Intelligent Knowledge Distribution for Multi-Agent Communication, Planning, and Learning

Michael C. Fowler

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Ryan K. Williams, Co-chair
T. Charles Clancy, Co-chair
Cameron D. Patterson
Pratap Tokekar
Michael J. Roan

April 28, 2020
Blacksburg, Virginia

Keywords: Multi-agent System, Distributed Decision Making, Markov Decision Processes, Relational Learning, Probabilistic Constraint Satisfaction, Wireless Communications

Intelligent Knowledge Distribution for Multi-Agent Communication, Planning, and Learning

Michael C. Fowler

(ABSTRACT)

This dissertation addresses a fundamental question of multi-agent coordination: what information should be sent to whom and when, with the limited resources available to each agent? Communication requirements for multi-agent systems can be rather high when an accurate picture of the environment and the state of other agents must be maintained. To reduce the impact of multi-agent coordination on networked systems, e.g., power and bandwidth, this dissertation introduces new concepts to enable Intelligent Knowledge Distribution (IKD), including Constrained-action POMDPs (CA-POMDP) and concurrent decentralized (CoDec) POMDPs for an *agnostic plug-and-play* capability for fully autonomous systems.

Each agent runs a CoDec POMDP where all the decision making (motion planning, task allocation, asset monitoring, and communication) are separated into concurrent individual MDPs to reduce the combinatorial explosion of the action and state space while maintaining dependencies between the models. We also introduce the CA-POMDP with action-based constraints on partially observable Markov decision processes, rewards driven by the value of information, and probabilistic constraint satisfaction through discrete optimization and Markov chain Monte Carlo analysis. IKD is adapted real-time through machine learning of the actual environmental impacts on the behavior of the system, including collaboration strategies between autonomous agents, the true value of information between heterogeneous systems, observation probabilities and resource utilization.

Intelligent Knowledge Distribution for Multi-Agent Communication, Planning, and Learning

Michael C. Fowler

(GENERAL AUDIENCE ABSTRACT)

This dissertation addresses a fundamental question behind when multiple autonomous systems, like drone swarms, in the field need to coordinate and share data: what information should be sent to whom and when, with the limited resources available to each agent? Intelligent Knowledge Distribution is a framework that answers these questions. Communication requirements for multi-agent systems can be rather high when an accurate picture of the environment and the state of other agents must be maintained. To reduce the impact of multi-agent coordination on networked systems, e.g., power and bandwidth, this dissertation introduces new concepts to enable Intelligent Knowledge Distribution (IKD), including Constrained-action POMDPs and concurrent decentralized (CoDec) POMDPs for an *agnostic plug-and-play* capability for fully autonomous systems. The IKD model was able to demonstrate its validity as a "plug-and-play" library that manages communications between agents that ensures the right information is being transmitted at the right time to the right agent to ensure mission success.

Dedication

This is dedicated to my wife and children, who provided me encouragement and strength despite the sacrifices they made so that I could have time to pursue a Ph.D.

Acknowledgments

There are many people who I owe a debt of gratitude and appreciation in their support during these years. I definitely owe a lot to my beautiful wife, who has had to raise four kids with less support from me than she deserved while her parents took up the slack. To my father, Mike, for encouraging me in this effort and assisting my family when needed. To my advisor, Ryan Williams, for providing the necessary mentoring as my area of focus settled into software and machine intelligent. To my co-advisor, Charles Clancy, for taking the leap of trust that I would be a successful Ph.D. student when his reputation was also at stake. To Dr. Jerry Park for setting the right expectations and mindset to ensure I achieved something to be proud of. Finally to my friend, Odge Hutton, for being a positive and encouraging driving force behind completing the requirements for this degree in a "timely" fashion.

Contents

List of Figures

List of Tables

List of Abbreviations

ADC Analog to Digital Converter

AI Artificial Intelligence

AOH Action Observation Histories

AP Access Point

BnB Branch and Bound

BYOD Bring Your Own Device

CA-POMDP Constrained-Action Partially Observable Markov Decision Process

CCN Content Centric Networking

CDF Cumulative Distribution Function

CG Coordination Graph

CISL Concurrent Individual and Social Learning

CMDP Constrained Markov Decision Process

CoDec POMDP Concurrent Decentralized POMDP

CoDec POMDP Concurrent Decentralized Partially Observable Markov Decision Process

CoMDP Concurrent MDPs

CPOMDP Constrained Partially Observable Markov Decision Process

CPS Cyber Physical Systems

CRP Chinese Restaurant Process

CSP Constraint Satisfaction Problem

DBN Dynamic Bayesian Network / Deep Belief Network

DBSCAN Density-based spatial clustering of applications with noise

Dec-MDP Decentralized Markov Decision Processes

Dec-POMDP Decentralized Partially Observable Markov Decision Process

DFS Depth First Search

DNN Deep Neural Network

DoD Deparment of Defense

EKF Extended Kalman Filters

EPC Electronic Product Codes

FANET Flying Ad-Hoc Networks

FSC Finite State Controller

GPS Global Positioning System

HMDP hierarchical Markov Decision Process

HMM Hidden Markov Model

iid independent and identically distributed

IKD Intelligent Knowledge Distribution

IoT Internet of Things

IRM Infinite Relational Model

ISTAR Intelligence, surveillance, target acquisition, and reconnaissance

KF Kalman Filters

KL Kullback-Leibler

KMDP Kalman Markov Decision Process

LE Linear Equations

LMI Linear Matrix Inequality

LP Linear Programming

M2M Machine-to-Machine

MANET Mobile Ad-hoc Networks

MAPE Monitor, Analyze, Plan, and Execute

MAS Multiagent Systems

MCMC Markov chain Monte Carlo

MDP Markov Decision Process

MILP Mixed Integer Linear Programming

MSN Mobile Sensor Networks

ND-POMDP Networked Distributed Partially Observable Markov Decision Process

NDN Named Data Networking

NLP Nonlinear Programming

PBVI Point-based Value Iteration

PGM Probabilistic Graphical Model

POSG Partially Observable Stochastic Games

RTOS Real-Time Operation Systems

SCP Stochastic Constraint Programming

SDN Software-defined Networking

SWAP Size, Weight, and Power

UAANET Unmanned Aeronautical Ad-hoc Networks

UAV Unmanned Aerial Vehicles

uCode Ubiquitous Codes

UGV Unmanned Ground Vehicle

UKF Unscented Kalman Filters

VOI Value of Information

WSN Wireless Sensor Networks

Chapter 1

Introduction

1.1 Motivation

Coordination and collaboration between multiple autonomous systems in the field often creates a significant burden on communication systems in order for them to share environmental and state information to achieve global objectives. To provide resiliency to systems, decentralized or distributed approaches have been implemented that allow for an agent to act independently with little to full information of the other agents' states and actions. The key concepts and contributions of our approach is to provide a framework for *Intelligent Knowledge Distribution* which decides on what information is transmitted to whom and when, while minimizing the impact this coordination between agents will have on the limited resources available to each agent, such as power or bandwidth.

In disaster response or military operations, remote sensing by autonomous systems provides a stream of data back to critical personnel. For example, Figure 1.1 shows a military squad collaborating with autonomous system for the DARPA Squad-X program [1] to assist with intelligence, surveillance, target acquisition, and reconnaissance (ISTAR). A significant amount of time and bandwidth is used to direct these assets to locations that can assist their human counterparts in executing their duties, but if these assets can autonomously determine where they are needed and what information they need then we free up individuals to perform their primary responsibilities. Disaster recovery and other sensor based applications

Figure 1.1: Multi-agent Scenario for the collaobration of heterogeneous assets in an infranty-level military operation [1]

rely on high bandwidth links that are stretched to their limits under unpredictable channel and link qualities [10] in support of infrared, electrooptical, video surveillance, chemical sensors, radar, electronic sensing, etc. At the same time, the communications necessary for multiple agents to coordinate in support of a mission needs to stay within a constrained allowance so as to minimize impact on the systems performance or the overall information flow in the system. Along with common bandwidth constraints, unmanned aerial vehicles (UAV) for example have the additional restriction of minimizing the utilization of energy to maximize their flight time in support of field operations.

Intelligent Knowledge Distribution builds upon multiple sequential decision making algorithms that integrate into a framework that allows for the long-term benefit of a series of actions while reducing the state space explosion of complex models. At the core of an individual's decision making is a concurrent decentralized partially observable Markov decision process (CoDec POMDP), where multiple MDPs are solved concurrently with dependencies

between the MDPs in states, transitions, and rewards. Unlike many models [11, 12], the multiple MDPs of an agent are not in a hierarchical structure where a higher order MDP decides to execute and relinquish control to a lower-level MDP waiting for an end or goal state to return decision making control, but the CoDec POMDP has the MDPs running simultaneously and their solutions are dependent on the model information from other MDPs. The CoDec MDP reduces the state space for team learning and the complexity of solving for the policies by providing a factored representation not in the MDP itself but across MDPs. In their work on concurrent MDPs (CoMDPs), the authors used advice exchange for collaboration between agents in a team foraging problem which has a large communication overhead and assumes homogeneous agents [13].

We assume each agent is executing its own decision making in a decentralized or distributed multi-agent network as in decentralized POMDPs (Dec-POMDP) [12, 14, 15], where the agent has local state observations of itself but local indirect observations of other agents and the environment. Therefore, an agent can run independently without the necessity of communicating or having information of other agent's states or rewards, but this will lead to sub-optimal behavior as tasks are not being covered by the agents. The key is to determine what information another agent will need at what point in time to improve the overall efficiency of the actions or tasking without assuming unconstrained resource availability, e.g., communication are instantaneous and free [16].

In this work, we introduce a new MDP concept in the CoDec MDP, a Constrained-Action POMDP (CA-POMDP) for *Intelligent Knowledge Distribution* (IKD), which was validated using a model of monitoring of ground assets during a disaster response to ensure they safely avoid hazards and dangerous situations they might not be aware of from their perspective on the ground. The constraints are soft and the resulting infinite-horizon policy is evaluated as the probability it will not violate the constraint during a period of time. Soft constraints are

often relevant in communication decisions over hard constraints for multiple reasons: (i) infinite-horizon POMDPs are often represented as a finite state controller [17] and evaluating hard constraints on cyclic graph is intractable; (ii) resource availabilities are not instantaneous but long-term as with total battery power to maintain flight times needed for safety monitoring; and, (iii) available bandwidth in the network can handle increased messages caused by collaboration due to network protocols like ZeroMQ [18], which permit queueing techniques to account for myopic link saturation.

The CA-POMDP utilizes the concept of utilizing Kullback-Leibler (KL) Divergence to determine the *value of information* and heuristic bounds for a collaboration award between agents. The value of information maintains a belief state for the local agent and a belief state for each of the other agents on their current collaboration. As time passes and collaboration and communication between the two agents doesn't occur, the local agent updates the other agent's belief state to worsen increasing the separation between the two belief probabilities and therefore increasing the value of information that the agent may collect. This is expressed as an upper and lower heuristic bound on the probability of communication to inform the agent on when the system is approaching a point where mission objectives are at risk.

The primary CoDec POMDP architecture of Intelligent Knowledge Distribution interweaves information sources that an autonomous agent needs to make informed local decisions. This permits the systems to work together to learn collaboration strategies, to reduce model complexity by only communicating with those that are critical, and the value of information they have or need from other agents. Simultaneously, the CA-POMDP component of the CoDec POMDP is also adapting its controller through machine learning in response to real observation probabilities and resource utilization as environmental and state information changes.

1.2 Innovative Contributions

Collaboration between multiple heterogeneous autonomous agents with limited resources to maintain a high-level of joint cooperation required the development of a new framework, *Intelligent Knowledge Distribution*. The innovative contributions required to develop IKD is listed below and included not only a new decision making algorithm but also included incorporating into the framework an *agnostic* approach and online learning so it can adapt the controller to meet environmental reality of collaboration strategies and resources utilization.

1. **Constrained-Action Partially Observable Markov Decision Process**

   The development of Intelligent Knowledge Distribution required a fundamentally new algorithm for the development of policies from a POMDP. Applying constraints to the state space of a POMDP has been developed, but the constraints in IKD are action based as they are the component that utilizes the limited resources of an autonomous system. For the decision making to determine who, what, and when information should be shared with other autonomous systems, IKD needed at its core the ability to support Constrained-Action POMDPs. The CA-POMDP is an approach that will solve for any action-based constraints that does not depend on transforming or reformulating a problem so that the constraints are represented in the state space or reward function, which is not allows feasible as with IKD. A policy iteration algorithm, which results in a policy represented as a finite state controller (FSC), was therefore adapted and expanded to handle solving constrained-action POMDPs. With detailed information from solving for an optimal policy, the CA-POMDP algorithm transforms the FSC policy to be compliant to probabilistic constraint satisfaction through discrete optimization, policy evaluation, and an innovative approach to constraint evaluation of an FSC.

2. **Constraint Evaluation of a Finite State Controller**

   A method was necessary to evaluate the FSC policy of a POMDP for its probabilistic constraint satisfaction so that the FSC policy could be transformed to provide compliance. The most common method for calculating POMDPs is value iteration, but evaluating the policy that results from value iteration was not feasible. Therefore, policy iteration was used to obtain a policy that is represented as a finite state controller, which allows the algorithm to sample the FSC for probabilistic constraint satisfaction. Random sampling of the FSC policy required the application of a Markov chain Monte Carlo algorithm because an FSC policy is a representation of a Markov process where samples are therefore not independent and identically distributed (iid). The MCMC sampling of the FSC provides the optimization algorithm with a probability that the policy will be compliant by examining the cumulative probabilities bounded by the constraints.

3. **Intelligent Knowledge Distribution**

   Intelligent Knowledge Distribution has been formulated with a CA-POMDP model to calculate a communication policy from information provided by a designer or within the concurrent decentralized POMDP (CoDec POMDP) architecture for fully autonomous systems. The CA-POMDP model for IKD defines two core components: relevance and collaboration. Relevance is an information theoretic evaluation of the value of the information obtained from the environment compared to previous information. Collaboration is a function of the communication between agents and is not only based on the information theoretic evaluation of global belief states but also on an evaluation of the heuristic bounds that drive mission success. As the information being observed and the number of agents that are collaborating increase, the combinatorial explosion drastically increases the computational resources needed to solve. Therefore,

various approaches in the application of coordination graphs were explored, especially relational learning.

4. **Concurrent Decentralized Partially Observable Markov Decision Processes**
   After formulating an approach for Intelligent Knowledge Distribution, it was necessary to provide a framework for incorporating the CA-POMDP for IKD into an approach for a fully autonomous system that included individual tasking, task allocation, and asset monitoring capabilities. Therefore, the CA-POMDP model for IKD was expanded to incorporate a model for receiving and disseminating information from other concurrent MDPs that are operating the drone and will have information to share with other agents. This includes transfer learning of new tasks experienced by the individual agent, observations and models learned about an asset being monitored, and task allocation observations and rewards.

5. **Relational Learning for Coordination Graphs**
   There are three primary approaches to creating coordination graphs to reduce the combinatorial explosion of increased agents and data: spatial, belief-dependence, and relational. The first two have been well established, but relational learning has not been a focus for coordination graphs. Spatial learning (e.g., geographical clustering) is overly simplistic which does not account for the complexities in a heterogenous system and belief-dependence is complex. Therefore, an ontological approach to coordination that learns relationships between agents and data types to develop semantic knowledge of the data the agents are communicating between each other to form a coordination graph for multi-agent collaboration was developed.

1.3 Related Work

Capitan's paper [16] is a key comparison in analyzing the performance of our paper in a multi-agent coordination setting. However, [16] assumes that point-to-point communications are instantaneous and cost-free between nodes, while applying a collaboration policy similar to consensus. In this work, we instead remove the assumption of instantaneous and cost-free communication to yield multi-agent communication that respects resource constraints. This results in knowledge distribution that is more nuanced than consensus, i.e., data flooding vs. putting data where it needs to be.

