2 - \frac{\rho}{4|\mathcal{N}(i)|} \left\| \theta(A_i x_i, b, q_i^k, \nu_i^{k-1}) \right\|_2^2 \\ &+ \rho |\mathcal{N}(i)| \left\| \lambda_i - \frac{1}{2|\mathcal{N}(i)|} \theta(C_i x_i, d, r_i^k, \lambda_i^{k-1}) \right\|_2^2 - \frac{\rho}{4|\mathcal{N}(i)|} \left\| \theta(C_i x_i, d, r_i^k, \lambda_i^{k-1}) \right\|_2^2 \\ &\text{subject to } 0 \leq x_i \leq 1. \end{aligned}$$

The inner minimization is separable in ν_i and λ_i given $x_i = x_i^k$, which leads to,

$$\nu_i^k = \frac{1}{2|\mathcal{N}(i)|} \theta(A_i x_i^k, b, q_i^k, \nu_i^{k-1}), \tag{6.19}$$

$$\lambda_i^k = \underset{\lambda_i \in \mathbb{R}^{l_2}}{\arg\min} \left\| \lambda_i - \frac{1}{2|\mathcal{N}(i)|} \theta(C_i x_i^k, d, r_i^k, \lambda_i^{k-1}) \right\|_2^2 \text{ subject to } \lambda_i \geq 0. \tag{6.20}$$

The λ_i^k update has a simple solution: set $\lambda_i^k = \frac{1}{2|\mathcal{N}(i)|} \theta(C_i x_i^k, d, r_i^k, \lambda_i^{k-1})$ and change any elements

Algorithm 17 Dual Consensus ADMM

Given: initial $x_i^0 \in \mathbb{R}^m$, $\nu_i^0 \in \mathbb{R}^{l_1}$, $\lambda_i^0 \in \mathbb{R}^{l_2}$ and $q_i^0 = r_i^0 = 0$ for each robot i; penalty parameter $\rho > 0$
repeat
  for each $i \in \mathcal{V}$ **do**
    Update q_i^k, r_i^k using Equation (6.18)
    x_i^k = Equation (6.21), ν_i^k = Equation (6.19), and λ_i^k = Equation (6.20)
until iteration limit reached or robot solutions converge

of λ_i^k that are negative to zero. Given $\nu_i = \nu_i^k$ and $\lambda_i = \lambda_i^k$, the outer maximization is then,

$$\begin{aligned} x_i^k = \underset{x_i \in \mathbb{R}^m}{\arg\min} \quad & \alpha_i^\intercal x_i + \frac{\rho}{4|\mathcal{N}(i)|} \left\| \theta(A_i x_i, b, q_i^k, \nu_i^{k-1}) \right\|_2^2 + \frac{1}{2}(\lambda_i^k)^\intercal C_i x_i \\ \text{subject to} \quad & 0 \leq x_i \leq 1, \end{aligned} \tag{6.21}$$

after simplification; this form is also a constrained QP which can be solved efficiently. The solution of (6.17) is found by choosing a variable update order: the primal variable is updated using λ_i^{k-1} and then the dual variables ν_i^k,λ_i^k are updated with x_i^k. Algorithm 17 summarizes the dual consensus method.

Proposition 6 ([164])**.** *Given Assumptions 1 and 2, each robot solution converges to the optimal solution: $\nu_i^k \to \nu^\star, \lambda_i^k \to \lambda^\star \; \forall i \in V$ as $k \to \infty$ where $\nu^\star, \lambda^\star$ is optimal for (6.15). Further, any limit point of $\{x_i^k \mid i \in \mathcal{V}\}$ is optimal for Problem 2.*

Our formulation (6.16) differs from the approach by Chang et al. [164] due to the dual variables λ_i introduced for the inequality constraints in Problem 2. Using an augmented variable $\begin{bmatrix} \nu_i & \lambda_i \end{bmatrix}^\intercal$ yields a comparable formulation to the prior approach [164, §4]. The additional constraint $\lambda_i \geq 0$ does not modify the convergence argument as it is satisfied at each iteration by the update equation (6.20). We refer to Theorem 2 by Chang et al. [164] and omit the details here.

In many applications, it is desirable to eliminate optimization statements like (6.21) in favor of closed-form solutions. We now use proximal gradients [182] to achieve this. A first-order proximal update around x_i^{k-1} of the objective in (6.21) leads to the modified objective,

$$\begin{aligned} \Big[\alpha_i + \frac{1}{2|\mathcal{N}(i)|} A_i^\intercal \theta(A_i x_i^{k-1}, b, q_i^k, \nu_i^{k-1}) + \frac{1}{2} C_i^\intercal \lambda_i^k\Big]^\intercal (x_i - x_i^{k-1}) \\ + \frac{\gamma_i}{2} \left\| x_i - x_i^{k-1} \right\|_2^2, \end{aligned}$$

Table 6.2: Average number of iterations for convergence for the task assignment problem with 20 robots and 20 tasks over 50 trials of randomly generated costs.

	Primal	Dual	Inexact Dual
Average Iterations	28	159	387

where $\gamma_i > 0$ is a penalty parameter. This objective leads to the optimization,

$$x_i^k = \underset{x_i \in \mathbb{R}^m}{\arg\min} \ \frac{\gamma_i}{2} \Big\| x_i - x_i^{k-1} + \frac{1}{\gamma_i}\alpha_i + \frac{1}{2\gamma_i |\mathcal{N}(i)|} A_i^\intercal \theta(A_i x_i^{k-1}, b, q_i^k, \nu_i^{k-1}) + \frac{1}{2\gamma_i} C_i^\intercal \lambda_i^k \Big\|_2^2 + g_i(x_i),$$

since the additional terms are constant w.r.t. x_i. The constraint in (6.21) is now enforced by g_i (6.11) in the objective as this form resembles the proximal operator [182],

$$\text{prox}(z; \gamma_i, g_i) = \underset{y \in \mathbb{R}^m}{\arg\min} \ \frac{\gamma_i}{2} \|y - z\|_2^2 + g_i(y).$$

This projection has a simple solution for (6.11): each element of z is clipped to the interval $[0, 1]$. Thus, the x_i update (6.21) is replaced by the following strategy: set $x_i^k = x_i^-$ where,

$$x_i^- = x_i^{k-1} - \frac{1}{\gamma_i}\alpha_i - \frac{1}{2\gamma_i |\mathcal{N}(i)|} A_i^\intercal \theta(A_i x_i^{k-1}, b, q_i^k, \nu_i^{k-1}) - \frac{1}{2\gamma_i} C_i^\intercal \lambda_i^k.$$

Then, change elements of x_i^k that are less than zero to zero and change elements greater than one to one. Convergence of Algorithm 17 with this modified update is guaranteed if the parameters γ_i are sufficiently large [164]. A similar idea of proximal updates for the primal approach is also possible. In the next section, we present simulations comparing the performance of the ADMM methods.

6.3.6 Simulation Experiments

In the following discussion, we refer to the algorithms we previously developed as ADMM task assignment (ADMM-TA). First, we examine the convergence rate of the ADMM-TA methods when using the primal (Algorithm 16), dual (Algorithm 17), and inexact dual (proximal gradient modification) formulations. For this study, we use these methods to solve the relaxation of Problem 1 with 20 robots and 20 tasks. Table 6.2 shows the average number of iterations required to reach 0.1% of the optimal centralized objective value. Convergence is slowest on the inexact dual ADMM-TA method but benefits from having very simple update equations. The primal ADMM-TA method achieves the fastest convergence rate and therefore we use this method for the remaining simulation studies. We also exploit structure for the primal method by constraining all entries of each robot's solution to be in the interval $[0, 1]$.

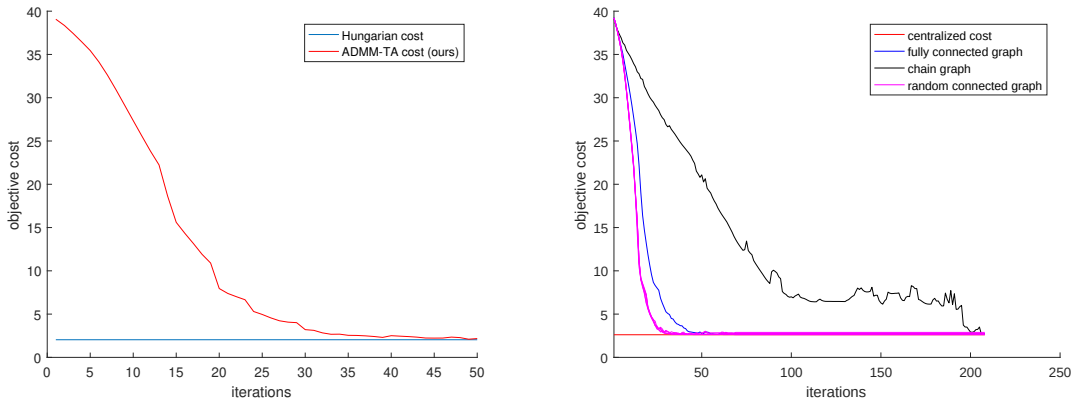

Figure 6.12: (left) The ADMM-TA estimate converges to the Hungarian solution of the task assignment problem within 50 iterations for 50 robots and tasks. (right) Convergence of the ADMM-TA solution to the centralized solution on different communication graphs. The slowest rate of convergence is observed on the linear-chain graph.

Convergence of ADMM to Optimal Solution. Next, we examine the convergence of the primal ADMM-TA method to the optimal centralized solution of the task assignment problem. The Hungarian method is used to solve the task assignment problem without relaxations [170]. We consider 50 robots and 50 tasks using a fully-connected graph where the assignment costs were randomly generated. Figure 6.12 (left) shows the convergence to the optimal cost. During the first few iterations, each robot selects tasks with minimal cost without consensus on the global assignments. The consensus constraints enforce agreement on the assignments for all robots and consequently, each robot adjusts its solution to achieve global consensus. After convergence, the assignments are consistent with the problem constraints.

Performance on Various Graph Topologies. We also compare the performance of our method on different connected communication networks. The worst-case scenario for consensus methods is linear-chain graphs since information propagates slowly between the first and last robot. We compare the convergence rate of linear-chain to randomly generated and fully-connected graphs and also show that all topologies converge to the centralized optimal cost. We consider the case of 50 robots and 30 tasks, which our method easily accommodates, and randomly generated costs. Figure 6.12 (right) compares the convergence rates of the different topologies. The ADMM-TA estimate converges to the optimal solution on all the communication graphs. Notably, the rate of convergence of the ADMM-TA solution to the optimal solution does not depend significantly on the graph topology with the exception of the linear-chain graph. Even in the worst-case graph topology the ADMM-TA method converges in about 200 iterations. On more general graph topologies, convergence of the ADMM-TA method requires roughly 30 iterations. We noticed faster convergence to the optimal solution on randomly-generated connected graphs which is counter-intuitive as typically the fully-connected

graph topology produces the fastest convergence rate. We plan to investigate this behavior in future work.

Comparison with Benchmark Methods. We now compare the scaling of our ADMM-TA approach to other distributed methods for task assignment with respect to number of robots. For each team size, we use an equal number of tasks to as most prior work is designed for this case. We used the consensus-based auction approach (CBAA) [179], market-based consensus [180], and a distributed Hungarian method [176]. The CBAA method does not guarantee optimality after it terminates; in all of the trials we considered, CBAA did not give the optimal solution. In contrast, the market-based consensus and distributed Hungarian methods are optimal. Figure 6.13 (left) shows the number of iterations required by each method to converge (by their own criteria) on a fully-connected graph. For ADMM-TA, we again check if the cost is within 0.1% of the optimal cost to provide a worst-case scaling trend. ADMM-TA scales comparably to the distributed Hungarian method, better than the market-consensus method, and worse than CBAA. We emphasize that CBAA is sub-optimal and also note that the distributed Hungarian method is limited to linear sum assignment problems (LSAPs), whereas our method can be used with more general objective functions.

Next, we compared the performance of our method to the distributed Hungarian method on randomly-generated connected graphs. We did not use the CBAA method since it did not produce optimal estimates as previously discussed. In addition, we did not include the market-consensus method as the algorithm was not amenable to changes for arbitrary communication graphs. Figure 6.13 (right) shows the number of iterations required for each method to produce the optimal task assignments. For small network sizes, ADMM-TA scales better than the distributed Hungarian method. Both methods require about the same number of iterations for much larger network sizes.

Case Study: Persistent Surveillance. We have focused on single (or one-shot) assignments in the preceding studies. Realistic multi-robot systems might require assignments to be made periodically as tasks are accomplished or as the environment changes. One such situation is persistent surveillance subject to battery constraints. We consider the case in which a group of robots with a fully-connected communication graph must maintain coverage over a region while periodically swapping between surveillance paths and charging stations to ensure that no batteries fully discharge. To accomplish this objective, each robot computes a cost value based on battery levels and travel distances. To ensure persistent coverage, robots at surveillance stations can only charge if another robot chooses to swap with the surveillance robot. We consider ten robots maintaining coverage of two surveillance stations. At each assignment episode, we compute the assignments using our proposed ADMM-TA method and compare the resulting cost to the optimal cost obtained from the Hungarian method. Figure 6.14 shows the assignment costs over 1000 assignment episodes and the ADMM-TA method converges to an assignment with the optimal cost at every episode. As the simulation runs, the robots reach an oscillating pattern of swapping between surveillance and charging stations, resulting in a limit cycle of the total cost.