Regarding constraints in decision-making, constrained Markov Decision Processes (MDPs) have been used historically to solve two major drawbacks of standard discrete MDPs: Multiple objectives and limited resources [19, 20, 21, 22, 23, 24, 25, 26]. Applying constraints to MDPs has been well established in restricting utilization of states to prevent collisions [21], and has been expanded to hierarchical cases to reduce the complexity of the linear programming formulation [22, 23]. The primary objective of the Constrained MDP (CMDP) is to find a policy that is within a restricted state, cost, and/or reward structure of an unconstrained MDP. Constrained POMDPs (CPOMDPs), which provide constraints for partially observable environments, have also been explored using mixed integer linear programming (MILP) [25, 26]. Both [25] and [26] solve the POMDP using value iteration, though [26] also describes a point-based value iteration (PBVI) approach to provide an upper bound to a heuristic search. [25] also introduces a policy iteration approach based upon a deterministic finite-state controller and policy improvement from [24]. An online algorithm has also been proposed [26] to address drawbacks related to the propagation of risk aversion. A finite horizon policy is computed off-line with a "constraint-penalty-to-go" for every step of dynamic programming value iteration as defined in [25] for solving the MILP.

The above approaches to CMDPs and CPOMDPs apply constraints to the state-space of the model and projection into the value space, but our formulation requires *action-based* constraints as we are limiting the utilization of resources that an action consumes which cannot be tied to physical constructs, such as states that represent "no-fly" or "stay-away" zones. As many scenarios require an indefinite length of operation and no predefined goal states (as in our case study), our CA-POMDP approaches needs to be solved as an infinite-horizon policy. Infinite horizon POMDPs are solved by policy iteration with a finite state controller representation [27], and thus in our context will require analyzing how a cyclic graph utilizes resources with respect to soft constraints. To the best of the authors' knowledge, it is not possible to represent soft resource constraints on actions of a cyclic controller in the state or value space of state-of-the-art CMDPs and CPOMDPs.

The modeling of multi-agent systems (MASs) that have common objectives can be represented as a Decentralized Markov decision process (Dec-POMDP). A Dec-POMDP is a construct that allows multiple independent POMDPs running on different agents to act independently while working towards an objective function that is dependent on all the agents' actions [14]. In brief, the agents only have access to their local observations and as the objective function is dependent on the behavior of all the agents, each agent must maintain a *belief* of the other agents policies. There are many subclasses of the Dec-POMDP that address joint and local observations, communications, model independence, etc [6, 14]. The Dec-POMDP-Comm is the dominant subclass related to this paper where the agents are reasoning on when to communicate due to delays and costs. The most commonly seen approach is to make the decision a part of Dec-POMDP model based upon joint state and actions [14]. There is also an approach where there is a centralized POMDP plan that is communicated between the agents which triggers communications when their individual Dec-POMDP plan differs from the centralized one. Unlike Dec-POMDP-Comm, our CA-POMDP formulation

does not utilize communications to inform the joint policy of multiple agents driving communications, instead the model is an *agnostic "plug-and-play"* approach that separates other collaborative tasks of an agent from communication decisions while enforcing soft resource constraint satisfaction.

Driving communications from the *value of information* similar to CA-POMDP is not unique, but has been the focus of research behind reward shaping and belief-dependent rewards [28, 29]. These techniques use information-theoretic measures between probability distributions, like KL-Distance, as part of the reward function for belief-dependent rewards [29] and for determining communication [28]. [28] focuses on restricting communications to when they are needed but does not provide soft constraint satisfaction to the policy controller which is the focus of this work. The authors in [28] use the Dec-POMDP-Comm as their basis but change the perspective from a cost to a reward for communication to highlight an opportunity to use a resource. They state efficient policy generation techniques will be adapted to allow for scalability, where they use the information-theoretic concepts from Dec-POMDP-Value-Comm [30] as a measure of belief divergence. Alternatively, [31] purposely restructures the action space so they can remain in a classic POMDP problem. In this paper, we approach the IKD problem similarly to [31] where information rewards drive cooperative perception, but instead of restructuring the action space we restructure the observation model to describe belief on the value of information.

To search for an FSC policy that meets probabilistic constraint satisfaction, CA-PODMP needs to solve a constrained optimization problem. In the realm of traditional constraint satisfaction problems (CSP), there are multiple approaches that have been developed to address uncertainty in the constraints or in the environment. Constraint satisfaction problems often deal with solving the optimal value of a function where none of the constraints can be relaxed and therefore when the system is over-constrained no solution is feasible [32] or it produces

a conservative formulation, especially when the solution is in the vicinity of the constraints [33]. As a result, the constrained algorithm determines a preference rating with an ordering of constraints or the solution that satisfies the most constraints [32]. Possibilistic Constraint Satisfaction [34] and Stochastic Constraint Programming (SCP) [35] are both approaches that utilize the backtrack algorithm to solve probabilistic constraint satisfaction with SCP also utilizing forward checking to prune values [35]. The backtracking algorithm can be seen as analogous to the branch and bound methodology used in CA-POMDP with the exception that branch and bound searches for the optimal solution and can support depth first (DFS), breadth-first (BFS), and best-first searches.

Concurrent MDPs (CoMDP) were developed by Girard to provide a mechanism to isolate the individual and task allocation planning algorithms [13], which is based on Concurrent Individual and Social Learning (CISL) [36]. The Concurrent Decentralized (CoDec) POMDP used for IKD in this paper builds on and adapts the CMDP approach by including communications as one of the planning algorithms along with the option of asset monitoring and prediction. Though CMDP is based on CISL, CoDec does not utilize social learning nor advice sharing, but does not preclude their future incorporation as the main key is the intelligent distribution of data among collaborative agents.

1.4 Rationale for Action-based Constraints

It is important to explain the rationale around the necessity of utilizing action-based constraints rather than formulating the problem within Constrained MDP and POMDP models. Basically, the application of constrained MDP and POMDPs as previously discussed is not relevant to the situation because of the basic concept of Intelligent Knowledge Distribution (IKD). With IKD, we are dealing with information states and collaboration, where the

states are more abstract and not particularly concrete making direct correlations to states and rewards difficult. It can be difficult even within the constrained models because of the difficulty to reliably depend on the predictability of the states and transitions. For example, a drone in substantial wind may not have a state transition to a desired location based upon an action because of the strength of the wind.

With IKD, we are dealing with information states that are quantized into discrete states from clustering algorithms for the desired size of the state space. When the system has multiple state transitions $s \rightarrow s'$, there is a high likelihood the system might stay within the same state $s = s'$, but the action a_{t-0} still utilized the limited resources $u_h(a_{t=0})$ available to the system that might be different per action $a \in A$. In partially observable environments, the state transition $s \rightarrow s'$ is not even directly observable but consists of a probability distribution over a belief b of what state the system might be in that is maintained. The difficulty with previous approaches is the accounting of the resource utilization of multiple actions $\pi = \{a_{t=0}, a_{t=1}, \ldots, a_n$ without an observation or state transition $s \rightarrow s'$.

Even more importantly, CA-POMDPs are looking at the probability of meeting *soft constraints*. A system might be able to easily support instantaneous increases in resource utilization. For example, an operator is interested in the overall flight time of a drone and the use of communications will not cause an excess in power availability, but they are interested in the frequency or over utilization of a resource that would impact that flight time. The constrained MDP and POMDPs consider hard constraints and stochastic process have no guarantee of meeting hard constraints. Therefore, a probabilistic approach that takes into account action-based constraints is needed and a solution is provided here.

1.5 Dissertation Outline

This dissertation is divided in such a way as to provide an ease in understanding Constrained-Action POMDPs and Intelligent Knowledge Distribution with respect to its motivation, background information, critical frameworks, experiments, and results. In Chapter 2, we provide an understanding of our motivations with mobile sensor systems. There are unique challenges that research is trying to resolve in the architecture, management, and utilization of every increasingly powerful sensor systems from wireless sensor networks to unmanned systems. With the drastic number of internet connected devices predicted by many authors, the necessity of pushing computing to the edge to prevent network saturation and overloading of cloud computing resources requires more intelligent edge devices.

Sequential decision making is at the core of the framework and therefore Chapter 3 introduces the concepts behind modeling long-term strategic decisions and uncertainty. Markov Decisions Processes (MDPs) are introduced including their variants that allow for handling uncertainty in the either the environment, states, or the model. This includes hierarchical and concurrent MDPs that are used to reduce the combinatorial explosion of complex state and observation models and Decentralized MDPs which handle multi-agent collaborations and their uncertainties

As discussed in the Introduction (1.1) and highlighted in Innovative Contributions (1.2), the core decision making component of IKD is the Constrained-Action POMDP (CA-POMDP). In Chapter 4, the CA-POMDP model is covered along with the how to solve for an infinite horizon policy with probabilistic constraint satisfaction of action-based resource utilization.

In Chapter 5, everything is pulled together to provide a formalization and approach for utilizing CA-POMDPs for Intelligent Knowledge Distribution in coordination with concurrent autonomous algorithms. This includes how IKD adapts to handle unknown environments

to establish collaboration strategies with unknown agents and determine if information you have or another agent has is of value to their needs.

All the experiments and evaluations performed to validate the CA-POMDP and IKD are covered in Chapter 6. The CA-POMDP was validated using a disaster monitoring scenario, where multiple UAVs are collaborating over protecting a ground asset from approaching hazards it is not aware of from its perspective on the ground. To highlight the utilization of IKD within multi-agent fully autonomous systems, a framework for implementing target tracking, where multiple UAVs are tracking multiple assets while trying to maintain coverage and predict its short-term behavior, is described.

1.6 Relevant Publications

Journal papers:

1. **M. C. Fowler**, T. C. Clancy, and R. K. Williams, "Intelligent Knowledge Distribution: Constrained-action POMDPs for Resource-Aware Multi-agent Communication," *Cybernetics, IEEE Transactions on*, 2020 [Accepted with Revisions].

2. **M. C. Fowler**, T. C. Clancy, and R. K. Williams, "Intelligent Knowledge Distribution: Constrained-action POMDPs for Resource-Aware Multi-agent Communication," *arXiv preprint arXiv:1903.03086*, 2019.

3. **M. C. Fowler**, T. C. Clancy, and R. K. Williams, "Intelligent Knowledge Distribution: Concurrent Decentralized POMDPs for Multi-Agent Planning & Resource-Aware Communication," *Robotics, IEEE Transactions on*, [In Progress].

Directly related Patents and Publications:

1. **M. Fowler** and R. Williams, "Intelligent Distribution of Data for Robotic and Autonomous Systems," International Publication No. WO 2019/178147, September 2019.

2. **M. Fowler** and R. Williams, "Intelligent Distribution of Data for Robotic and Autonomous Systems," U.S. Patent Application 19/021923, March 2019.

Directly related conference papers:

1. **M. Fowler**, P. Tokekar, T. C. Clancy, and R. Williams, "Constrained-action POMDPs for Multi-agent Intelligent Knowledge Distribution,", *International Conference on Robotics and Automation*, IEEE 2018.

2. **M. Fowler**, and R. Williams, "Learning Coordination Graphs through Relational Learning for Multi-agent Collaboration,", *International Conference on Intelligent Robots and Systems (IROS)*, IEEE [In Progress].

Indirectly related conference papers:

1. Shi, Yi, Kemal Davaslioglu, Yalin E. Sagduyu, William C. Headley, **Michael Fowler**, and Gilbert Green. "Deep Learning for RF Signal Classification in Unknown and Dynamic Spectrum Environments." *2019 IEEE International Symposium on Dynamic Spectrum Access Networks* (DySPAN). IEEE, 2019.

2. S. Hitefield, **M. Fowler**, and T. C. Clancy, "Exploiting buffer overflow vulnerabilities in software defined radios," in *Computer and Information Technology (CIT), 2018 IEEE International Conference on*, IEEE, 2018.

3. J. F. Ziegler and **M. Fowler**, "OFDM Precision Jamming in support of a Low-power Nvidia SoC implementation," in *Military Communications Conference*, 2016. MILCOM 2016. [Restricted, Accepted], IEEE, 2016.

4. S. D. Hitefield, Z. Leffke, **M. Fowler**, and R. W. McGwier, "System overview of the Virginia Tech ground station," in Aerospace Conference, 2016 IEEE, IEEE, 2016.

5. **M. Fowler**, A. O'Donnel, D. DePoy, and T. C. Clancy, "A Cognitive Subsystem for the optimization of Radar Doctrine for Radar/Comms Coexistence," in Military Communications Conference, 2015. MILCOM 2015., IEEE, 2015.

6. J. F. Ziegler, J. M. Ernst, and **M. Fowler**, "Blind OFDM Signal Parameter Estimation for use in efficient Jamming Techniques," in Military Communications Conference, 2015. MILCOM 2015., IEEE, 2015.

7. S. Hitefield, **M. Fowler**, C. Jennette, and T. C. Clancy, "Link hijacking through wireless exploitation of a vulnerable Software Defined Waveform," in Military Communications Conference, 2014. MILCOM 2014. IEEE, IEEE, 2014.

8. M. D. Adams, S. D. Hitefield, B. Hoy, **M. C. Fowler**, and T. C. Clancy, "Application of Cybernetics and Control Theory for a New Paradigm in Cybersecurity," *arXiv preprint arXiv:1311.0257* (2013).

Technical Reports:

1. L. Beex, D. Doyle, **M. Fowler**, et al. "Re-Imagining Network-based Communications," Final Report 17-C-0182 CLIN 0003AC, U.S. Government, January 2020.

2. **M. Fowler**, "Autonomous Model-based Systems Engineering for Cyber Risk Assessments," Quarterly Report N00421-15-R-0020, U.S. Navy Naval Air Systems Command, January 2020.

3. J. Black, **M. Fowler**, et al. "Autonomous Command and Control for Robust and Resilient Wireless Network Communications," Final Report 17-C-0182 CLIN 0003AA, U.S. Government, August 2019.

4. **M. Fowler**, "Intelligent Autonomous Cyber Testing of Avionic Systems," Quarterly Report N00421-15-R-0020, U.S. Navy Naval Air Systems Command, January 2018.

5. **M. Fowler**, T. C. Clancy, and R. McGwier, "Distributed Coordination among Tactical Electronic Warfare Platforms Phase 1: Cross-Layer Cognitive Mission Management," Final Report HR0011-15-C-0149, Defense Advanced Research Projects Agency, September 2016.

6. **M. Fowler**, T. C. Clancy, and R. McGwier, "Heterogeneous Parallel Mobile Computing for Cognitive Electronic Warfare," Final Report FA8650-11-D-1004-0005, Office of Naval Research, June 2016.

7. **M. Fowler**, "EMS monitor & broadcast training capacity enhancement - Phase 2: Analytical framework for electronic warfare," final report, Air Force Research Laboratory, April 2015.

8. **M. Fowler**, T. C. Clancy, and R. McGwier, "Spectral coexistence through Cognitive Radar program - Phase 2," Final Report N00014-12-C-0702, Office of Naval Research, March 2015.

9. **M. Fowler**, T. C. Clancy, and R. McGwier, "Spectral coexistence through Cognitive Radar program - Phase 2: Executive summary," Executive Summary N00014-12-C-0702, Office of Naval Research, March 2015.

Chapter 2

Mobile Sensor Networks

Wireless sensor networks (WSN) cover a wide range of technologies that monitor the environment that do not have hardline connectivity to a network or the internet. If those sensors are not static but are mobile sensor networks (MSN) like unmanned vehicles, then the complexity of handling their command, control, and communications becomes staggering. The information they are gathering about the environment is the most critical and therefore any bandwidth necessary for command and control needs to minimize its impact on this critical information. In this chapter, we provide the motivation behind *Intelligent Knowledge Distribution* to address the impacts command and control have on communications between mobile sensor agents, especially as it pertains to the applications, architectures, and management of wireless sensor networks (WSN) to ensure the compatibility of IKD. In autonomous systems, the management of these wireless sensor assets becomes internalized with the system and decentralized with the current management approaches providing insight into the advantages and gains provided by distributed autonomous systems.