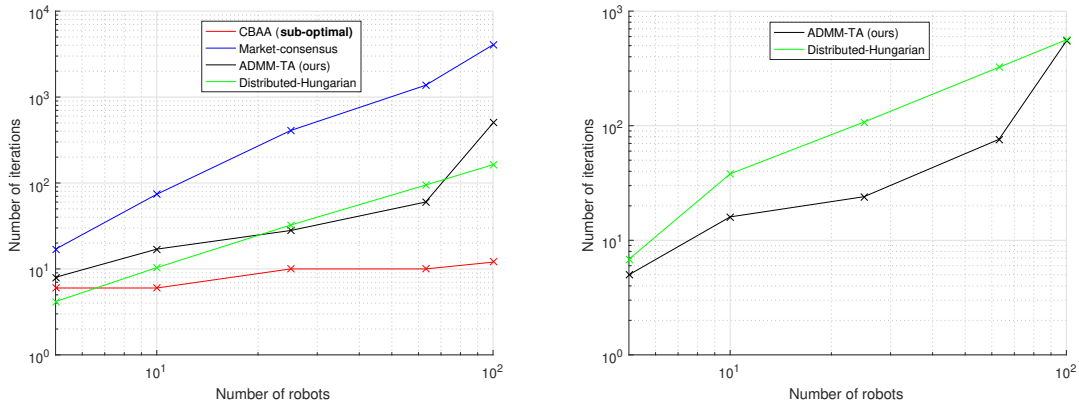

Figure 6.13: (left) Comparison of the ADMM-TA method to other distributed methods for task assignment on a fully-connected graph. The ADMM-TA method produces the optimal solution after fewer iterations compared to the CBAA (sub-optimal) and market-consensus methods. (right) Comparison of the ADMM-TA method to the distributed Hungarian method on randomly-generated connected graphs.

6.4 Summary

In this chapter, we first introduced two different frameworks based on cooperative teams of autonomous aerial vehicles. We showed that it is possible to develop frameworks consisting of large teams which are applied to large GMDPs, and that they be designed to scale independently of number of agents in the team or the size of the GMDP. This property is particularly important as autonomous agents are expected to take over dangerous and difficult jobs currently performed by humans. However, it is also clear that a more general design approach is also needed to enable additional analysis of cooperative autonomous teams, as certain guarantees must be provided to deploy robots in real world scenarios. We took the first steps in addressing a more general problem statement for cooperative team problems in the final framework of this chapter. Our distributed optimizations methods are relatively simple and can easily be analyzed under different conditions. Ultimately, our frameworks will need to be improved by carrying out hardware experiments, since current hardware limitations will dictate algorithmic design choices. As on-board processing power, embedded sensors, and robot design continue to evolve, we expect our algorithms provide a basis to develop suitable frameworks with guarantees for environmental models represented by a graph-based Markov decision process.

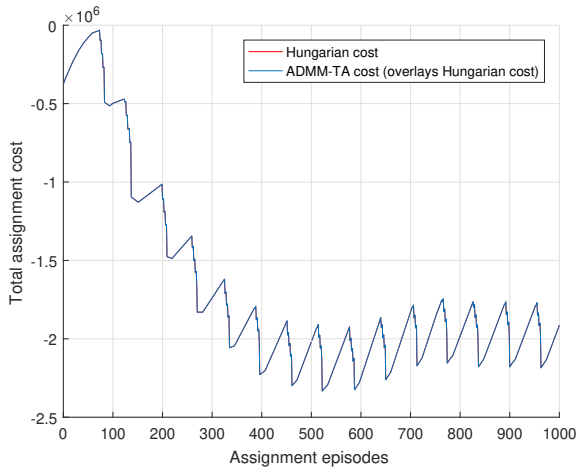

Figure 6.14: The ADMM-TA method on a surveillance task with ten robots and two surveillance stations. The ADMM-TA method produces the optimal cost at all assignment episodes and the costs reach a limit cycle after equilibrium is attained.

Chapter 7

Conclusions and Future Directions

7.1 Current Feasibility and Technologies

There are a number of avenues to apply the methods developed in this thesis to current and future problems. Currently, there are many aerial vehicle platforms, such as quadrotors, which are popular for both consumer and academic use. Thus, an appealing and relatively straightforward setup is to deploy a team of quadrotors in combination with our methods, in particular for assisting in disaster response for forest wildfires. In 2020 alone, California experienced its worst fire season in modern state history [183], in part due to climate change and poor forest management. Furthermore, the number of fires and their intensity have long been predicted to increase [136]. Therefore, a critical short-term goal is to combine our algorithms with readily available hardware to assist first responders in managing and suppressing a forest fire. There are several challenges to producing such a solution, the first of which is to enable aerial robots to survive a caustic environment involving extreme temperatures and hazardous particulates, such as ash and smoke. Prior work on hardware platforms for fighting fires suggests possible strategies to develop suitable aerial vehicles (see [184] and references therein), along with efforts to develop autonomous helicopters to address forest fires [185]. Next, a suitable sensor is also required to gather information and to enable autonomous decision making. A significant amount of work in literature has developed computer vision techniques specifically for fire surveillance, suggesting that sensing modalities from these works could be directly applied [117, 118, 119, 120, 121, 122, 186]. In the long term, we believe our decision making strategies will streamline the response to disasters such as forest fires, by improving coordination and information gathering between robots and human resources. Our other examples, the COVID-19 pandemic and understanding user activity in social networks, are also current pressing problems which we are able to address through the methods in this thesis. We are able to propose models with influence structures and understand how well a given model fits the observed data. At this point, we are able to gain insight from the data and provide recommendations for addressing the

spread of a virus and understanding opinion dynamics in complex networks. Given the increasing prevalence of public data on these important processes, future work can focus on improving model assumptions and developing further analysis techniques.

7.2 Summary

The future of modeling frameworks will include increasingly complex and detailed representations for processes and interactions with robots and humans, for tasks such as disaster response, infrastructure inspection, warehouse automation, and more. In this thesis, we have proposed algorithms to enable the use of large graph-based Markov decision processes (GMDPs) as a modeling framework to address these applications. We considered the necessary problems to eliminate the barriers to using GMDPs: optimally controlling a GMDP with limited control effort, online state estimation of GMDPs given uncertain measurements, and learning the model parameters of a GMDP using time histories of data. We also introduced frameworks that utilized a cooperative team of autonomous agents to illustrate potential design methods and notable challenges.

Modeling approaches in prior work, including network models, have many limitations. Typically, the model is proposed for a specific application, limiting the extension to other processes which may be described by a similar model. In addition, it is difficult to find complete frameworks for a given model, e.g., a control strategy is developed with the assumption that a suitable filter exists to reason about state and model uncertainty. As a result, it is unclear on how to use a given model in a practical setting, where all aspects of the algorithmic pipeline must be considered.

Overall, our algorithms emphasize efficiency and scalability to address significantly large GMDPs, as we have considered discrete models that are substantially larger than those presented in relevant prior work. We have applied our methods to some of today's most pressing issues, including natural disasters and public health crises, and demonstrated a complete algorithmic pipeline to provide solutions to these problems. We believe that our work will spur additional interest in using large structured Markov models to model processes and human-robot interactions. Although we have used some additional structural assumptions, our methodology can be applied to more general model and problem formulations. Continuing to address the challenges of large-scale systems will enable a streamlined approach to future public crises and natural disasters.