As more devices or nodes are implemented in a system, the command and control of those assets grows exponentially requiring increasing degrees of autonomy. This autonomy, out of necessity, needs to comply with the architectures and requirements of the host systems while trying to eliminate or drastically reduce human-in-the-loop requirements, so that operators can concentrate on their roles and responsibilities of providing warfighter efforts, disaster response, emergency management, et cetera. Multi-UAV systems require autonomous flight

operation to simplify and abstract the user interface:

> Finally, most applications require some autonomy in the flight operation of the UAVs. While this may be *preferable* for single-UAV applications, autonomous flight operation is *required* for multi-UAV systems. Autonomy helps to simplify and abstract the user interface. With autonomy and an efficient user interface design, the users can focus on the overall mission and do not need to deal with individual UAVs (as we have demonstrated with our map-based user interface). Methods to achieve high levels of autonomy and low levels of user interaction are required.[3]

Random deployment of sensor nodes requires that the network protocols and *algorithms* have the capability to self-organize [37]. This also applies to sensor nodes that have self-mobility capabilities as in unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) It is this desire to reduce the strain and effort of human operators that autonomous systems become more independent as edge computing nodes in applying the appropriate intelligence in tasking and knowledge acquisition and distribution.

2.1 Applications & Field Implementations

The constant improvement in battery technologies, embedded computing, low-power communications, and sensor miniaturization has created a proliferation of sensor systems in industry, military, and home applications. Military and commercial sensors are the most relevant, since they typically utilize platforms that have more computational capability that could support the artificial intelligence (AI) behind Intelligent Knowledge Distribution (IKD).

Figure 2.1 shows a typical field implementation of sensors in a sensor field where the sensor

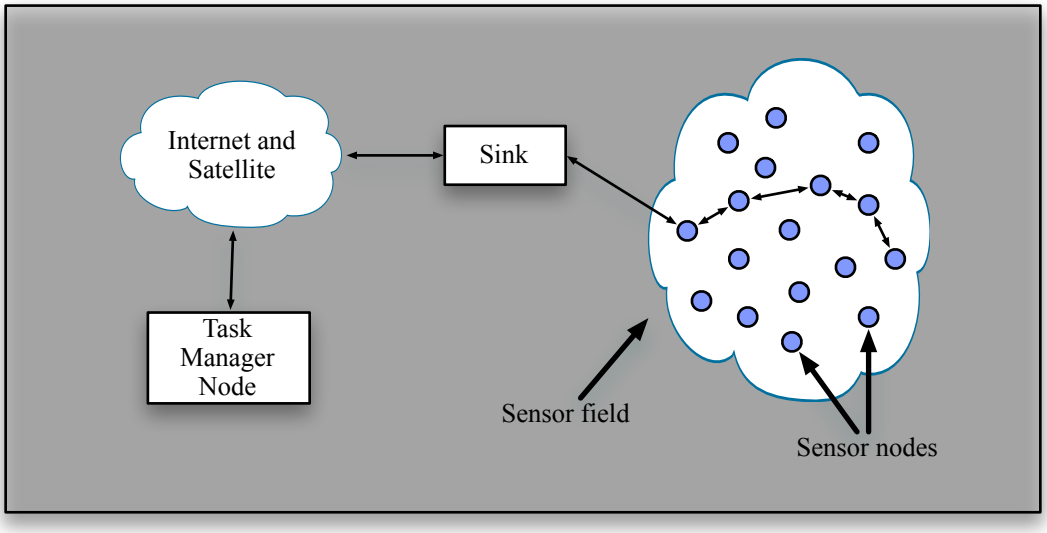


Figure 2.1: Sensor nodes deployed in the environment

field could be a battlefield area of operation or a disaster site needing emergency services. The sensors need to dynamically create a communication network so that relevant information being monitored in the field can reach the human operators to make critical decisions on the deployment of soldiers or emergency personnel. The design of a sensor network is dependent on many factors, including:[37]

- **Fault tolerance** – A failure of node should not impact the overall mission objectives of the sensor network. The fault tolerance of a sensor node is modeled as a Poisson distribution with the probability of not having a failure within the time interval $(0, t)$:

$$R_k(t) = e^{-\lambda_k t}, \tag{2.1}$$

  where λ_k is the failure rate of the sensor node k. Fault tolerance is also a driver for the importance of learning collaboration strategies, since any collaborative relationship will have to find new collaboration strategies if a key strategic partner fails.

- **Scalability** – The ability to support a growing number of sensors ranging from a few dozen to thousands depending on the application. The more collaboration or cooperation these sensors need, the more difficult it is to support scalability. Research is still ongoing to achieve proper scaleability in Mobile Ad-hoc Networks (MANET) [38], which is why intelligent knowledge distribution is necessary to reduce the quantity of communications on these networks for autonomous command and control.

- **Production costs** – The volatility of the environment or ability to recover sensors may require the production costs to be so low that they can be considered expandable by the end customer. The more expensive the platform, the less risk an end customer will take in lost or damaged devices.

- **Operating environment** – Is the sensor field large or small? Are the sensors mobile or immobile? Is the environment hostile or corrosive?

- **Sensor network topology** – There are multiple means to establish a network topology for the sensors. This can range from point to point communications directly to a hub (e.g., an LTE Tower) to full MANET. Low-power devices will often use a broadcast approach to the network topology or a hierarchical network topology. If humans or a centralized system is needed, then the topology will have to make it to a "sink" that then connects to a user interface or cloud-based system. In hub networks, IKD does not provide communication gains but still could be effective in minimizing battery usage and CA-POMDP can be utilized for any other action-based resource consuming decisions.

- **Hardware constraints** – Hardware constraints can involve sensor limitations in accuracy or area, computational limitations, memory and storage limitations, or failure rates. This will impact the capability of using complex AI algorithms due to their

memory and computational requirements.

- **Transmission media** – Transmission media specifically deals with how the communications are carried out. Wireless communication systems that utilize the RF spectrum are common, but they can also include infrared, laser, and optical techniques. Some of these transmission media can be omnidirectional, broadcasting in all directions simultaneously or require direct line-of-sight (e.g., high gain antennas and lasers).

- **Power consumption** – The availability of power or restrictions on what can be harvested from the environment limit the amount of energy the device can consume or its longevity in operation until it needs to be replaced. This is key to the efficacy of IKD as computational requirements on power are less than communication requirements.

Commercial applications can cover a wide range of scenarios from monitoring the engine of your car to the energy generation and transmission for an entire region of a country and is driving a new industry behind "Sensing as a Service" to provide cloud services for the Internet of Things [39]. Our research focuses on the application of UAVs as a sensor network, which are commonly used as relays for ad-hoc networks [40, 41], search and rescue operations [42], ground target detection and tracking [43, 44], automatic forest fire monitoring and measurement [45], disaster monitoring [46], remote sensing [47], agricultural remote sensing [47], squad support [48], etc.

2.2 System Architectures

There are two primary system architectures of interest to this research: the Internet of Things (IoT) and unmanned systems, especially unmanned aerial vehicles (UAVs). Though our primary motivation is based around the management of UAVs, there has been a lot of

research into architectures, protocols, and requirements for IoT that can be leveraged to improve the efficiency and universality of autonomous systems in a wide range of fields. There are numerous fields that the CoDec POMDP and IKD is planned to be applied to and ensuring a more agnostic approach of the framework will permit the ease of production and commercial development within a wide range of applications.

2.2.1 Internet of Things Architectures

With the proliferation of internet connected devices and estimates that there will be over 24 billion connected devices by 2020 [49] and 41.6 billion by 2025 [50], there has been extensive work in establishing resilient and robust architectures to handle their management. There are six primary elements in the Internet of Things [51]:

- **Identification** – The ability to name and identify services that are demanded from the system. Many identification methods have been proposed for IoT especially electronic product codes (EPC) and ubiquitous codes (uCode) [52]. Providing classifications and taxonomies for a system will aid in the dissemination of available services between agents and is important for IKD to understand the ontologies for learning collaboration strategies.

- **Sensing** – The collection of data from hardware and software for analysis to take specific actions in response to the information. Many architectures often refer to sending the data back to a data warehouse for the analysis, which requires extensive bandwidth limiting how much is available for command and control.

- **Communication** – Connecting heterogeneous devices to provide smart services through a variety of commonly used protocols, including WiFi, Bluetooth, Low-Rate Personal

Area networks, Z-wave, and LTE. The selection fo a network protocol and topology will determine the usefulness of IKD for that deployment. Mesh topologies provide optimal robustness and resiliency to a distributed system, which is the environment IKD is the most viable.

- **Computation** – Refers to the processing units and their supporting operating systems and libraries. Some common hardware platforms developed to run IoT applications include Arduino, Friendly ARM, Intel Galileo, Raspberry PI, and BeagleBone. Several Real-Time Operating Systems (RTOS) have been used to support IoT, especially TinyOS, LiteOS, and Riot OS [53].

- **Services** – Services can be categorized into four classes: Identity-related Services, Information Aggregation Services, Collaborative-Aware Services and Ubiquitous Services. Each services builds on the previous from identifying the object, to collecting the data, to acting on the data, to finally providing services anytime and anywhere to anyone.

- **Semantics** – Semantics deals with ability to provide knowledge extraction and allocation of services to the correct devices on demand. This same knowledge extraction is the drive behind developing an approach for coordination graphs through ontologies and semantic networks, described in Section 6.7.5.

Figure 2.2 shows the multiple generic architectures used in IoT. The first is the basic three-layer approach, but more modern approaches [51, 54] have moved onto models that provide more abstractions of the layers to provide more differentiation of services and objects. The five-layer perception architecture on the far right is the basis of the model used in this research and are divided into:

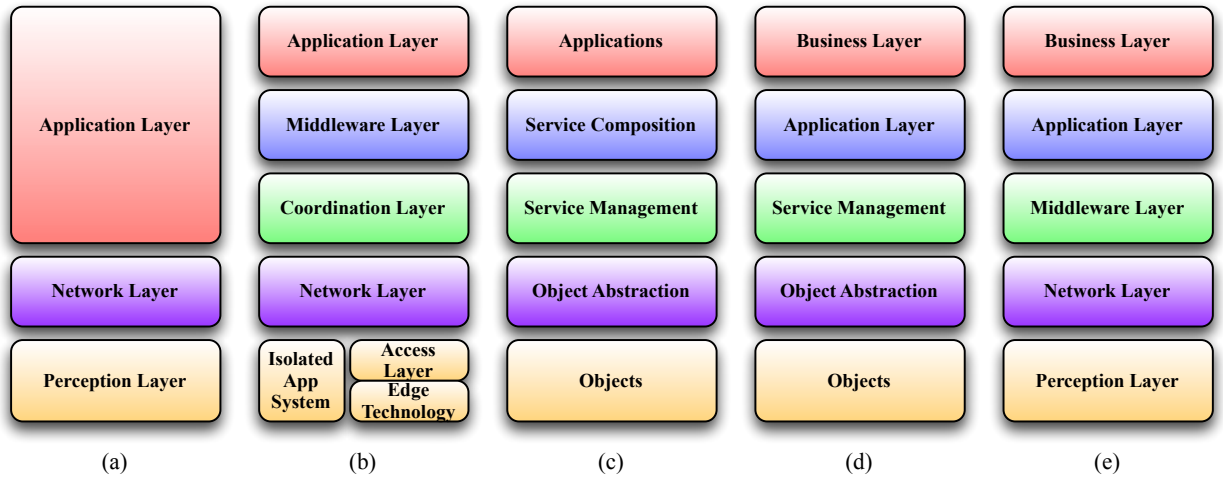


Figure 2.2: IoT generic architectures: (a) Three-layer (b) Middle-ware based (c) SOA based (d) Five-layer (e) Five Layer Perception

- **Perception Layer** – This layer gathers the information from the sensors and manages the overall device, which is sent on through the network layer to the central information processing system. Uncertainties in the accuracy of the perception layer can be used to understand the relevance of data used in IKD.

- **Network Layer** – This layer is responsible for *securely* transferring the information across the communication channels to the central information processing system. The IKD communication policy, that determines what information should be transmitted to whom and when, lives at this area and controls the communication of command and control information.

- **Middleware Layer** – This layer is a service-orientated layer which manages and stores the information for retrieval and processing. The middleware also provides information for the IKD to learn collaboration strategies being understanding the services and capabilities available to work with other agents towards mission objectives.

- **Application Layer** – This layer is for inclusive application of the data available from

the middleware layer and has a commonly understood mission (e.g., smart home).

- **Business Layer** – This layer provides a holistic understanding of the application and services for executive level decisions about the business. A holistic understanding is critical to mission objectives and understanding those is key to determining the impact the lack of communication will have on meeting those objectives.

Software-defined networking, which is trying to revolutionize tradition networks, is also being used to address the potential overload of the internet through more intelligent networks [55], but this doesn't address that there is still the same amount of information trying to traverse the internet. It is believed that as devices produce more information from internet connected devices or deployed wireless sensor networks, that this could leave the network paralysed [49]. Therefore, we need to address how we manage *what* information flows and not just how.

2.2.2 UAV Architectures

Unmanned Aerial Vehicles (UAVs) come in a wide range of configurations and capabilities that need to be addressed to ensure that the correct asset is being selected for the task or mission. Figure 2.3 shows three typical UAVs that are available commercially from public companies. Depending on their design, they might be very efficient at hovering and maintaining a stable position over a point, as you would desire for an active rescue or for maintaining communication links. Faster speeds or maneuverability in tight spaces might be needed, similar to the design of a racing drone, which are typically needed with surveillance drones. At other times, the desire is to have an asset in the area for an extended period of time, therefore an aerodynamic design that is more efficient, as with fixed wing aircraft, would be suitable.

The Department of Defense has established a classification system, shown in Table 2.1, that

(a) A DJI Mavic Pro quadrotor UAV with great hovering efficiency but lacks speed or endurance

(b) A Storm Racing Drone quadrotor UAV with great speed but lacks efficient hovering or endurance

(c) An AeroVironment Mantis fixed wing UAV with great speed and endurance but inefficient at station keeping compared to a multi-rotor UAV.

Figure 2.3: Comparison of UAV designs and their abilities to maintain surveillance.

breaks down UAV assets into their size, weight, operating altitude, and airspeed. Within the DoD and other agencies, the classification of asset is often also an indicator as what level of the organization owns such an asset. Therefore, these classifications may also have an impact in cross-organization interoperability.

Table 2.1: UAV Classification according to the US Department of Defense (DoD) [9]

Category	**Size**	**Maximum Gross Takeoff Weight (lbs)**	**Normal Operating Altitude (ft)**	**Airspeed (knots)**
Group 1	Small	0-20	<1,200 AGL	<100
Group 2	Medium	21-55	<3,500	<250
Group 3	Large	<1320	<18,000 MSL	<250
Group 4	Larger	>1320	<18,000 MSL	Any
Group 5	Largest	>1320	>18,000	Any

These aren't the only characteristics being considered by operators in the field. They are also interested in the range and endurance of the UAV to address the distance a UAV must cover to perform its task, e.g. power-line inspection or surveillance. Table 2.2 lists some of the common range and endurance categories used in the industry. The more a system provides information about the capabilities of a system, the more intelligent and relevant the collaboration strategies become for IKD.

Of interest to autonomous drones and this paper, collaborative drones need to understands

Table 2.2: UAV Range and Endurance Classifications

Category	**Range**	**Endurance**
Low cost close-range	5 km	20-45 min
Close-range	50 km	1-6 hrs
Short-range	150 km	8-12 hrs
Mid-range	650 km	12+ hrs
Endurance	$\geq$ 300 km	36 hrs

each-others capabilities when working together. Figure 2.4 begins to address developing a taxonomy that classifies drones into the services [2] they provide. Interestingly it also takes into account how a human operator interacts with the agents themselves and even how system should provide security and privacy.

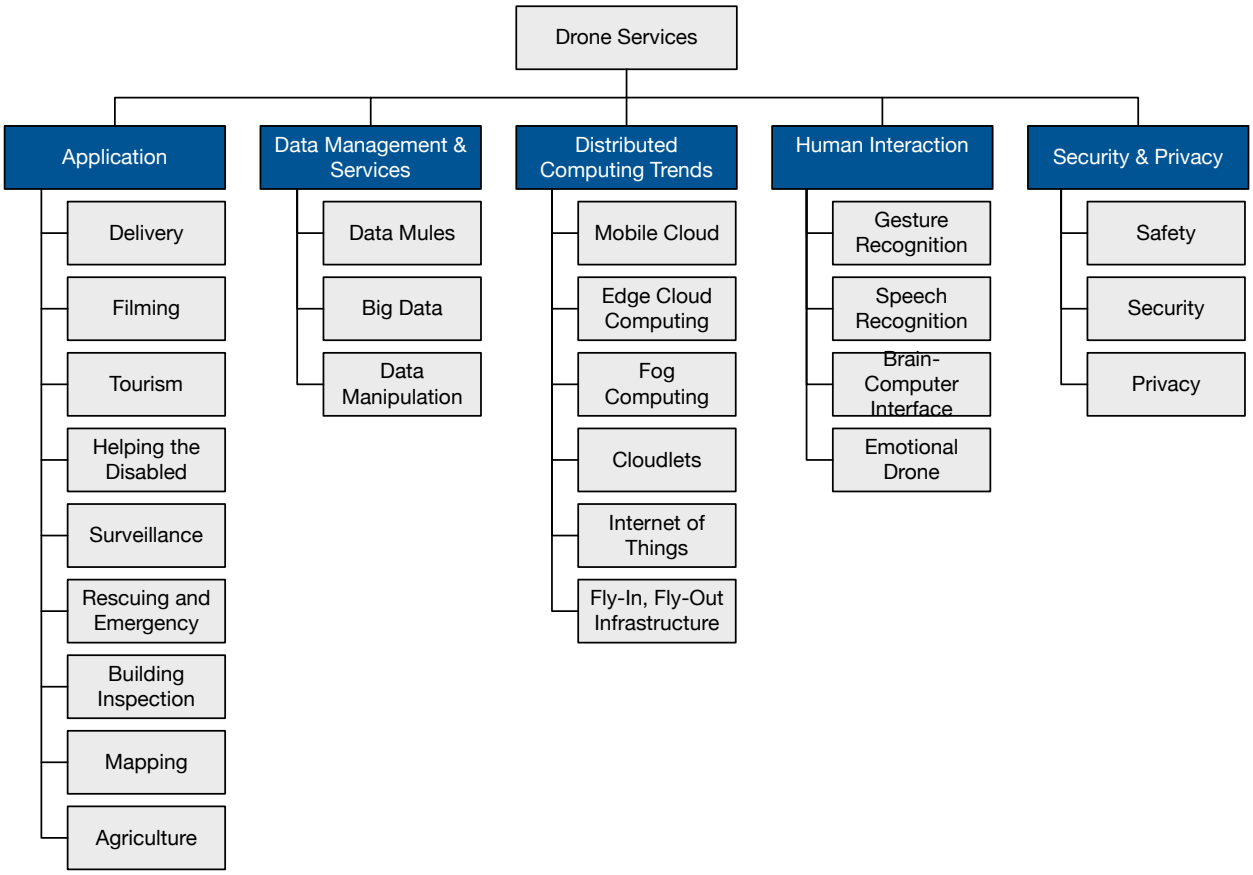


Figure 2.4: Taxonomy of Drone Services [2]

The development of architectures to address Unmanned Aerial Vehicles (UAVs) has arisen

from the necessity to handle multi-UAV operations and swarm control. UAVs provide a unique set of problems that requires significant processing capabilities to address some of the most common issues including obstacle avoidance, high-bandwidth video, and coordination between drones.

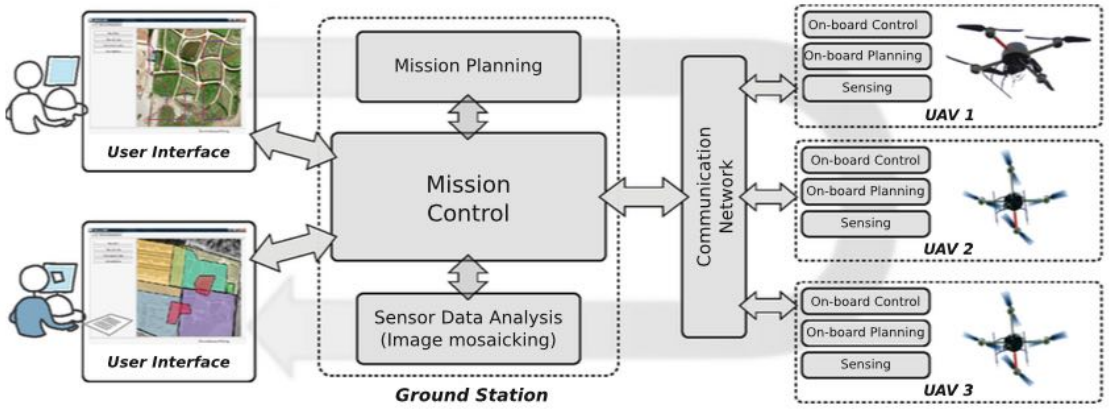


Figure 2.5: System architecture: double-headed arrows indicate interactions between individual modules while the shaded arrow in the background indicates the basic operation flow [3].

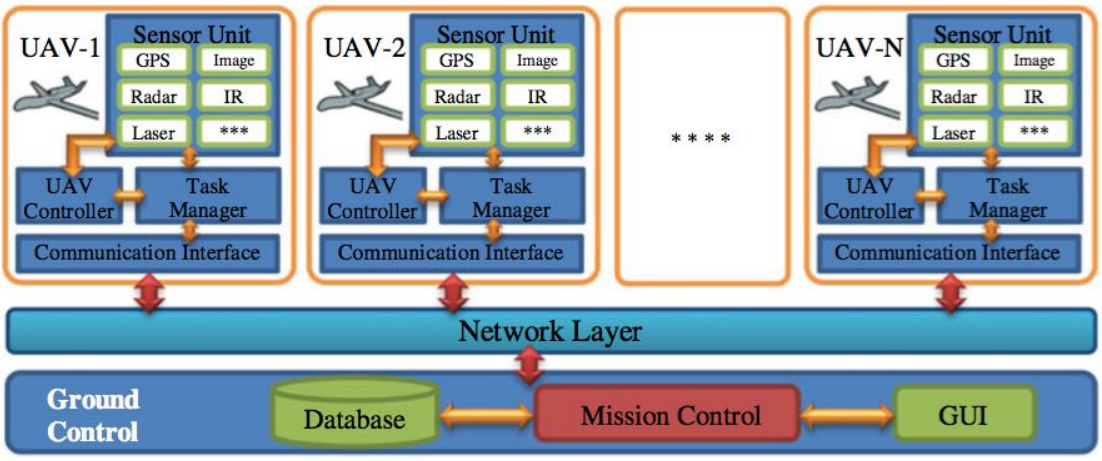


Figure 2.6: A multi-UAV system's processing units and communication/network layer [4]

Figure 2.5 and 2.5 shows two approaches that were developed separately to address multi-UAV control. Though developed separately, they have a large number of common components in their architecture. Of interest in this work is specifically the *On-board Planning*

in Figure 2.5 or *Task Manager* in Figure 2.5 and the *Mission Control* shown in both. Mission Control contains most of the processing and computational load that is managing the "autonomy" of the multi-UAV system. This can led to many great demonstrations of multi-UAV capabilities as shown during the opening of the 2018 Winter Olympic Games which controlled 1,200 drones in a dramatic light show. Like cloud computing, these multi-UAV architectures depend heavily on communication links back to a central service or manager that controls the overall mission of the system.

Centralized command and control of multi-UAV systems can not be assumed in heterogeneous autonomous systems managed by different agencies which might all react to a scenario, where communications are not reliable, or when resilient systems should not depend on centralized control. All these systems manage coordination between drones and our system also needs to address distributed Task Allocation and Motion Planning [3].

2.3 Management & Data Architectures

The previous section covered the service level architectures and layer abstractions used to handle unmanned aerial vehicles and the internet of things. In this section, we look at how critical communications are handled and how they handle the distribution or routing of information in highly dynamic, mobile systems. The protocols and techniques in controlling the network topology need to support and understand the coordination between autonomous assets as to allow the application of IKD and minimize their impact.

2.3.1 Wireless Architectures & Protocols

As wireless networks grow and the demand increases, the original concept of single-hop wireless networks is ever becoming difficult to implement with the expense of installing an infrastructure, obtaining spectrum, et cetera [56]. In a tactical environment, the infrastructure-based wireless network doesn't allow for the flexibility of mobility, varying environment, and the limited access to an access point (AP). Therefore, multi-hop wireless networks, a.k.a. wireless *ad hoc* and mesh networks, are seen as necessary for the future as was the advent of *bridging* on Ethernet (802.3) for wired networks [56].

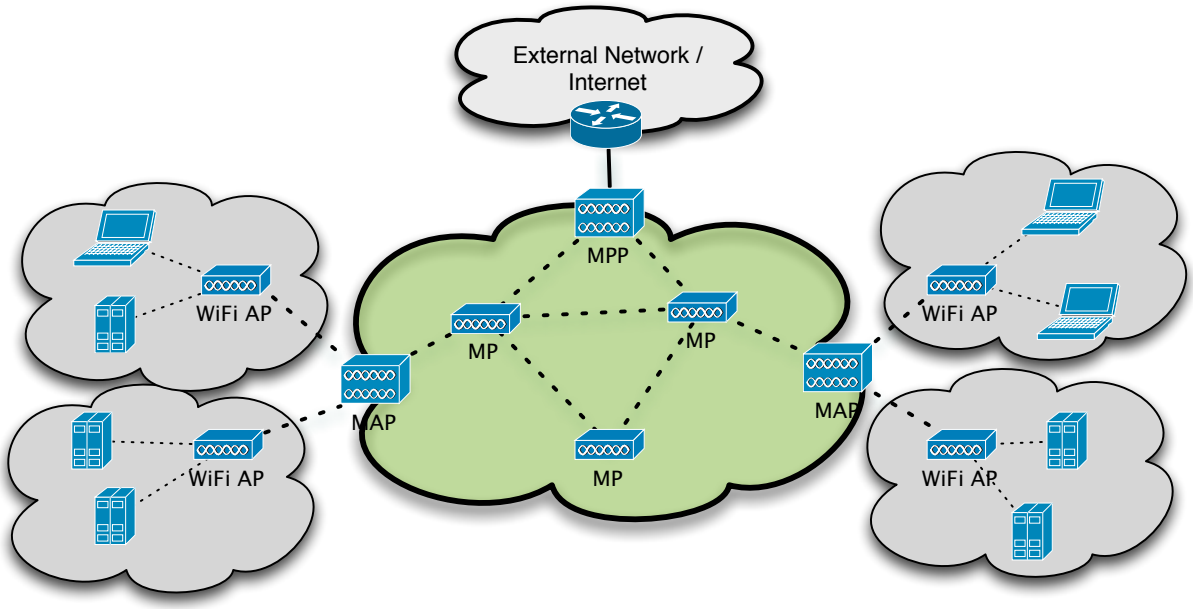


Figure 2.7: WLAN Mesh Network

A wireless *ad hoc* network is a type of wireless network that is decentralized and does not depend on any infrastructure access point to provide broadband network communications or other services, and where all the nodes associate with each other are of equal importance, which can often take on a mesh topology. Figure 2.7 illustrates the topology of a wireless mesh network, where the mesh nodes associate with each other (center; numbers reduced for

clarity), and certain nodes can have additional features or modes to allow for interoperability with traditional WiFi clients (MAP) or an infrastructure access point (MPP) to the internet [37].

Mobile Ad-Hoc Networks (MANET) were developed to handle dynamic network topologies that don't necessarily require providing all the information back to a "sink" for dissemination but might have a significant amount of machine-to-machine (M2M) communication. Fully autonomous systems place a high demand on M2M communications and therefore a topology that is optimized for the intelligent communications, IKD, are necessary to minimize the impact of command and control on the systems primary purpose. Though MANETs were specifically designed to handle dynamic topologies, the difficulties of highly dynamic networks brought about adaptations to handle vehicles, Vehicle Ad-Hoc Networks (VANET).

The VANET builds on the MANET by providing a self organizing network that can handle the highly mobility of vehicles [57] within the car and between the car and other cars, roadside side infrastructure, and cloud infrastructure. As the concept of MANETs was being applied to multi-UAV systems, the mobility and latency requirements of UAVs [58] led to research in Flying Ad-Hoc Networks (FANET), also known as Unmanned Aeronautical Ad-hoc Networks (UAANET), which are built on VANETs.

IEEE 802.11 Task Group S has been, since 2004, working on standards to address multi-hop communications using the widely accepted and inexpensive 802.11 (WiFi) standards with protocols to address wireless routing and security for *ad hoc* and mesh networks [56]. The 802.11s standard requires Hybrid Wireless Mesh Protocol (HWMP) [59] for routing, but allows for *ad hoc* or dynamic link state protocols (OLSR, B.A.T.M.A.N.) or static routing (WDS, OSPF). HWMP is based on AODV and tree based routing, but uses MAC addresses for routing, instead of IP addresses, which is why it is often referred to as a path selection rather than route selection protocol.

2.3.2 Software-defined Wireless Sensor Networks

Software-defined Networking (SDN) disassociates the forwarding process of network packets (data plane) from the routing process (control plane), but the centrality of the approach can come at the cost of security, scalability and elasticity. Some companies are starting to add proprietary techniques like Cisco's Open Network Environment and Nicira's network virtualization platform. It is a flow-based forwarding scheme, where multiple header fields determine how the incoming packet will be handled and all network devices recording and report traffic statistics to the controller. The SDN was developed to provide a paradigm shift of traffic pattern from a server-client model to the modern internet of Machine-to-Machine (M2M), Bring Your Own Device (BYOD, both personal and corporate), cloud services, big data and its associated bandwidth requirements. Software-defined networking started at Stanford University and culminated in the OpenFlow protocol [60] and in 2014 was being used with OpenStack [61], an open source software for creating private and public clouds by controlling large pools of compute, storage, and networking resources [62]. In an effort to move from our current static network architectures to support flexibility and manageability, the objectives of an SDN architecture consists of:

- **Directly programmable**: The forwarding functions are decoupled from the network control allowing direct programmability for improved network-wide performance;
- **Agile**: By abstracting control from forwarding plane, administrators or autonomous algorithms can dynamically adjust network-wide traffic flow to meet changing needs;
- **Centrally managed**: Network intelligence is *logically* centralized in software-based SDN controllers which are capable of maintaining a global view of the network, which appears to applications and policy engines as a single, logical switch;
- **Programmatically configured**: SDN should allow network managers configure,

manage, secure, and optimize network resources via rapid, dynamic, and automated software, which does not depend on proprietary software; and,

- **Open standards-based and vendor-neutral**: All SDN devices should permit network design and operation through open standards, instead of multiple, vendor-specific devices and protocols.