7.3 Future Directions

Based on the results of this thesis, we have developed a strong foundation for the general use of GMDPs, and there are still many interesting directions for future work.

7.3.1 Parameterized Policies

For the optimal control problem with a capacity constraint, we derive an approximate dynamic programming approach which is computationally tractable and has a metric to evaluate the approximate quality. Nevertheless, additional structure is required to derive the constrained policy. At a high level, we are more interested in producing a constrained policy which allow for theoretical analysis than having to first produce a value or state-action function. Therefore, it would be instructive to consider policy iteration or similar methods to achieve this. Furthermore, it would be very useful to have a policy parameterized by the model parameters, so that the policy does not need to be recomputed if the model parameters are updated online. While our percolation-based control framework addresses some of these shortcomings, the theoretical analysis is somewhat limited and a structured lattice representation of the GMDP is required. Although percolation techniques exist for a more general class of models called isoradial graphs, it is not straightforward to consider arbitrary graph structure. Investigating relaxations or modifications to achieve this is an interesting direction to explore.

7.3.2 Graph-based POMDPs

With regard to the online filtering problem, we have shown that a fast accurate filter is necessary to enable effective solutions for other problems, though it was necessary to introduce several approximations to derive our method. Continuing to explore variational inference methods with more theoretical analysis or insight would be extremely useful as a result. A more fruitful direction to explore may be factored or graph-based POMDP solution techniques, as considering state uncertainty is a critical component of this modeling framework. Prior work has suggested methods to address factored POMDP representations and it is clear that approximate methods will be required given the complexity of the full problem. Additional research on graph-based POMDPs would also be useful for robotics problems which must deal with model parameter and state uncertainty due to sensor limitations.

7.3.3 Structural Assumptions

Leveraging the Anonymous Influence property has been key for improving the complexity of our algorithms. We note that our methods can also handle general GMDPs without this property, but at the cost of reducing the model size for which the algorithms will finish in reasonable time. Considering hybrid models, where some vertices in the graph do not have the Anonymous Influence property, would be an interesting extension of our work. Furthermore, applying our methods to more applications which may or may not contain Anonymous Influence would help evaluate the effect of using this property as a simplifying assumption.

7.3.4 Additional Models

Additional applications are valuable for furthering validating the ideas in this thesis. One example is modeling and predicting the trajectories of a crowd of people in order to enable the use of socially-aware robots. Graph-based models are a popular representation in this domain and a number of solution methods have been proposed as well, such as recurrent neural networks [187, 188]. A GMDP can be used to represent the social interactions in a crowd by considering a discrete set of possible outcomes, e.g., passing on the left or the right, or choosing to interact, with a particular autonomous agent. This agent-centered model description also allows for reasoning about interactions as a control action, e.g., choosing to interact or to navigate without collision. It may be necessary to allow for a time-varying graph to model the interactions, and prior work suggests methods to incorporate this aspect [26].

7.3.5 Model Abstraction and Validation

We have proposed a variety of models in this thesis, with a varying number of individual MDPs and different formulations of the individual state and action spaces. Striking a balance between fine detail and high-level abstraction is necessary to minimize computational complexity while still providing meaningful insight into a process. Therefore, it will be necessary to develop analysis techniques to understand this trade-off for different models, e.g., modeling a pandemic in the United States at the city, county, or state level. Once we learn a model with data, we must also be able to validate how well it reflects the data and how well we expect the model to predict future trajectories. There is a wealth of literature on model verification and validation techniques, and this work should be leveraged to develop appropriate metrics to evaluate our learned models.

7.3.6 Hardware Experiments

Finding robotics-related applications where it is possible to design and execute hardware experiments would provide essential validation and feedback for our algorithms. A significant amount of insight can be gained using physical hardware due to the limitations of current state-of-the-art sensors and locomotion hardware. We believe experiments with an autonomous fleet of mobile or aerial robots should be the next step to continue the work we have presented.

Bibliography

[1] A. A. Pervozvanskii and I. N. Smirnov, "Stationary-state evaluation for a complex system with slowly varying couplings," *Cybernetics*, vol. 10, no. 4, pp. 603–611, Jul 1974.

[2] V. G. Gaitsgori and A. A. Pervozvanskii, "Aggregation of states in a Markov chain with weak interactions," *Cybernetics*, vol. 11, no. 3, pp. 441–450, May 1975.

[3] F. Delebecque and J.-P. Quadrat, "Optimal control of Markov chains admitting strong and weak interactions," *Automatica*, vol. 17, no. 2, pp. 281 – 296, 1981.

[4] Q. Zhang, G. Yin, and E. K. Boukas, "Controlled Markov chains with weak and strong interactions: Asymptotic optimality and applications to manufacturing," *Journal of Optimization Theory and Applications*, vol. 94, no. 1, pp. 169–194, Jul 1997.

[5] N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L. P. Kaelbling, T. Dean, and C. Boutilier, "Solving very large weakly coupled Markov decision processes," in *AAAI Conference on Artificial Intelligence*, 1998, pp. 165–172.

[6] J. N. Tsitsiklis and B. Van Roy, "Feature-based methods for large scale dynamic programming," *Machine Learning*, vol. 22, no. 1, pp. 59–94, Mar 1996.

[7] C. Boutilier and R. Dearden, "Approximating value trees in structured dynamic programming," in *Proc. 13th International Conference on Machine Learning*. Morgan Kaufmann, 1996, pp. 54–62.

[8] C. Boutilier, R. Dearden, and M. Goldszmidt, "Stochastic dynamic programming with factored representations," *Artificial Intelligence*, vol. 121, no. 1-2, pp. 49–107, 2000.

[9] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999, pp. 467–475.

[10] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," in *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, pp. 689–695.

[11] Z. Ghahramani and M. I. Jordan, "Factorial hidden Markov models," in *Advances in Neural Information Processing Systems 8*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. MIT Press, 1996, pp. 472–478.

[12] A. Pentland, N. Oliver, and M. Brand, "Coupled hidden Markov models for complex action recognition," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*. Los Alamitos, CA, USA: IEEE Computer Society, jun 1997, p. 994.

[13] S. Zhong and J. Ghosh, "A new formulation of coupled hidden Markov models," Tech. Rep., 2001.

[14] X. Boyen and D. Koller, "Tractable inference for complex stochastic processes," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, p. 33–42.

[15] C. Boutilier, T. Dean, and S. Hanks, "Decision-theoretic planning: Structural assumptions and computational leverage," *Journal of Artificial Intelligence Research*, vol. 11, pp. 1–94, 1999.

[16] D. Koller and R. Parr, "Computing factored value functions for policies in structured MDPs," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999, pp. 1332–1339.

[17] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, "Efficient solution algorithms for factored MDPs," *Journal of Artificial Intelligence Research*, vol. 19, pp. 399–468, 2003.

[18] C. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored MDPs," in *Advances in Neural Information Processing Systems*, 2002.

[19] D. P. De Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Operations research*, vol. 51, no. 6, pp. 850–865, 2003.