As mentioned previously, an SDN architecture separates the data plane from the control plane, which requires a set of components to provide the command and control of the network. Therefore, an SDN architecture is typically divided into the following basic components:

- The **Application or Management Plane** communicates their network requirements and desired network behavior via a *northbound interface*;
- The Controller or **Control Plane** translate quality of service and service level agreement requirements to properly manage the network to maintain those requirements based upon information statistics provided by the network devices;
- The Data Path or **Data Plane** is a simple rule-based response based upon header information along with statistical information like the pattern field, counter field, and action field, which can include modification, dropping, and/or reporting;
- The **Southbound Interface** is an API between a controller and a network switch that is encrypted and authenticated for the controller to manage the network; and,
- The **Northbound Interface** is an API between a network switch and the controller that is encrypted and authenticated for the switch device to report statistics, request new rules, et cetera.

Wireless Software-defined networking combine the capabilities and benefits of software-defined networking (SDN) and software-defined radios (SDR) [63]. A lot of issues arise

with the introduction of centralized management and the APIs to support the open network configuration, including device hijacking and data theft [64]. In wireless networks, there is open research into combining physical security, including isolating the control and data plane in different frequency bands, to mitigate some of these risks in open air communications

In the other end of wireless communications, software-defined networks are also being utilized to address issues in cyber physical systems (CPS), especially around guaranteeing reliable real-time low latency and reducing the overhead impact of control communications and network switch statistics. Controller-based CPSs allow self-reconfigurable CPS with a closed-loop feedback mechanism based on the monitor, analyze, plan, and execute (MAPE) process. The CPS Network Manager than can allocate appropriately for each time-sensitive application and optimize transmission scheduling on the CPS nodes. NIST's CPS Public Working Group is adopting the SDN paradigm to dynamically manage CPS Networks.

One of the critical discussions in trusting SDN in CPS networks is ensuring safety critical operations with real-time and time-sensitive SDN [65]. Avionic systems analyzed for achieving deterministic latencies with a software-defined AFDX using a bootstrap controller. Then, optimized the control channel to reduce the number of out-of-band messages between switches and multiple OpenFlow controllers. Real time CPS in safety critical CPS (SAE AS6802), with static pre-configuration, was able to guarantee bounded delays with a centralized scheduler to compute and allocate time slots. Current trends have sensors (publishers) and actuators (subscribers) dynamically plugged where OpenFlow-based control receives subscription service requests with the QoS specifications

Though SDN can balance the topology to maximize the efficiency of information being transmitted between the agents and the energy consumed in relaying packets, it does not address the initial consumption of resources by an agent to transmit information to another agent. IKD addresses that initial consumption and could cooperatively work with an SDN controller

to optimize the network by providing feedback on the current collaboration strategies to minimize impacts on the topology and to quickly reorganize the topology as those collaboration strategies change.

2.3.3 Named Data Networking

Named data networking (NDN) is founded on the concept of Content Centric Networking (CCN) [66] where information flows through the network based on the content of the information itself. NDN also focuses heavily on network security, specifically routing robustness, data integrity, and traffic management. It is the information centric approach that is of interest to future research in IKD, where agents can learn the information that is most relevant to them and then have the full capability of sharing that interest with the network to optimize the flow of information and minimize overall power and communication resources in the system.

Named data networking operates fundamentally differently from traditional TCP/IP. Instead of host-destination addressing and inquiries, a recipient specifies the name of the data they want, and the network routes the request to the appropriate data provider [67]. NDN is similar to the "thin waist" model of Figure 2.8 in traditional TCP/IP network stacks. In a TCP/IP network, the fundamental protocol relies on a single packet structure which lies between more advanced protocols eg. TCP, HTTP, and near-metal protocols. NDN does not use packet encapsulation, opting instead for named data chunks at the waist. This allows for simple in-network caching schemes and additional intrinsic security features.

We will use the example of a consumer which requires data from a sensor/producer somewhere within the network. In NDN, the consumer would not need to know anything about the producer and can retrieve the requisite information using only the name of the data

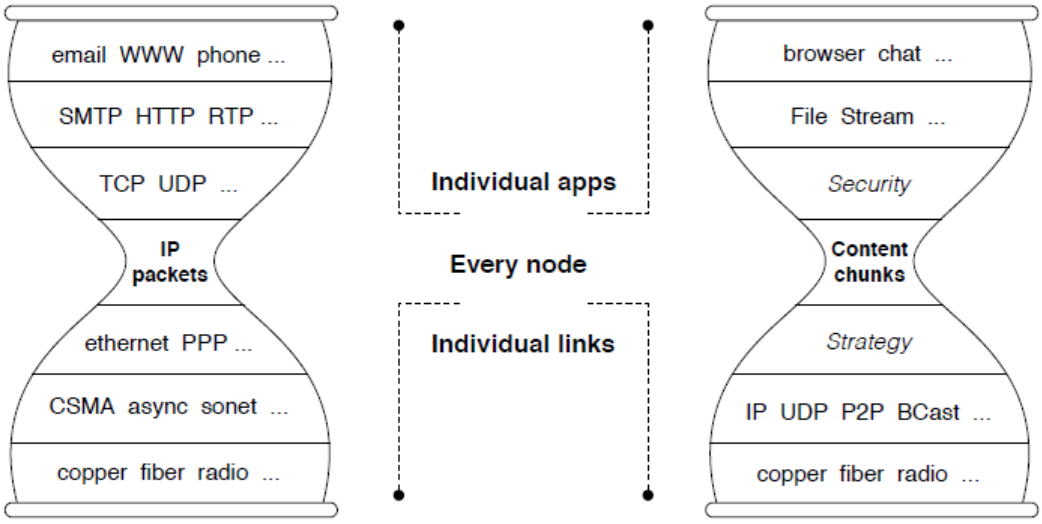


Figure 2.8: The Thin Waist Model [5]

itself, eg. /sensors/solar/light_level. The request for information is sent as a signed and timestamped "interest packet" (IP). IP's are forwarded through the switches in the network until they reach the data producer, which returns a signed "data-packet" (DP).

Switches in an NDN network operate using a stateful forwarding plane, meaning they keep information on the current state of their ports and flows. Every switch contains three main components: the content store (CS), the pending interest table (PIT), and the forwarding information base (FIB). The CS is used to cache DP's through the network thereby increasing data delivery robustness and reducing latency. Each switch also has a PIT which remembers the entry point for an IP and the IP itself. The PIT can then be used to deliver DP's using a "bread crumb" scheme where each DP is forwarded backwards following the corresponding IP's PIT entries. To determine which "Face" to send the IP or DP to, a switch consults the FIB which manages traffic control and routing strategies.

It is also useful to consider the case where the nodes are mobile in the network. In a traditional IP network, if the sensor were moved to a different sub-network (eg. now hard-wired

instead of wireless), the controllers would need to determine its network address once again. In the case of a building sensor, mobility isn't a major issue, so consider instead a tracking device carried by the mail cart. It may be pertinent for building management to keep track of the cart at all times; however, it is rapidly moving between network hosts and potentially being assigned new addresses. In an NDN system a controller would simply request the information it desires, eg. "/building/assets/mailcart/position/1205" and the network will find and deliver the information. In a traditional IP network, the controller would first need to discover the network address and then send a request, potentially through several network address translation (NAT) layers. Is is apparent then that NDN offers simplified networking in a mobile system [68][69] However, there are still several issues which complicate the final mobile network implementation of NDN, including producer mobility and rehoming mid-transmission [70], but some methods have already been proposed to aleviate some of these issues [71][72].

Chapter 3

Sequential Decision Making under Uncertainty

Autonomous agents utilize artificial intelligence in the field to make independent rational decisions towards achieving a goal in the presence of uncertainty from both the environment and the agent itself. Decision-making algorithms rely heavily on algorithm approaches based on maximizing expected utility. Many of these technique take a myopic approach and fail to address situations where actions may not be immediately rewarded or achieve a goal. Instead a sequence of actions, or policy, needs to be computed that takes into account a long-term perspective on achieving goals. In this chapter, we discuss the foundation behind sequential decision making and how these models are adapted to handle uncertainty in the environment, the states, or the model itself.

3.1 Markov Decision Processes

It is often necessary to determine an optimal set of actions where the rewards are delayed or realized at some future point in time and the agent needs to determine a set of actions, or policy (π), that achieves this goal. Due to the uncertainty of the environment or the agent itself, a probabilistic or stochastic processed needs to be defined that a set of actions needs to be defined. To reduce the complexity of attempting to maintain a complete action and

observation history, a Markov process can be assumed. By adding actions into a Markov process model, we obtain a Markov decision process (MDP) with a graphical representation shown in Figure 3.1..

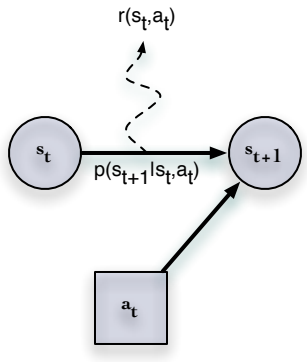


Figure 3.1: Markov Decision Process

3.1.1 MDP Formulation

Markov decision processes (MDP) are a decision theoretic approach to determining the best actions over a time horizon based upon the current state of the system (s), the probability of transition to another state ($T(s'|s)$), and the rewards associated with a state and action ($R(s)$) [17, 73, 74].

Due to the long-term goal utilized in MDPs, a state may be visited many times due to a failure to achieve objectives. This possibility of failure is due to there being uncertainty in the agent's own actions or the environment, by *chance* and not due to adversarial behaviors in the environment, because the model accounts for probabilistic transitions, and consists of the tuple

$$< \boldsymbol{S}, \boldsymbol{A}, \boldsymbol{T}, \boldsymbol{R}, \boldsymbol{\Pi} >$$

where $\boldsymbol{S}$ is the set of states an agent can be in, $\boldsymbol{A}$ is the set of actions the agent can take, $\boldsymbol{T} : S \times A \times S \times D \rightarrow [0, 1]$ is the set of transition probabilities ($T(s'|s, a)$) between states based on an action, $\boldsymbol{R} : S \times A \times S \times D \rightarrow \Re$ is the set of rewards ($R(s, a)$) for taking an action in a state, and $\boldsymbol{\Pi}$ is the set of policies, π, that are feasible consisting of a set of vectors of actions, $\{\pi_0, \pi_1, \ldots, \pi_t\}$. The objective is to select a policy, π, that maximizes the expected utility over time

$$\pi^*(s) = \arg\max_{\pi} U^{\pi}(s). \tag{3.1}$$

The assumption of an MDP is the uncertainty is from agent's actions or the environment and therefore the state space is fully observable and the sensors are perfect. When this is not the case, the problem becomes more complex but is solvable as shown later in the chapter. What MDPs are perfectly formulated for is reinforcement learning, where the rewards and transitions of actions can be learned online and the models adapted as necessary over the course of time.

3.1.2 Solving MDPs: Value vs. Policy Iteration

Optimal Solutions will depend on the performance criterion ($E(\sum r|s_0)$), See Table 3.1 with variants for handling rewards vs. costs or even delayed reward (e.g., Blackjack). Solutions for what actions to take at any specified state, or optimal policy, π^*, can be solved over an finite, infinite, or indefinite horizon. A finite policy is calculated over a specified finite amount of time or epochs, whereas an infinite-horizon policy does not have a finite time period or end state. The indefinite horizon is a special case where there is no specified time to complete the objectives but there is a desired goal or end state and it is often formulated using the infinite-horizon discounted reward criterion. Policies can be either deterministic of

an action per state, $a(s)$, a decision tree with actions at each point in time $a(s,t)$, or a finite state controller. Finite and indefinite horizon rewards allows for the policy to become more aggressive towards actions that have higher immediate rewards as it approaches the horizon, since higher long-term rewards are less likely to be realized in a short period of time.

Table 3.1: Performance Criterion for Rewards in Markov Decision Processes

Criterion		Equation
Finite Horizon	$\lvert\mathcal{D}\rvert = T_{max} < \infty$	$E[r_0 + r_1 + r_2 + \cdots + r_{N-1} \lvert s_0]$
Infinite-Horizon Discounted	$0 \leq y < 1$	$E[r_0 + yr_1 + y^2 r_2 + \cdots + y^t r_t + \ldots \lvert s_0]$
Total		$E[r_0 + r_1 + r_2 + \cdots + r_t + \ldots \lvert s_0]$
Average		$\lim_{n\to\infty} \frac{1}{n} E[r_0 + r_1 + r_2 + \cdots + r_{n-1} \lvert s_0]$

There are two dynamic programming formulations used to solve MDPS: value and policy iteration. Value iteration continued to update a bellman backup until the bellman residual between the value function of the current iteration, π_t, is within a bellman residual of the previous iteration, π_{t-1}. Whereas policy iteration will converge one the policy from the current iteration, π_t, is within a bellman residual of the previous iteration, π_{t-1}.

They seem similar but there is a major difference. During each iteration, the value function could be continuously updating for a large number of iterations as rewards are dispersed through the system even though the policy has not changed since the last iteration. This tends to have policy iteration converge quicker than value iteration, but policy iteration has some drawbacks that particularly for motion planning has roboticists using value iteration. If a critical or dominant reward is large number of actions and/or states from others, then policy iteration *could* terminate prematurely under certain conditions while value iteration will find the optimal value. Unless specified otherwise, the algorithm of Figure 3.2 was utilized for policy iteration including an iterative algorithm for evaluating the current policy

procedure PolicyIteration($\boldsymbol{A}, \boldsymbol{S}, \boldsymbol{T}, \boldsymbol{R}, \gamma, \epsilon$)
  $k \leftarrow 0$
  repeat
    $U^{\pi_k} \leftarrow$ PolicyEvalution($\boldsymbol{S}, \boldsymbol{T}, \boldsymbol{R}, \pi, \lambda, \epsilon$)
    $\pi_{k+1}(s) = \arg\max_a R(s,a) + \gamma \sum_{s'} T(s'|s,a) U^{\pi_k}(s')$
    $k \leftarrow k + 1$
  until $\pi_k = \pi_{k-1}$
  return π_k
end procedure
procedure PolicyEvaluation($\boldsymbol{S}, \boldsymbol{T}, \boldsymbol{R}, \pi, \gamma, \epsilon$)
  $U_0^{\pi}(s) \leftarrow 0 \quad \forall s$
  repeat
    $U_t(s) \leftarrow R(s, \pi(s)) + \gamma \sum_{s'} T(s'|s, \pi(s)) U_{t-1}^{\pi}(s')$
  until $\|V_t - V_{t-1}\| < \epsilon \frac{1-\lambda}{\lambda}$
  return U_t
end procedure

Figure 3.2: MDP Policy Iteration Algorithm

3.1.3 Partially Observable Markov Decision Processes

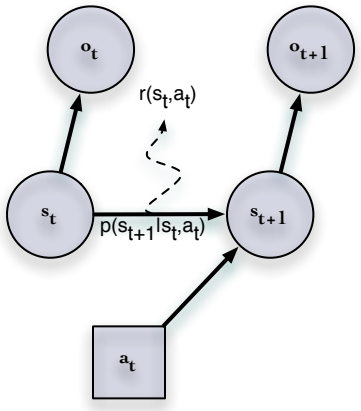


Figure 3.3: Partially Observable Markov Decision Process

If the state of the agent is not fully observable then the system becomes a Partially Observable Markov Decision Process (POMDP) where belief states, $b \in \mathbf{B}$, are used to represent the probability distribution of being in any particular state based upon observations [6, 73]. The graphical representation of a POMDP looks like a cross between an MDP and Hidden

Markov Model (HMM) as shown in Figure 3.3, where an action a_t while in a state s_t causes a transition to another state s_{t+1} but only the observation o_{t+1} is known and the reward $r(s_t, a_t)$ is realized. These belief states are tied to an observation model, O, that maps the probability of being in a state based upon what was observed.

$$< \mathbf{S}, \mathbf{O}, \mathbf{B}, \mathbf{A}, \mathbf{T}, \mathbf{R}, \ >$$

The transition probabilities are similar to those in the discrete MDP but are expanded in

$$\begin{aligned} \tau(b'|b,a) &= P(b'|b,a) \\ &= \sum_{o} P(b'|b,a,o) \sum_{s'} O(o|s') \sum_{s} T(s'|s,a) b(s), \end{aligned} \tag{3.2}$$

to account for the belief states and observation model, where $P(b'|b,a,o)$ is the Kronecker delta function, δ, to update the belief [6]. The immediate reward is simply a summation of the product of the reward for each state by the belief that the agent is in that state.

$$R(b,a) = \sum_{s} R(s,a) b(s) \tag{3.3}$$

Solving for the optimal policy, π^*, in a POMDP is also capable with dynamic programming by treating the belief states as continuous-state MDP but over a finite horizon [17]. Value iteration algorithms have been used to calculate an infinite-horizon policy but they are not guaranteed to converge [17] and therefore policy iteration approaches are utilized in the CA-POMDP, covered in Section 3.1.4.

1: **procedure** PolicyIteration(π_0, ϵ)
2: $t \leftarrow 0$
3: **repeat**
4: $\pi_{t+1} \leftarrow$ ExhaustiveBackup(π_t)
5: $V^{\pi_{t+1}} \leftarrow$ PolicyEvalution(π_{t+1})
6: **repeat**
7: $\hat{\pi}_{t+1} \leftarrow \pi_{t+1}$
8: **for all** $i \in I$ **do**
9: $\hat{\pi}_{t+1} \leftarrow$ Prune($\hat{\pi}_{t+1}, i$)
10: UpdateController($\hat{\pi}_{t+1}, i$)
11: $V^{\hat{\pi}_{t+1}} \leftarrow$ PolicyEvalution($\hat{\pi}_{t+1}$)
12: **end for**
13: **until** $|\pi_t| = |\hat{\pi}_t|$
14: $t \leftarrow t + 1$
15: **until** $\frac{\gamma^{t+1}|R_{max}|}{1-\gamma} \leq \epsilon$
16: **return** π_t
17: **end procedure**

Figure 3.4: POMDP Policy Iteration Algorithm [6]

3.1.4 POMDP Policy Iteration

The policy iteration algorithm of Hansen in [27] is the basis for improving the finite state controller which uses simple transformations that search within the policy space until epsilon convergence is observed [27]. Hansen was selected for its basic implementation to validate the concept and there are newer methods that will be considered (e.g., point-based [75]) for improving the final CA-POMDP and IKD libraries. The function starts by initializing the policy for a finites state controller δ and then evaluates the value function of a policy through a system of linear equations. A dynamic programming update is used to generate a new set of candidate machine states that the policy improvement loop uses to compare the vectors of V and V' and modifies the FSC accordingly. Any change to the finite state controller (FSC) for a link $l(i, z)$ from a machine state i due to an observation z triggers a recalculation of the policy. The algorithm terminates when the Bellman residual between the two value functions V and V' has converged within epsilon accuracy.

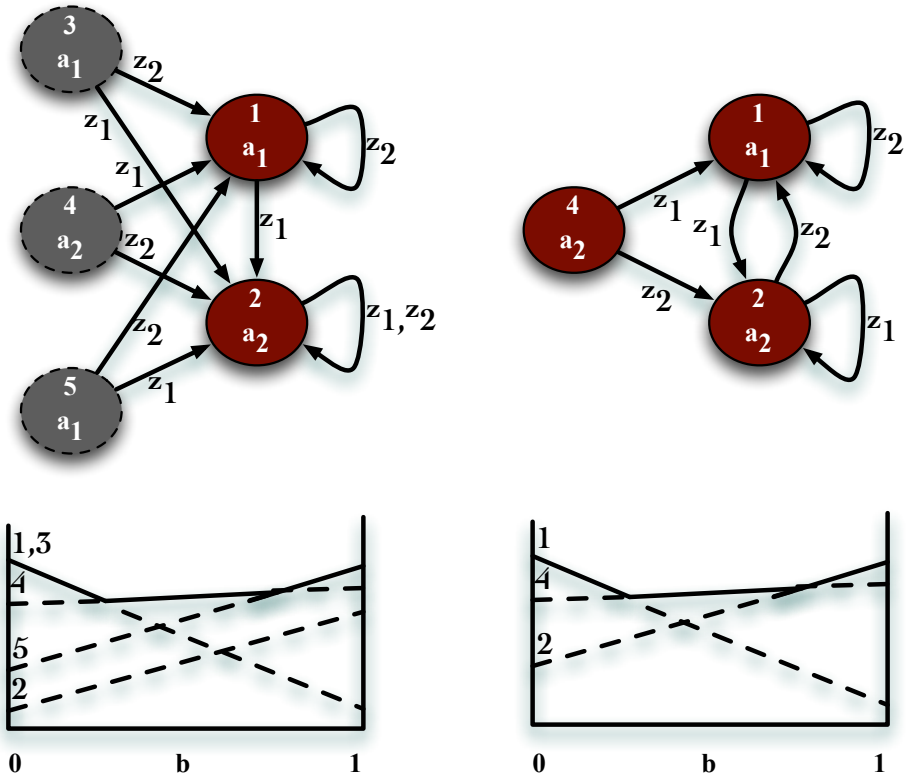


Figure 3.5: The new vectors 3, 4, and 5 from the dynamic programming update are connected to the original controller δ. 3 and 1 are matches for action, values, and transitions and therefore 1 is kept in δ'. 5 point-wise dominates 2 and therefor 2 inherits the parameters of 5.

Evaluating the policy of a finite state controller requires solving a system of linear equations [17] formulated per

$$v^i(s) = r(s, a(i)) + \gamma \sum_{s'z} P(s'|s, a) P(z|s', a) V^{l(i,z)}(s'), \tag{3.4}$$

where, for each machine state in the controller and state in the model, the value of a particular machine state i is the reward for taking the action dictated by the machine state plus the

discounted summation of the rewards of transitions to another state $l(i, z)$, based upon receiving the observation, z. The dynamic programming update uses

$$\Theta_n^{a_i,o_i}(b,s) = \frac{r(s,a_i)}{|\Omega|} + \gamma \sum_{s' \in S} p(s,a_i,s') O(s',o_i) \Theta_{n-1}^{a_i,o_i}(b^{a_i,o_i},s) \tag{3.5}$$

$$\Theta_n^{a_i}(b) = \sum_{o \in \Omega} \Theta_n^{a_i,o}(b) \tag{3.6}$$

$$\Theta_n(b) = \max_{a \in \mathbf{A}} \Theta_n^a(b) \tag{3.7}$$

to derive a new set of vectors for consideration from each vector (machine state) in the current policy, δ, with the current machine states 1 and 2 shown in Figure 3.5. For each vector in the existing policy, the algorithm determines the resulting value vector from each action proceeding after the current action. The fraction $\frac{r(s,a_i)}{|\Omega|}$ in (3.5) averages the reward across observations so that the final sum of $\Theta_n^{a_i}(b)$ across all observations, $o \in \Omega$, is accurate.

This new set of vectors is run through a linear programming (LP) formulation (3.8) to prune the set of vectors to just the dominant vectors, which are represented by the new machine states 3, 4, and 5 in Figure 3.5. Vectors that are dominated by another will not realize the best action for any region in the belief space and therefore needs to be removed. It introduces Δ to the objective function of the LP with the value coefficients for the value factors set to zero. If it finds a value of $\Delta > 0$ or the formulation fails to find a solution, then that vector is dominated by another and is removed from the set [17].

$$\begin{aligned} Objective: &\quad \max \Delta \\ Subject\ to: &\quad x(\theta - \bar{\theta}) \geq \Delta, \quad \forall \bar{\theta} \in\ , \theta \neq \bar{\theta} \end{aligned} \tag{3.8}$$

The algorithm makes two checks between the new controller V' and the previous controller V. The first determines if there are any existing vectors in V that have the same action and observation edge transitions, $l(a, z)$, as a vector in V' and places the previous vector into the new controller δ' as it was done with 1 and 3 in Figure 3.5. If there is no match, then it determines if there is any vector i in V that is point-wise dominated ($V_i(a, s) > V'_j(a, s); \quad \forall a \in \boldsymbol{A}; s \in \boldsymbol{S}$) by a vector j in V' and places the previous vector into the new controller but with the actions and transitions of the new vector in V' as 5 point-wise dominated 2 in the graph. The final step in improving the controller requires pruning the new controller δ', which removes vectors $i \in \delta$ from the controller δ' that have no correlating vector in V' if they are not reachable from a machine state in V'.

3.2 Reducing Model Complexity

In this section, we cover two approaches used to reduce the complexity of solving large state models: Hierarchical and Concurrent. They provide computational efficiencies in solving policies of large complex models, but at a cost of memory, overhead, and possibly optimality. Despite these drawbacks, their usefulness in real-time processing can more than compensate for these impacts.

3.2.1 Hierarchical Markov Decision Processes

A hierarchical Markov decision process (HMDP) reduces the model complexity by clustering states into groups therefore solving a higher abstraction of the state space but smaller state models for each group [76]. If we map an MDP $\mathcal{M}$ with state space $\mathcal{S}$ as nodes $\mathcal{N}$ of a graph $\mathcal{G}$ and edges $\mathcal{E}$ of $\mathcal{G}$ dictated by transition probabilities $T(a, s, s') > 0$, then we have an initial graph $\mathcal{G}^0$ consisting of all $s \in \mathcal{S}$. Each level of abstraction n is a clustering of the graph $\mathcal{G}^{n-1}$ of the previous level $n-1$.

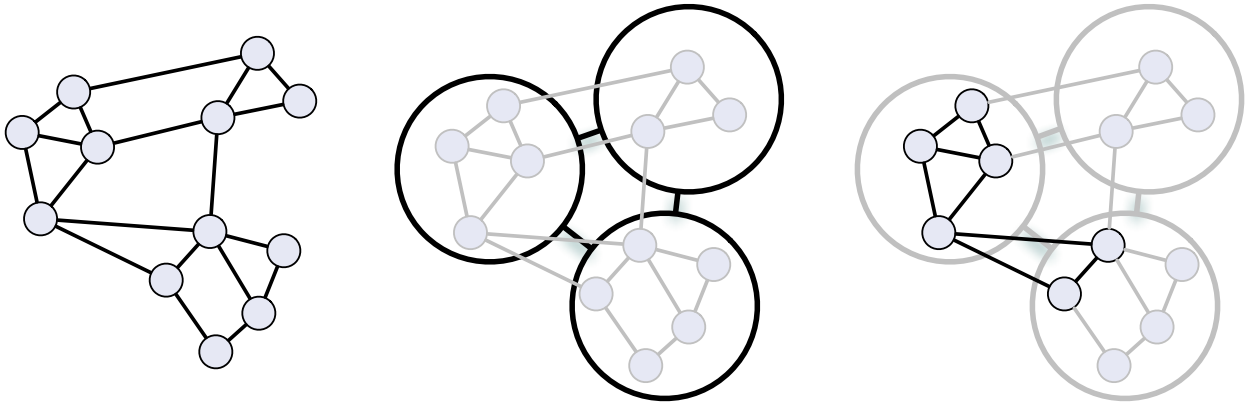


Figure 3.6: General construct of Hierarchical MDP with goal or end states and transfers between upper and lower models.

As can be seen in Figure 3.6, the left-most graph shows the initial graph $\mathcal{G}^0$. A clustering algorithm is used on $\mathcal{G}^0$ to create the next level $\mathcal{G}^1$ shown in the middle with edges (i, j) of the new graph dependent on any $s_i \in \mathcal{G}_i^1$ having a connection to any $s_j \in \mathcal{G}_j^1$ where $j \neq i$. If $\exists s_i \in \mathcal{G}_i^1, s_j \in \mathcal{G}_j^1$ with $T(\cdot, s_i, s_j) > 0$, then $\exists (i, j) \in \mathcal{G}_1$.

The difficulty of clustering states into a hierarchical structure is the translation of the rewards and transitions to higher levels of abstractions created by the clustering. The probability of transitioning between any higher level graph $\mathcal{G}^n$ is $p(s_m^n | s_k^n, a^n)$ which is averaging over $p(s_m^{n-1} | s_k^{n-1}, a^{n-1})$ of the lower level graph $\mathcal{G}^{n-1}$ for states $s_m \in \mathcal{G}^n$. The associated reward R^n is determined by $r(s_k^n, a^n, s_m^n)$ for each state transition from s_k^n to s_m^n the corresponding

$r^{n-1}(s_i^{n-1}, a^{n-1}, s_j^{n-1})$ and averaging over them. Any graph $\mathcal{G}_m^n$ includes all states s that a state in the graph can transition to, which provides a trigger for determining when the system is moving to another graph $\mathcal{G}_m^n$.

3.2.2 Concurrent Markov Decision Processes

A concurrent MDP involves converting a complex or intractable multi-agent MDP into multiple separate *dependent* MDPs. The dependencies allow for each one of the MDPs to reduce the complexity caused by state and action explosion into smaller sets based upon other MDPs.

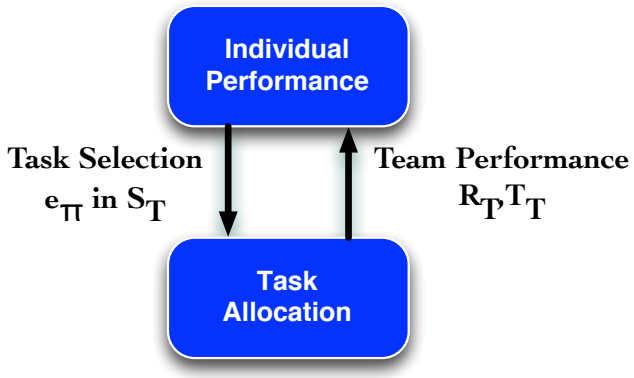


Figure 3.7: Example of a Concurrent Markov Decision Process where separate MDPs have dependent interaction between an Individual MDP and a Task Allocation MDP.

Figure 3.7 is an example of a concurrent MDP from [13] where an Individual MDP and a Task Allocation MDP are dependent on each other. This approach was utilized to reduce the complexity of MMDP or Dec-MDP models of a generalized hierarchical model into a concurrent variant that utilizes advice exchange. The calculation of policies in a concurrent MDP, though there are dependencies, are not complex unless convergence is attempted simultaneously between the two models without proper construction to facilitate. The difficulty of the concurrent MDP approaches is the multi-agent learning, which the authors in

[13] developed concurrent individual and social learning (CISL). CISL is not a topic relevant to this paper but future work.

In order for the agents to collaborate, [13] uses an advice exchange mechanism where the an agent will use actions from another agent's policy. It is assumed that an agent that is acquiring rewards more often than an agent will have learned a better policy, and therefore an agent will use that agent's Individual MDP policies to execute actions.

3.3 Cooperative Decision Making in Multi-agent Systems

The models discussed in the previous section are examples of those used by a single agent who is maximizing the long-term reward for its own actions. When multiple agents are in the environment, there is a need to coordinate the actions to ensure to maximize an overall objective shared by the all agents to prevent sub-optimal task assignments or multiple agents attempting to perform the same task when only one is needed.

The complexity of solving multi-agent cooperative policies is the level of observability of the agents in the entire environment. Per Figure 3.8, they are divided into local or joint observability and partial or full observability. As in the case of MDPs and POMDPs, if an agent is able to observe its state with certainty then its is fully observable. In multi-agents systems, this concept is broadened to determine if an agent is fully observing the environment on its own or requires information being observed from other agents.