[20] Z. Feng and E. A. Hansen, "Approximate planning for factored POMDPs," in *Proceedings of the 6th European Conference on Planning*, 2001.

[21] C. Guestrin, D. Koller, and R. Parr, "Solving factored POMDPs with linear value functions," in *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01) workshop on Planning under Uncertainty and Incomplete Information*, 2001, pp. 67–75.

[22] N. Forsell and R. Sabbadin, "Approximate linear-programming algorithms for graph-based Markov decision processes," in *Proceedings of the European Conference on Artificial Intelligence*, 2006, pp. 590–594.

[23] R. Sabbadin, N. Peyrard, and N. Forsell, "A framework and a mean-field algorithm for the local control of spatial processes," *International Journal of Approximate Reasoning*, vol. 53, no. 1, pp. 66–86, 2012.

[24] Q. Cheng, Q. Liu, F. Chen, and A. T. Ihler, "Variational planning for graph-based MDPs," in *Advances in Neural Information Processing Systems*, 2013, pp. 2976–2984.

[25] F. Chen, Q. Cheng, J. Dong, Z. Yu, G. Wang, and W. Xu, "Efficient approximate linear programming for factored MDPs," *International Journal of Approximate Reasoning*, vol. 63, pp. 101–121, 2015.

[26] W. Dong, A. S. Pentland, and K. A. Heller, "Graph-coupled HMMs for modeling the spread of infection," in *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, 2012, pp. 227–236.

[27] K. Fan, C. Li, and K. A. Heller, "A unifying variational inference framework for hierarchical graph-coupled HMM with an application to influenza infection," in *AAAI Conference on Artificial Intelligence*, 2016, pp. 3828–3834.

[28] P. Robbel, F. Oliehoek, and M. Kochenderfer, "Exploiting anonymity in approximate linear programming: Scaling to large multiagent MDPs," in *AAAI Conference on Artificial Intelligence*, 2016, pp. 2537–2543.

[29] R. N. Haksar and M. Schwager, "Controlling large, graph-based MDPs with global control capacity constraints: An approximate LP solution," in *57th IEEE Conference on Decision and Control (CDC)*, Dec 2018, pp. 35–42.

[30] R. N. Haksar, F. Solowjow, S. Trimpe, and M. Schwager, "Controlling heterogeneous stochastic growth processes on lattices with limited resources," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, Dec 2019, pp. 1315–1322.

[31] R. N. Haksar, J. Lorenzetti, and M. Schwager, "Scalable filtering of large graph-coupled hidden Markov models," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 1307–1314.

[32] R. N. Haksar and M. Schwager, "Constrained control of large graph-based MDPs under measurement uncertainty," *IEEE Transactions on Automatic Control (TAC)*, 2020, under review.

[33] ——, "Learning large graph-based MDPs with historical data," *IEEE Transactions on Control of Network Systems (TCNS)*, 2020, in preparation.

[34] ——, "Distributed deep reinforcement learning for fighting forest fires with a network of aerial robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1067–1074.

[35] R. N. Haksar, S. Trimpe, and M. Schwager, "Spatial scheduling of informative meetings for multi-agent persistent coverage," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 3027–3034, April 2020.

[36] R. N. Haksar, O. Shorinwa, P. Washington, and M. Schwager, "Consensus-based ADMM for task assignment in multi-robot teams," in *2019 International Symposium on Robotics Research (ISRR)*, Oct 2019, in press.

[37] I. I. Dikin, "Iterative solution of problems of linear and quadratic programming," *Dokl. Akad. Nauk SSSR*, vol. 174, pp. 747–748, 1967.

[38] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, Dec 1984.

[39] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[40] M. Kochenderfer, *Decision making under uncertainty : theory and application*. Cambridge, Massachusetts: The MIT Press, 2015.

[41] V. Raghavan, G. ver Steeg, A. Galstyan, and A. G. Tartakovsky, "Coupled hidden Markov models for user activity in social networks," in *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2013, pp. 1–6.

[42] D. Blatner, *Spectrums: Our Mind-boggling Universe from Infinitesimal to Infinity*. A&C Black, 2013.

[43] B. Szabó and I. Babuška, *Introduction to Finite Element Analysis: Formulation, Verification and Validation*. Hoboken, N.J.: Wiley, 2011.

[44] Y. Efendiev and T. Hou, *Multiscale finite element methods: theory and applications*. New York, NY: Springer, 2009.

[45] D. Boychuk, W. J. Braun, R. J. Kulperger, Z. L. Krougly, and D. A. Stanford, "A stochastic forest fire growth model," *Environmental and Ecological Statistics*, vol. 16, no. 2, pp. 133–151, Jun 2009.

[46] J. D. Griffith, M. J. Kochenderfer, R. J. Moss, V. V. Misic, V. Gupta, and D. Bertsimas, "Automated dynamic resource allocation for wildfire suppression," *Lincoln Laboratory Journal*, vol. 22, no. 2, pp. 38–59, 2017.

[47] M. A. Finney, "Mechanistic modeling of landscape fire patterns," *Spatial Modeling of Forest Landscapes: Approaches and Applications. Cambridge University Press, Cambridge*, pp. 186–209, 1999.

[48] V. Raghavan, G. Ver Steeg, A. Galstyan, and A. G. Tartakovsky, "Modeling temporal activity patterns in dynamic social networks," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 1, pp. 89–107, 2014.

[49] C. Baier, H. Hermanns, and J.-P. Katoen, *The 10,000 Facets of MDP Model Checking.* Cham: Springer International Publishing, 2019, pp. 420–451.

[50] C. Moore and M. E. J. Newman, "Epidemics and percolation in small-world networks," *Phys. Rev. E*, vol. 61, pp. 5678–5682, May 2000.

[51] C. Favier, "Percolation model of fire dynamic," *Physics Letters A*, vol. 330, no. 5, pp. 396–401, 2004.

[52] J. Balthrop, S. Forrest, M. E. J. Newman, and M. M. Williamson, "Technological networks and the spread of computer viruses," *Science*, vol. 304, no. 5670, pp. 527–529, 2004.

[53] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World.* USA: Cambridge University Press, 2010.

[54] D. Acemoglu and A. Ozdaglar, "Opinion dynamics and learning in social networks," *Dynamic Games and Applications*, vol. 1, no. 1, pp. 3–49, Mar 2011.

[55] World Health Organization (WHO), "Ebola data and statistics," http://apps.who.int/gho/data/node.ebola-sitrep, accessed 2018-03-19.

[56] E. Altman, *Constrained Markov decision processes.* CRC Press, 1999.

[57] D. Adelman and A. J. Mersereau, "Relaxations of weakly coupled stochastic dynamic programs," *Operations Research*, vol. 56, no. 3, pp. 712–727, 2008.

[58] F. Ye, H. Zhu, and E. Zhou, "Weakly coupled dynamic program: Information and Lagrangian relaxations," *IEEE Transactions on Automatic Control*, vol. 63, no. 3, pp. 698–713, 2018.