In this section, we introduce decentralized Markov decision processes (Dec-MDP) that address multi-agent decision making for both fully and partially observability environments with a variety of subclasses that attempt to address the uncertainty in multi-agent collab-

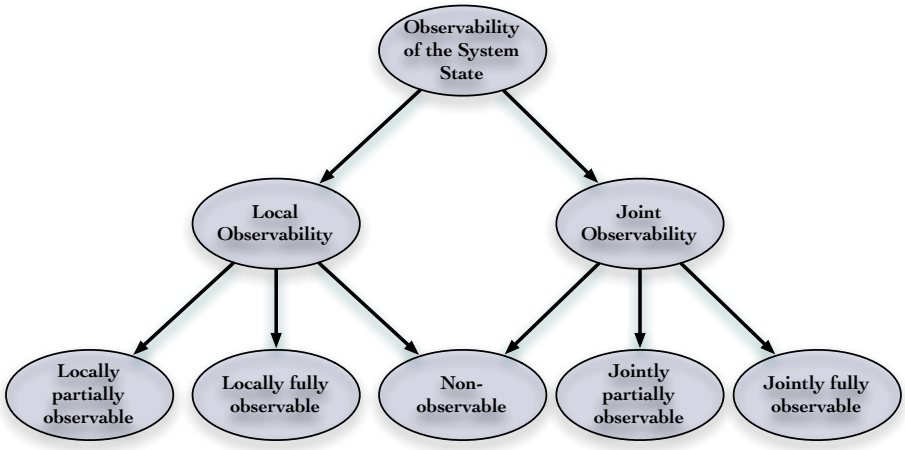


Figure 3.8: A graph tree of observability that dictates the source of observability of an agent as either capable of determining the state of the environment or requiring assistance from another agent.

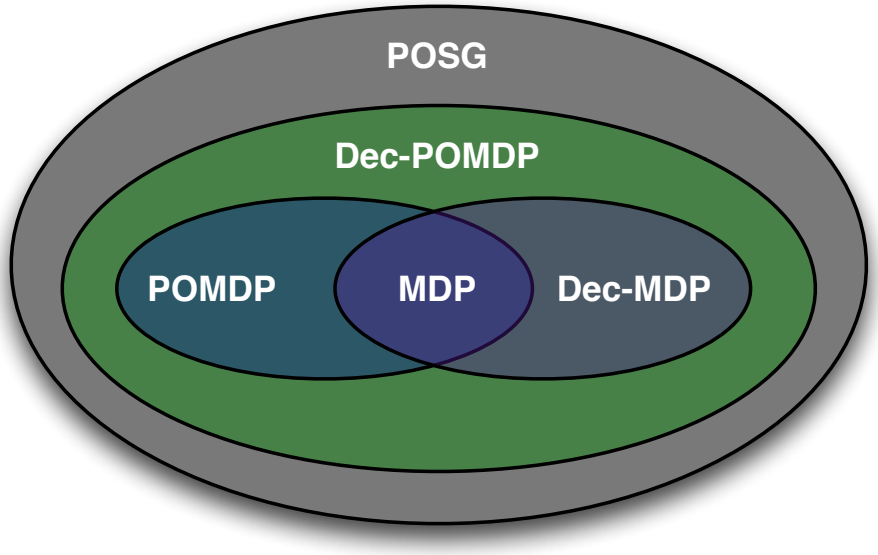


Figure 3.9: Relationships between the various MDP models

oration. Figure 3.9 shows a Venn Diagram of the relationship of the MDP and POMDP discussed earlier and Dec-MDP. If the agents aim for maximization of their local policies, then you have a partially observable stochastic game [77].

A classic problem for discussing decentralized solutions is the multi-robot navigation and

exploration problem [6, 7]. Figure 3.10 shows an example two agents robot navigation problem, where the objective of the agents is to meet as quickly as possible.

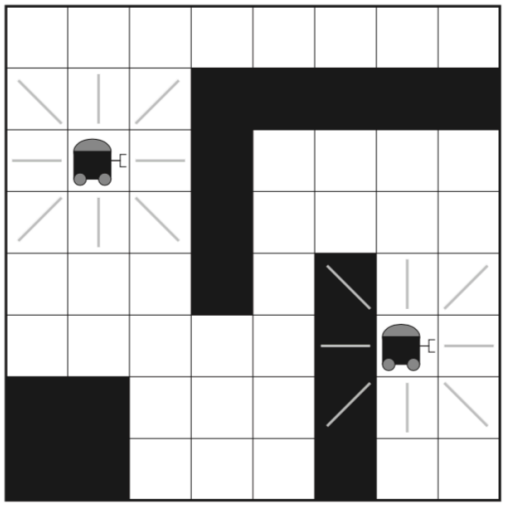


Figure 3.10: A common robot navigation scenario with two agents used for multi-agent benchmarking.

The state space is simply correspond to the position and the actions are stay, up, down, left, and right. Movement in the desired direction has a 0.6 probability of success with an equal 0.1 probability for any of the remaining four positions. Choosing to stay is guaranteed to remain in the same location. There are perfect observation within one square of the agent as shown in the grey lines in the figure, therefore it has perfect observation of the immediate grid cells but not its own location. When both agents occupy the same grid, a reward of 1 is given.

3.3.1 Decentralized Markov Decision Processes

Decentralized Markov decision processes (Dec-POMDP) are a construct that allows multiple independent POMDPs running on different agents to act independently while working towards an objective function that is dependent on all the agent's actions [14, 78]. The

Dec-POMDP consists of the tuple $< I, S, A_i, T, R, \Omega, O >$, where I is a finite set of agents, S are states with initial state distribution b_0, A_i is the set of actions for agent i, $T(s'|s, \mathbf{a})$ is the transition probability, $R(s, \mathbf{a})$ is the reward function, Ω are the set of observations, and $O(\mathbf{o}|s', \mathbf{a})$ is the observation model [77]. In brief, the agents only have access to their local observations and not global observations while objectives and therefore rewards are dependent on all the agents. Decentralized decision-making works with one or more of the following hypothesis [73]: **(a)** each agent has a complete knowledge of the system state, **(b)** each agent has a partial knowledge of the system state, **(c)** the agents can communicate, and/or **(d)** the agents cannot, or can partially, communicate

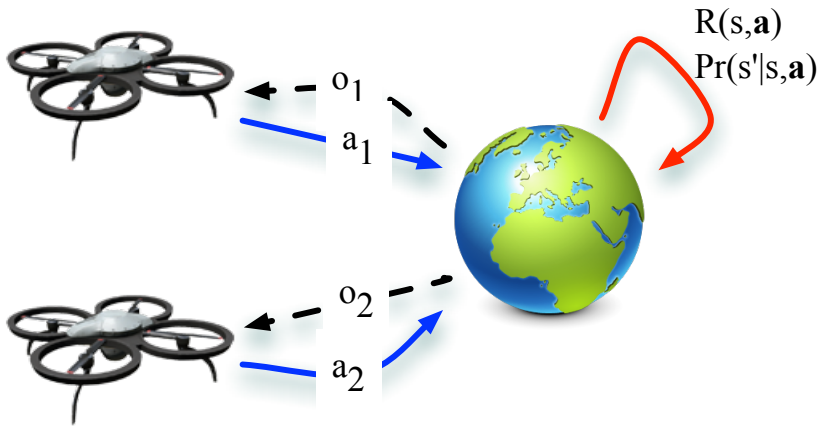


Figure 3.11: The Action, Observation, and Reward Execution of a DecPOMDP.

The basic idea of the Dec-POMDP model is illustrated in Figure 3.11 with a two agent model. Each agent takes its own independent action, $< a_1, a_2 >$, and receives only its own local observation, $< o_1, o_2 >$, but the transitions $Pr(s'|s, a)$ and the rewards $R(s, a)$ are joint. An agent executes its own local policy and the set of local policies is the joint policy. Dec-POMDPs attempt to create a joint policy to maximize the overall expected utility of all the agents. Referring back to Figure 3.9, a Dec-MDP is basically a Dec-POMDP with joint full observability.

At this point, it might be worthwhile to explain the major differences between a Dec-POMDP

and our traditional POMDP in a multi-agent environment. The key difference is that any decision or action by one agent impacts other agents and there is insufficient information to properly reason the actions of other agents. Therefore, belief states are an insufficient statistic and agents may have to remember action and observation histories to *act optimally*.

Provide an excellent explanation of how the action, observation, and state space are tied in a Decentralized POMDP using Figure 3.11 …Since the objective function is dependent on the behavior of all the agents and in the standard model for Dec-POMDP, the agents have no information on the state of the other agents and need to generalize the belief states to account for their *belief* of the other agent's policies. A generalized belief state, b_G, is used to account for the possible policies of the other agents in the utility function

$$U(p, b_G) = \sum_{q,s} b_G(s, q) U(p, q, s), \tag{3.9}$$

though the distributions may not be known, but is useful in the DM process to track their generalized belief.

If there are two agents in the environment working on different tasks and their actions do not effect the rewards or transitions of each other than they can be implemented as independent POMDPs – Otherwise, it is a Dec-POMDP. The complexity of the Dec-POMDP algorithm can be simplified with assumptions of independence in the model, especially assuming inde-

pendence of transitions, observations, and rewards:

$$T(s'|s,\mathbf{a}) = T_0(s'_0|s_0)\prod_i T_i(s'_i|s_i,a_i) \tag{3.10}$$

$$O(\mathbf{o}|s,\mathbf{a}) = \prod_i O_i(o'_i|s_i,a_i) \tag{3.11}$$

$$R(s,\mathbf{a}) = f(R_1(s_i,a_i),\ldots,R_n(s_i,a_i)) \tag{3.12}$$

$$R(s,\mathbf{a}) = R_0(s_0) + \sum_i R_i(s_i,a_i)) \tag{3.13}$$

3.3.2 Stochastic Controller

Partially observable Markov decision processes (POMDPs) are often represented as decision trees or finite state controllers. In Dec-POMDPs, the decision tree can still be used to represent the policy for a finite horizon with the tree branches driven by observation and action histories, though only observation histories can be used [78]. For finite-horizon Dec-POMDPs, policies can be solved that are represented by a policy tree, which is a record of action observation histories (AOH). Solving an infinite-horizon Dec-POMDP, as with POMDPs, is undecidable as it may utilize infinite resources, but policy iteration algorithms can solve an epsilon optimal solution with finite time and memory [7] as a *local* finite state controller. Since we are limited memory, stochastic action selection and node transition may be beneficial [7].

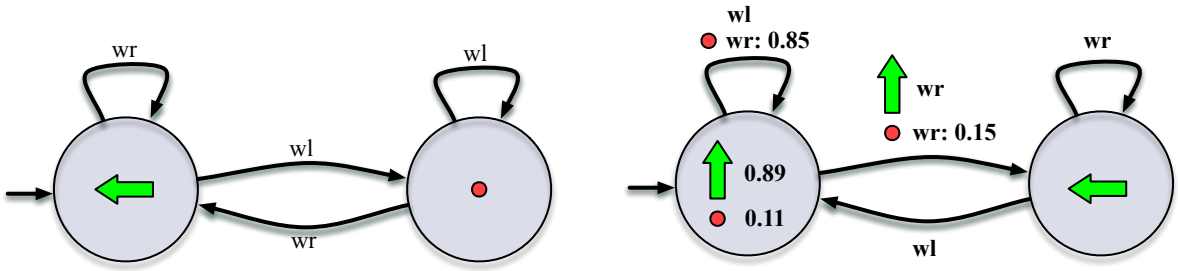
We define a local controller by the tuple $< Q_i, \Omega_i, A_i, \psi_i, \eta_i >$, where [7]

- Q_i is a finite set of controller nodes.
- Ω_i is a set of inputs, taken to be the local observations for agent i.
- A_i is a set of outputs, taken to be the actions for agent i.

- $\psi_i : Q_i \rightarrow \Delta A_i$ is an action selection function for agent i, defining the distribution of actions selected at each node of that agent's controller.

- $\eta_i : Q_i \times A_i \times \Omega_i \rightarrow \Delta Q_i$ is a transition function for agent i, defining the distribution of resulting nodes for each initial node and action taken of that agent's controller.

The combined action selection and node transition for agent i of its local stochastic controller is the conditional distribution $P(a_i, q_i'|q_i, o_i)$, which is parameterized by the above functions ψ_i and η_i. All the local controllers combined consist of the conditional distribution $P(\overrightarrow{a}, \overrightarrow{q}'|\overrightarrow{q}, \overrightarrow{o})$,which is denoted as the *independent joint controller*.



(a) Agent 1. (b) Agent 2.

Figure 3.12: A graphical representation of the stochastic controller for a two agent Dec-POMDP.

A stochastic controller for the two agent robot navigation and exploration problem of Figure 3.10 is shown in Figure 3.12. Agent 1 has no stochastic actions or transitions, but simply will either stay in its current location or move left depending on local observations from the environment. On the other hand, agent 2 has a more complicated stochastic controller that does include stochastic actions and transitions. The controller initializes with the left node and will randomly select to move up with probability 0.89 or stay in position with 0.11. If the random action selection was to stay and a wall is observed to the right (wr), there is 0.85 probability that it will return to the same node of the controller otherwise 15% of the

time it will transition to the right node. The right node has only one action selection, move left.

3.3.3 Correlated Joint Controller

For multi-agent systems, we extend the definition of a stochastic controller to include a *correlation device*, which is a shared source of randomness between all the agents [7]. It has been shown by [7, 79] that a joint probability distribution for action selection with a correlated policy will yield higher overall infinite discounted reward than the utilization of independent probability distributions. They introduce a new finite state controller, the correlation device, consisting of the tuple $< Q_c, \psi_c >$, where Q_c is a set of nodes and $\psi_c : Q_c \rightarrow \Delta Q_c$ is a state transition function.

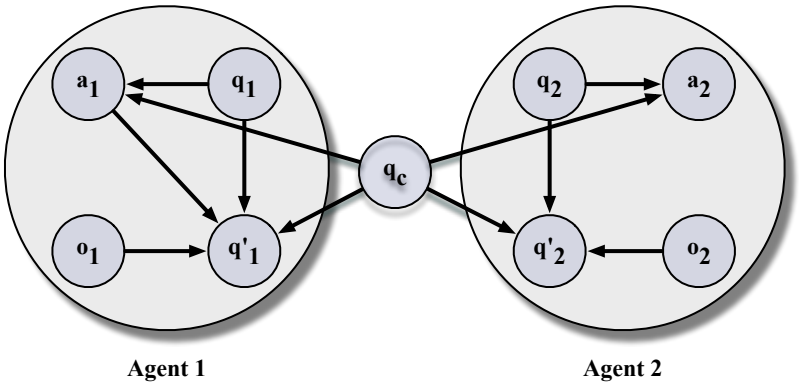


Figure 3.13: A graphical representation of the probabilistic dependencies in a correlated joint controller for two agents.

The local controller for an agent i is expanded to take the state of the correlation device as input to form the conditional distribution $P(a_i, q_i' | q_c, q_i, o_i)$. The *correlated joint controller* is combination of all the individual controllers and the correlation device to form a joint conditional distribution $P(\overrightarrow{a}, \overrightarrow{q}' | \overrightarrow{q}, \overrightarrow{o})$, where $\overrightarrow{q} =< q_c, q_1, \ldots, q_n >$. Figure 3.13 shows a graphical representation of probabilistic dependencies between the local controllers and the

correlation device. Therefore, the value function for a correlated joint controller is calculated by solving the set of linear equations [7] for each $s \in S$ and $q \in Q$

$$V(s, \overrightarrow{q}) = \sum_{\overrightarrow{a}} P(\overrightarrow{a} \mid \overrightarrow{q}) \left[R(s, \overrightarrow{a}) + \beta \sum_{s', \overrightarrow{o}, \overrightarrow{q}'} P(s', \overrightarrow{o} \mid s, \overrightarrow{a}) P(\overrightarrow{q}' \mid \overrightarrow{q}, \overrightarrow{a}, \overrightarrow{o}) V(s', \overrightarrow{q}') \right] \tag{3.14}$$

where $P(a|q)$ is the probability action a will be taken in node q and $P(q'|q, a, o)$ is the probability the controller will transition to node q' from node q after action a was taken and o was observed. For an initial state distribution, the value function is defined as $V(b) = \max_{\overrightarrow{q}} \sum_s b(s) V(s, \overrightarrow{q})$ and the controller is started in the node that maximizes the value function. The correlation device improves the value function of a fixed-size controller, but with an initial state distribution and relaxation of the size restrictions an uncorrelated joint controller that is larger than the correlated can be found with at least the same value [7].