[59] H.-J. Schütz and R. Kolisch, "Approximate dynamic programming for capacity allocation in the service industry," *European Journal of Operational Research*, vol. 218, no. 1, pp. 239 – 250, 2012.

[60] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[61] D. F. Ciocan and V. Farias, "Model predictive control for dynamic resource allocation," *Mathematics of Operations Research*, vol. 37, no. 3, pp. 501–525, 2012.

[62] C. Nowzari, V. M. Preciado, and G. J. Pappas, "Analysis and control of epidemics: A survey of spreading processes on complex networks," *IEEE Control Systems Magazine*, vol. 36, no. 1, pp. 26–46, Feb 2016.

[63] J. W. Essam, "Percolation theory," *Reports on Progress in Physics*, vol. 43, no. 7, pp. 833–912, July 1980.

[64] M. Sahini and M. Sahimi, *Applications of percolation theory.* CRC Press, 1994.

[65] S. Solomon, G. Weisbuch, L. de Arcangelis, N. Jan, and D. Stauffer, "Social percolation models," *Physica A: Statistical Mechanics and its Applications*, vol. 277, no. 1, pp. 239 – 247, 2000.

[66] A. Somanath, S. Karaman, and K. Youcef-Toumi, "Controlling stochastic growth processes on lattices: Wildfire management with robotic fire extinguishers," in *53rd IEEE Conference on Decision and Control (CDC)*, Dec 2014, pp. 1432–1437.

[67] A. Somanath and S. Karaman, "An optimal randomized policy for controlling stochastic growth processes on lattices," in *55th IEEE Conference on Decision and Control (CDC)*, Dec 2016, pp. 6240–6245.

[68] R. Williams and L. C. Baird, "Tight performance bounds on greedy policies based on imperfect value functions," Tech. Rep., 1993.

[69] H. Kesten, "The critical probability of bond percolation on the square lattice equals 1/2," *Communications in Mathematical Physics*, vol. 74, no. 1, pp. 41–59, 1980.

[70] G. R. Grimmett and I. Manolescu, "Bond percolation on isoradial graphs: criticality and universality," *Probability Theory and Related Fields*, vol. 159, no. 1, pp. 273–327, Jun 2014.

[71] K. B. Athreya and P. E. Ney, *Branching Processes.* Springer, Berlin, Heidelberg, 1972.

[72] D. Assaf, L. Goldstein, and E. Samuel-Cahn, "An unexpected connection between branching processes and optimal stopping," *Journal of applied probability*, vol. 37, no. 3, pp. 613–626, 2000.

[73] A. Fog, "Calculation methods for Wallenius' noncentral hypergeometric distribution," *Communications in Statistics: Simulation and Computation*, vol. 37, no. 2, pp. 258–273, 2008.

[74] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, July 2000.

[75] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings F - Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, 1993.

[76] N. Vaswani, A. Yezzi, Y. Rathi, and A. Tannenbaum, "Particle filters for infinite (or large) dimensional state spaces- part 1," in *IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 3, May 2006, pp. III–III.

[77] T. Seo, T. T. Tchrakian, S. Zhuk, and A. M. Bayen, "Filter comparison for estimation on discretized PDEs modeling traffic: Ensemble Kalman filter and minimax filter," in *55th IEEE Conference on Decision and Control (CDC)*, Dec 2016, pp. 3979–3984.

[78] A. Beskos, D. Crisan, A. Jasra, K. Kamatani, and Y. Zhou, "A stable particle filter for a class of high-dimensional state-space models," *Advances in Applied Probability*, vol. 49, no. 1, p. 24–48, 2017.

[79] A. Jasra, S. S. Singh, J. S. Martin, and E. McCoy, "Filtering via approximate Bayesian computation," *Statistics and Computing*, vol. 22, no. 6, pp. 1223–1237, Nov 2012.

[80] D. M. Blei, M. I. Jordan, and J. W. Paisley, "Variational Bayesian inference with stochastic search," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research. PMLR, 2012, pp. 1367–1374.

[81] M. Hoffman and D. Blei, "Stochastic Structured Variational Inference," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Lebanon and S. V. N. Vishwanathan, Eds., vol. 38. San Diego, California, USA: PMLR, 09–12 May 2015, pp. 361–369.

[82] A. Saeedi, T. D. Kulkarni, V. K. Mansinghka, and S. J. Gershman, "Variational particle approximations," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2328–2356, Jan. 2017.

[83] M. Yin and M. Zhou, "Semi-implicit variational inference," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 5660–5669.

[84] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1303–1347, 2013.

[85] V. Smidl and A. Quinn, "Variational Bayesian filtering," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 5020–5030, 2008.

[86] J. Winn and C. M. Bishop, "Variational message passing," *Journal of Machine Learning Research*, vol. 6, pp. 661–694, Dec. 2005.

[87] P. Poupart and C. Boutilier, "VDCBPI: an approximate scalable algorithm for large POMDPs," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. MIT Press, 2005, pp. 1081–1088.

[88] T. S. Veiga, M. T. J. Spaan, and P. U. Lima, "Point-based POMDP solving with factored value function approximation," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, ser. AAAI'14. AAAI Press, 2014, p. 2512–2518.

[89] J. Pajarinen, J. Peltonen, A. Hottinen, and M. A. Uusitalo, "Efficient planning in large POMDPs through policy graph based factorized approximations," in *Machine Learning and Knowledge Discovery in Databases*, J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–16.

[90] A. H. Jazwinski, *Stochastic processes and filtering theory*. Courier Corporation, 2007.

[91] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[92] J. Hensman, M. Rattray, and N. D. Lawrence, "Fast variational inference in the conjugate exponential family," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2888–2896.

[93] L. K. Saul and M. I. Jordan, "Exploiting tractable substructures in intractable networks," in *Advances in Neural Information Processing Systems 8*. MIT Press, 1996, pp. 486–492.

[94] M. Simon. (2018) The terrifying science behind California's massive Camp fire. [Online]. Available: https://www.wired.com/story/the-terrifying-science-behind-californias-massive-camp-fire/

[95] S. M. Chu and T. S. Huang, "Bimodal speech recognition using coupled hidden Markov models," in *ICSLP-2000*, vol. 2, 2000, pp. 747–750.

[96] I. Rezek and S. J. Roberts, "Estimation of coupled hidden Markov models with application to biosignal interaction modelling," in *Neural Networks for Signal Processing X. Proceedings of the 2000 IEEE Signal Processing Society Workshop (Cat. No.00TH8501)*, vol. 2, 2000, pp. 804–813.

[97] I. Rezek, P. Sykacek, and S. J. Roberts, "Learning interaction dynamics with coupled hidden Markov models," *IEE Proceedings - Science, Measurement and Technology*, vol. 147, no. 6, pp. 345–350, 2000.

[98] J. Kwon and K. Murphy, "Modeling freeway traffic with coupled HMMs," Tech. Rep., 2000.

[99] M. I. Jordan, Z. Ghahramani, and L. K. Saul, "Hidden Markov decision trees," in *Advances in Neural Information Processing Systems 9*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. MIT Press, 1997, pp. 501–507.

[100] R. D. Malmgren, J. M. Hofman, L. A. Amaral, and D. J. Watts, "Characterizing individual communication patterns," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 607–616.

[101] T. Chis and P. G. Harrison, "Modeling multi-user behaviour in social networks," in *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, 2014, pp. 168–173.

[102] A. De, I. Valera, N. Ganguly, S. Bhattacharya, and M. Gomez-Rodriguez, "Learning and forecasting opinion dynamics in social networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 397–405.

[103] C. Williams and G. Hinton, "Mean field networks that learn to discriminate temporally distorted strings," in *Connectionist Models*, 1991, pp. 18–22.

[104] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[105] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[106] A. P. Dunmur and D. M. Titterington, "On a modification to the mean field EM algorithm in factorial learning," in *Advances in Neural Information Processing Systems 9*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. MIT Press, 1997, pp. 431–437.

[107] L. K. Saul and M. I. Jordan, "Mixed memory Markov models: Decomposing complex stochastic processes as mixtures of simpler ones," *Mach. Learn.*, vol. 37, no. 1, p. 75–87, Oct 1999.

[108] California Department of Public Health, "COVID-19 cases," https://data.ca.gov/dataset/covid-19-cases, accessed 2020-10-28.

[109] Textblob, "Textblob: Simplified text processing," https://textblob.readthedocs.io/en/dev/.

[110] E. Summers, "Fake news tweets," https://archive.org/details/fakenews-tweets, accessed 2020-11-04.

[111] A. Hannak, E. Anderson, L. F. Barrett, S. Lehmann, A. Mislove, and M. Riedewald, "Tweetin' in the rain: Exploring societal-scale effects of weather on mood," in *International AAAI Conference on Web and Social Media*, 2012.

[112] J. M. Diaz, "Economic impacts of wildfire," *Southern Fire Exchange*, 2012.

[113] G. D. Richards, "An elliptical growth model of forest fire fronts and its numerical solution," *International Journal for Numerical Methods in Engineering*, vol. 30, no. 6, pp. 1163–1179, 1990.

[114] M. A. Finney, "Farsite: Fire area simulator-model development and evaluation," 1998.

[115] D. Boychuk, W. J. Braun, R. J. Kulperger, Z. L. Krougly, and D. A. Stanford, "A stochastic forest fire growth model," *Environmental and Ecological Statistics*, vol. 16, no. 2, pp. 133–151, 2009.

[116] J. R. Martinez-de Dios, B. C. Arrue, A. Ollero, L. Merino, and F. Gómez-Rodríguez, "Computer vision techniques for forest fire perception," *Image and vision computing*, vol. 26, no. 4, pp. 550–562, 2008.

[117] D. Stipaničev, M. Štula, D. Krstinić, L. Šerić, T. Jakovčević, and M. Bugarić, "Advanced automatic wildfire surveillance and monitoring network," in *6th International Conference on Forest Fire Research', Coimbra, Portugal.(Ed. D. Viegas)*, 2010.

[118] P. B. Sujit, D. Kingston, and R. Beard, "Cooperative forest fire monitoring using multiple UAVs," in *2007 46th IEEE Conference on Decision and Control*, Dec 2007, pp. 4875–4880.

[119] L. Merino, F. Caballero, J. R. Martínez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple UAVs: Application to automatic detection of forest fires," *Journal of Field Robotics*, vol. 23, no. 3-4, pp. 165–184, 2006.

[120] L. Merino, F. Caballero, J. R. Martínez-de Dios, I. Maza, and A. Ollero, "An unmanned aircraft system for automatic forest fire monitoring and measurement," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 533–548, 2012.

[121] D. W. Casbeer, R. W. Beard, T. W. McLain, S.-M. Li, and R. K. Mehra, "Forest fire monitoring with multiple small UAVs," in *Proceedings of the 2005, American Control Conference, 2005.*, June 2005, pp. 3530–3535 vol. 5.

[122] D. W. Casbeer, D. B. Kingston, R. W. Beard, and T. W. McLain, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of Systems Science*, vol. 37, no. 6, pp. 351–360, 2006.

[123] M. Kumar, K. Cohen, and B. HomChaudhuri, "Cooperative control of multiple uninhabited aerial vehicles for monitoring and fighting wildfires," *Journal of Aerospace Computing, Information, and Communication*, vol. 8, no. 1, pp. 1–16, 2011.

[124] C. Phan and H. H. T. Liu, "A cooperative UAV/UGV platform for wildfire detection and fighting," in *2008 Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing*, Oct 2008, pp. 494–498.

[125] L. Ntaimo, J. A. Gallego-Arrubla, J. Gan, C. Stripling, J. Young, and T. Spencer, "A simulation and stochastic integer programming approach to wildfire initial attack planning," *Forest Science*, vol. 59, no. 1, pp. 105–117, 2013.

[126] X. Hu and L. Ntaimo, "Integrated simulation and optimization for wildfire containment," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 19, no. 4, p. 19, 2009.

[127] N. K. Ure, S. Omidshafiei, B. T. Lopez, A. a. Agha-Mohammadi, J. P. How, and J. Vian, "Online heterogeneous multiagent learning under limited communication with applications to forest fire management," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 5181–5188.

[128] K. D. Julian and M. J. Kochenderfer, "Autonomous distributed wildfire surveillance using deep reinforcement learning," in *2018 AIAA Guidance, Navigation, and Control Conference*, 2018, p. 1589.

[129] D. Bertsimas, J. D. Griffith, V. Gupta, M. J. Kochenderfer, and V. V. Mišić, "A comparison of Monte Carlo tree search and rolling horizon optimization for large-scale dynamic resource allocation problems," *European Journal of Operational Research*, 2017.

[130] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 3675–3683.

[131] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *International Conference on Machine Learning*, 2017, pp. 3540–3549.

[132] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[133] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[134] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The robotarium: A remotely accessible swarm robotics research testbed," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1699–1706.

[135] CAL FIRE, "California fire historical incident statistics," http://cdfdata.fire.ca.gov/incidents/incidents_statsevents, Accessed 2019-02-22.

[136] U. G. C. R. Program, "National climate assessment report," https://nca2014.globalchange.gov/report, Accessed 2019-03-05.

[137] Y. Meng, J. V. Nickerson, and J. Gan, "Multi-robot aggregation strategies with limited communication," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 2691–2696.

[138] C. Nowzari and J. Cortés, "Self-triggered coordination of robotic networks for optimal deployment," *Automatica*, vol. 48, no. 6, pp. 1077 – 1087, 2012.

[139] Y. Kantaros and M. M. Zavlanos, "Distributed communication-aware coverage control by mobile sensor networks," *Automatica*, vol. 63, pp. 209 – 220, 2016.

[140] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, no. 1, pp. 707–755, Apr. 2009.

[141] M. Corah and N. Michael, "Efficient online multi-robot exploration via distributed sequential greedy assignment," in *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017.

[142] B. J. Julian, M. Angermann, M. Schwager, and D. Rus, "A scalable information theoretic approach to distributed robot coordination," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 5187–5194.

[143] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

[144] S. Moon and E. W. Frew, "A communication-aware mutual information measure for distributed autonomous robotic information gathering," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3137–3144, Oct 2019.

[145] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to persistent monitoring tasks," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, Oct 2016.

[146] C. Phan and H. H. T. Liu, "A cooperative UAV/UGV platform for wildfire detection and fighting," in *2008 Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing*, Oct 2008, pp. 494–498.

[147] K. D. Julian and M. J. Kochenderfer, "Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 8, pp. 1768–1778, 2019.

[148] S. Susca, F. Bullo, and S. Martinez, "Monitoring environmental boundaries with a robotic sensor network," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 288–296, March 2008.

[149] S. W. Feng, S. D. Hand, and J. Yu, "Efficient algorithms for optimal perimeter guarding," in *Proceedings of Robotics: Science and Systems*, June 2019.

[150] L. Merino, F. Caballero, J. R. Martínez-de Dios, I. Maza, and A. Ollero, "An unmanned aircraft system for automatic forest fire monitoring and measurement," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 533–548, Jan 2012.

[151] N. K. Ure, S. Omidshafiei, B. T. Lopez, A. Agha-Mohammadi, J. P. How, and J. Vian, "Online heterogeneous multiagent learning under limited communication with applications to forest fire management," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 5181–5188.

[152] M. Kumar, K. Cohen, and B. Homchaudhuri, "Cooperative control of multiple uninhabited aerial vehicles for monitoring and fighting wildfires," *Journal of Aerospace Computing, Information, and Communication*, vol. 8, no. 1, pp. 1–16, 2011.

[153] H. X. Pham, H. M. La, D. Feil-Seifer, and M. C. Deans, "A distributed control framework of multiple unmanned aerial vehicles for dynamic wildfire tracking," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–12, 2018.

[154] P. B. Sujit, D. Kingston, and R. Beard, "Cooperative forest fire monitoring using multiple UAVs," in *2007 46th IEEE Conference on Decision and Control*, Dec 2007, pp. 4875–4880.

[155] K. A. Ghamry, M. A. Kamel, and Y. Zhang, "Cooperative forest monitoring and fire detection using a team of UAVs-UGVs," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2016, pp. 1206–1211.

[156] P. Stone and M. Veloso, "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork," *Artificial Intelligence*, vol. 110, no. 2, pp. 241 – 273, 1999.

[157] N. Roy and G. Dudek, "Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations," *Autonomous Robots*, vol. 11, no. 2, pp. 117–136, Sep 2001.

[158] D. W. Casbeer, D. B. Kingston, R. W. Beard, and T. W. McLain, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of Systems Science*, vol. 37, no. 6, pp. 351–360, 2006.

[159] M. Lauri and R. Ritala, "Planning for robotic exploration based on forward simulation," *Robotics and Autonomous Systems*, vol. 83, pp. 15 – 31, 2016.

[160] T. P. Hill, "Conflations of probability distributions," *Transactions of the American Mathematical Society*, vol. 363, no. 6, pp. 3351–3372, 2011.

[161] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering problem: A survey of recent variants, solution approaches and applications," *European Journal of Operational Research*, vol. 255, no. 2, pp. 315 – 332, 2016.

[162] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[163] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, 2010.

[164] T. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2015.

[165] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in *51st IEEE Conference on Decision and Control (CDC)*, 2012, pp. 5445–5450.

[166] S. Hosseini, A. Chapman, and M. Mesbahi, "Online distributed ADMM via dual averaging," in *53rd IEEE Conference on Decision and Control (CDC)*, 2014, pp. 904–909.

[167] S. Park, Y. Min, J. Ha, D. Cho, and H. Choi, "A distributed ADMM approach to non-myopic path planning for multi-target tracking," *IEEE Access*, vol. 7, pp. 163 589–163 603, 2019.

[168] S. Choudhary, L. Carlone, H. I. Christensen, and F. Dellaert, "Exactly sparse memory efficient SLAM using the multi-block alternating direction method of multipliers," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1349–1356.

[169] R. Van Parys and G. Pipeleers, "Online distributed motion planning for multi-vehicle systems," in *2016 European Control Conference (ECC)*, 2016, pp. 1580–1585.

[170] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[171] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optimization*, vol. 19, pp. 79–102, 2016.

[172] A. Falsone, K. Margellos, and M. Prandini, "A decentralized approach to multi-agent MILPs: Finite-time feasibility and performance guarantees," *Automatica*, vol. 103, pp. 141–150, 2019.

[173] A. Testa, A. Rucco, and G. Notarstefano, "A finite-time cutting plane algorithm for distributed mixed integer linear programming," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 3847–3852.

[174] M. Bürger, G. Notarstefano, F. Bullo, and F. Allgöwer, "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments," *Automatica*, vol. 48, no. 9, pp. 2298–2304, 2012.

[175] S. Giordani, M. Lujak, and F. Martinelli, "A distributed algorithm for the multi-robot task allocation problem," in *Trends in Applied Intelligent Systems*, N. García-Pedrajas, F. Herrera, C. Fyfe, J. M. Benítez, and M. Ali, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 721–730.

[176] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, "A distributed version of the Hungarian method for multirobot assignment," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 932–947, 2017.

[177] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas, "A distributed auction algorithm for the assignment problem," in *2008 47th IEEE Conference on Decision and Control*, 2008, pp. 1212–1217.

[178] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Distributed multi-robot task assignment and formation control," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 128–133.

[179] H. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[180] L. Liu and D. Shell, "Optimal market-based multi-robot task allocation via strategic pricing," in *Proceedings of Robotics: Science and Systems*, 2013.

[181] D. Bertsekas, *Convex analysis and optimization*. Belmont, Mass: Athena Scientific, 2003.

[182] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[183] H. Yan, C. Mossburg, A. Moshtaghlan, and P. Vercammen, "California sets new record for land torched by wildfires as 224 people escape by air from a 'hellish' inferno," Sept 2020. [Online]. Available: https://www.cnn.com/2020/09/05/us/california-mammoth-pool-reservoir-camp-fire/index.html

[184] V. Nikitin, S. Golubin, R. Belov, V. Gusev, and N. Andrianov, "Development of a robotic vehicle complex for wildfire-fighting by means of fire-protection roll screens," *IOP Conference Series: Earth and Environmental Science*, vol. 226, p. 012003, feb 2019.

[185] A. Davies, "These self-flying helicopters team up to fight fires and save lives," Oct 2017. [Online]. Available: https://www.wired.com/2016/11/lockheed-martin-kmax-sara-indago-firefighting-drones/

[186] J. M. de Dios, B. Arrue, A. Ollero, L. Merino, and F. Gómez-Rodríguez, "Computer vision techniques for forest fire perception," *Image and Vision Computing*, vol. 26, no. 4, pp. 550 – 562, 2008.

[187] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 961–971.

[188] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4601–4607.