3.3.4 Policy Iteration Algorithm for Decentralized Controllers

As is done with the standard POMDP policy iteration algorithm, the DecPOMDP algorithm (Figure 3.14) interleaves node generation with pruning. Though, it is still an open problem wether this interleaving of generation and pruning in multiagent cases can converge [7]. An exhaustive backup to ensure convergence has $|A_i||Q_i|^{|\Omega_i|}$ nodes being added per agent i to their local controller per one-step policy. Therefore, a joint controller grows by $|Q_c||\Pi_i||A_i||Q_i|^{|O_i|}$ nodes.

For the multiagent case, a transformation of a controller C to a controller D is considered a *value preserving transformations* if $C \leq D$, where $\leq$ is defined as the mapping $f : \vec{Q} \to \Delta\vec{R}$ [7] such that

1: **procedure** DECPOLICYITERATION(π_0, ϵ)
2: $t \leftarrow 0$
3: **repeat**
4: $\pi_{t+1} \leftarrow$ EXHAUSTIVEBACKUP(π_t)
5: $V^{\pi_{t+1}} \leftarrow$ POLICYEVAL(π_t)
6: **repeat**
7: $\hat{\pi}_{t+1} \leftarrow \pi_{t+1}$
8: **for** $i \in I$ **do**
9: $\hat{\pi}_{t+1} \leftarrow$ PRUNE($\hat{\pi}_{t+1}, i$) ▷ Equation (3.19)
10: UPDATECONTROLLER($\hat{\pi}_{t+1}, i$)
11: $V^{\hat{\pi}_{t+1}} \leftarrow$ POLICYEVAL($\hat{\pi}_{t+1}$)
12: **end for**
13: **until** $|\pi_t| = |\hat{\pi}_t|$
14: $t \leftarrow t + 1$
15: **until** $\frac{\gamma^{t+1}|R_{max}|}{1-\gamma} \leq \epsilon$
16: **return** π_t
17: **end procedure**

Figure 3.14: Policy iteration for Dec-POMDPs [6].

$$V(s, \vec{q}) \leq \sum_{\vec{r}} P(\vec{r}|\vec{q}) V(s, \vec{r}) \quad \forall \quad s \in S \quad \vec{q} \in \vec{Q} \tag{3.15}$$

Unlike the single agent POMDP, termination for ϵ-optimality cannot be calculated through a Bellman equation, since it cannot be reliably calculated in a joint controller [7], but instead terminates when discounting β after step t has reduced the maximum reward below ϵ (Figure 3.14 Line 15)

$$\frac{\beta^{t+1}|R_{max}|}{1-\beta} \leq \epsilon. \tag{3.16}$$

Using a dual interpretation of pruning applied to the multiagent case, we consider the removal of a node from the local controller or the correlated devices with the nodes of the other controllers to be a member of the hidden state. For example, to remove node q_i from agent

i's local controller, we need to find a distribution $P(\hat{q}_i)$ over nodes $\hat{q}_i \in Q_i \setminus q_i$ such that

$$V(s, q_i, q_{-i}, q_c) \leq \sum_{\hat{q}_i} P(\hat{q}_i) V(s, \hat{q}_i, q_{-i}, q_c) \quad s \in S, q_{-i} \in Q_{-i}, q_c \in Q_c \tag{3.17}$$

where Q_{-i} is the set of nodes for the other agents. A linear program, shown in Figure 3.15a, has been formulated to find if such a distribution exists when $\Delta \geq 0$. If found, the dominated node is merged into the combination of other nodes by redirecting incoming links to the old node based upon on the distribution $P(\hat{q}_i)$. In a similar way, a linear program, shown in Figure 3.15b, for a correlated device a distribution $P(\hat{q}_c)$ over nodes $\hat{q}_c \in Q_c \setminus q_c$ such that

$$V(s, \vec{q}, q_c) \leq \sum_{\hat{q}_c} P(\hat{q}_c) V(s, \vec{q}, \hat{q}_c) \quad s \in S, \vec{q} \in \vec{Q} \tag{3.18}$$

Others algorithms for solving DecPOMDP models are available, including best-first that searches a Cross Product MDP to solve for an FSC [80], Moore machines [81], nonlinear programming (NLP) [82], point-based [83], and heuristic approaches like directed pruning [83].

3.3.5 Notable Subclasses of Dec-POMDPs

There are many subclasses of the Dec-POMDP that address joint and local observations, communications, model independence, et cetera [6, 14]. There are specifically three subclasses relevant to this paper: Networked Distributed POMDP (ND-POMDP), Multi-agent Markov decision process (MMDP), and Dec-POMDP with Communications (Dec-POMDP-Com).

The ND-POMDP is a Dec-POMDP with a factored reward structure and transition and observation independence [6]. The dependence between agents is represented as a coordi-

$$
\begin{aligned}
&\textit{Objective}: \quad \max \Delta \\
&\textit{Subject to}: \quad V(s,q) + \Delta \leq \sum_a \left[x(a) R(s,a) + \beta \sum_{a,o,q'} P(o,s'|s,a) V(s',q') \right] \quad \forall s \\
&\sum_a x(a) = 1 \\
&\sum_{q'} x(a,o,q') = x(a) \quad \forall a, o \\
&x(a) \geq 0 \\
&x(a,o,q') \geq 0 \quad \forall a, o, q'
\end{aligned}
\tag{3.19}
$$

(a) The linear program to find a replacement for agent i's node q_i where $x(a)$ represents $P(a|q)$ and $x(a,o,q')$ represents $P(a,q'|q,o)$.

$$
\begin{aligned}
&\textit{Objective}: \quad \max \Delta \\
&\textit{Subject to}: \quad V(s, \overrightarrow{q}, q_c) + \Delta \leq \sum_{\hat{q}_c} x(\hat{q}_c) V(s, \overrightarrow{q}, \hat{q}_c), \quad \forall s, \overrightarrow{q} \\
&\sum_{\hat{q}_c} x(\hat{q}_c) = 1 \quad \forall \hat{q}_c \quad x(\hat{q}_c) \geq 0
\end{aligned}
\tag{3.20}
$$

(b) The linear program to find a replacement for a node q_c in the correlated device.

Figure 3.15: Linear programming formulations for controller reductions in DecPOMDP Policy Iteration [7].

nation graph [84, 85, 86], as shown in Figure 3.16, and the vertices and edges represent the dependence. The factorization of transition and observation by dependence or conditional independency allows for the reduction of the model and therefore the computational time to solve, which is especially important for embedded applications. The reward for the coordination graph in Figure 3.16 is given as

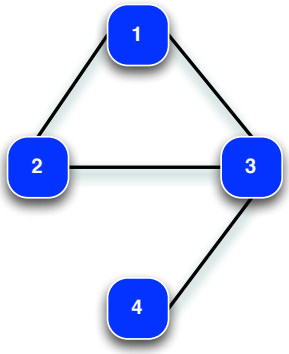


Figure 3.16: Coordination graph used in networked distributed POMDP (ND-POMDP) to factor the reward dependence between agents and reduce combinatorial explosion with multi-agent coordination.

$$R_{1234}(\cap_{k=1}^{4} s_k, a_k) = R_{123}(s_1, s_2, s_3, a_1, a_2, a_3) + R_{34}(s_3, s_4, a_3, a_4). \tag{3.21}$$

An MMDP is a simplification of the Dec-POMDP in that the agents are able to fully observe the true state and therefore know the states of the other agents. This allows an MMDP to be solved as an MDP with only some communication to ensure policy consistency. Though this situation is not always true in the model framework developed in this paper, with the application of a coordination graph of the ND-POMDP, there are points in time were the agents in during active communications will for an instant reduce to a MMDP allowing for belief states and observation models to drastically reduce uncertainty.

The Dec-POMDP-Comm [87] is the dominant subclass that will be expanded on in this paper where the agents are reasoning on when to communicate because of the delay and costs associated it. The model includes the addition of a tuple consisting an alphabet of possible communications $< n, S, A, \Sigma, C_\Sigma, P, R, \Omega, O, T >$, where Σ is the alphabet of

communications, $\sigma_i \in \Sigma$ is the message sent by agent i, and ϵ_σ is a null message with a cost C_Σ of zero. It creates two policies: (1) An action policy with belief states to actions and (2) a policy that associates the belief states with communication actions. There are two primary approaches for determining when a node should communication its observation histories with other agents. The most commonly seen approach is to make the decision a part of Dec-POMDP model based upon joint state and actions. There is also an approach where there is a centralized POMDP plan that is communicated between the agents which triggers communications when their individual Dec-POMDP plan differs from the centralized one.

A similar approach is Dec_POMDP_Value_Comm [88] that implements online policy generation over local observations. It uses a parameter α which values communication as a weighting of the information theoretic value of the message and the reward function. Unfortunately, the parameter needs to be hand-tuned for the model or domain. ACE-PJB-Comm [89] makes communication an online problem and solving a centralized POMDP offline. Agents still maintain an estimate of the joint belief state and make decision on communication based upon it.

3.4 Reward Shaping and Belief-dependent Rewards

Driving communications from the *value of information* similar to CA-POMDP is not unique, but has been the focus of research behind reward shaping and belief-dependent rewards [28, 29, 31]. Solving POMDPs with belief dependent rewards complicates the piecewise, linear, and convex assumption used in common solution techniques, especially for the typical value and policy iteration algorithms. The following approaches address solutions to these common problems and are the inspiration behind the solution covered in the CoDec POMDP for IKD.

3.4.1 Belief-Space Rewards

In many situations, the model needs to address the necessity of *active sensing* of the environment to acquire knowledge about state uncertainty. Ultimately, the objective needs to be expressed over the uncertainty on the state or minimize the entropy over a given state variable. ρPOMDP [29] extend the POMDP framework for a reward function ρ that depend on the current belief states rather than just on the action and state. Though, they assume that ρ is convex to minimize changes to POMDP exact and approximate solving algorithms.

At the core of ρ-based POMDPs is to replace the traditional reward function $r(s, a)$ with a function $\rho(b, a)$ with the most common methods based on Shannon's information theory, especially entropy and KL Distance. To use information theoretic measures the reward function must measure information rather than entropy, such as a negative entropy function

$$\rho_{ent}(b) = log_2(|S|) + \sum_{s \in S} b(s) log_2(b(S))$$

. If the function is a convex function, then the POMDP property holds and measurements of uncertainty are naturally convex functions.

Uses a finite-horizon solution where $V_0 = 0$ It represents the function as several Γ-sets, one Γ_ρ^a for each a and updates the value function of the POMDP to reflect the new definition. The reward is calculated as

$$\rho(b, a) = \max_{\alpha \in \Gamma_\rho^a} [b(s)\alpha(s)],$$

which leads to a new definition of the value function as

$$V_n(b) = \max \sum_s b(s) \left[\chi_\rho^\alpha(b, s) + \sum_o \sum_{s'} T(s, a, s,') O(s', a, o) \chi_{n-1}(b^{ao,o}, s') \right]$$

where $\chi_\rho^\alpha(b, s) = \arg\max_{\alpha \in \Gamma_\rho^a}(b \cdot \alpha)$ which generates $|\Omega| \times |A|$ Γ-sets.

For further understanding of how to solve a belief MDP with ρ-based rewards through exact updates or point-based approximations, see [29]. The belief-dependent rewards framework of ρPOMDP is beneficial for creating a framework that balances exploration and future work can investigate the incorporation of belief-dependent rewards in the IKD mode.

3.4.2 Reward Shaping

In the context of multi-agent systems, reward shaping is a process for adjusting the expected joint reward of multiple collaborative agents based upon how coordinated the they are [28]. The goals of the author was to utilize reward shaping to value communication within decentralized POMDP policy generation based on local observations and communication histories. Their application of reward shaping uses the concept of *belief divergence* to ascertain the necessity of communication. It uses reward shaping based upon an information theoretic measure between two probability distributions (either absolute difference, Relative Entropy or KL Divergence) with the two distributions being the reference belief of states of the DecPOMDP and the current belief

Authors use the dec_POMDP_com as the basis but change the perspective from a cost to a reward for communication to highlight an opportunity to use a resource and state efficient policy generation techniques will be adapted to allow for scalability. They use the information theoretic idea from dec_POMDP_value_comm by approximating α using a measure of belief divergence. They also don't like the dec_POMDP_com creating a separate policy for associating belief states with communication acts because the agent may need to use common resources

$$B_d(B_i, B_j) = D_{KL}(B_i||B_j) = \sum_{s \in S} \log \frac{B_i(s)}{B_j(s)} \tag{3.22}$$

approximate divergence $B_{da} = B_d(B_i, B_{ref})$, where B_{ref} is an established reference point of the belief from last synchronization of information between the agents.

If the agents have the same belief, then $B_d = 0$ and the policy is calculated as it would normally be for a decentralized POMPD. On the contrary, if the normalized belief divergence is maximized ($\bar{B} = 1$) then an agent will assume a joint action where the other agents act randomly:

$$E(a)_r = \frac{1}{|A_{-i}|} \sum_{a_{-i} \in \mathcal{A}_{-i}} \sum_{s \in S} \sum_{s' \in S} [P(s, < a_i, a_{-i} >, s') \cdot R(s, < a_i, a_{-i} >, s')] \tag{3.23}$$

Therefore, they calculate an expected reward of a joint action R_{rs}

$$R_{rs}(a, B_d) = f(B_d, E(a)_u, E(a)_r) \tag{3.24}$$

where the belief divergence is normalized $0 < B_d < 1$ and the shape is determined by a function f is determined by a domain expert. The authors provide an example for Multiagent Tiger and RoboCupRescue domains, common benchmarks in the multi-agent community, in their paper [28] for defining the function.

For our solution, the authors focus on restricting communications to when they are needed, whereas we are not trying to minimize their utilization but stay within soft constraints. CA-POMDP does utilize the belief-state distribution divergence metrics between the DecPOMDP planning algorithm and the CA-POMDP communication algorithm within the CoDec POMDP framework.

Chapter 4

The Constrained-Action POMDP

The *Constrained-Action POMDP* is a formulation that seeks to find a near optimal policy in a partially observable environment with action-based constraints that are probabilistically guaranteed to stay within specified *soft* limits. The framework first solves for an unconstrained optimal policy through policy iteration which provides a policy that is represented as a finite state controller (FSC), which is the first step in Figure 4.1. Then, the framework attempts to improve the probabilistic constraint satisfaction of the FSC through a branch and bound (BnB) discrete optimization technique (Section 4.4) with an objective of minimizing the loss in the value function when introducing new actions that utilize less resources than those in the FSC but are not as valuable.

Any action performed by an autonomous system will have a distribution representing its utilization of resources, and thus CA-POMDP aims to determine the probability that a series of actions within an FSC at each step will respect a soft constraint through an analysis of the cumulative distribution function (CDF). In the formulation of the CA-POMDP, the evaluation of probabilistic constraint satisfaction is performed by randomly sampling a FSC policy with Markov chain Monte Carlo (MCMC) (Section 4.3). The value function of the FSC is evaluated just as it is in POMDP policy iteration – through solving a set of linear equations (LE). The constraint improvement branch and bound algorithm (BnB) terminates on the outer loop when the introduction of a new state to the FSC does not improve the value function nor constraint satisfaction and on an inner loop when the BnB Tree has been

fully explored.

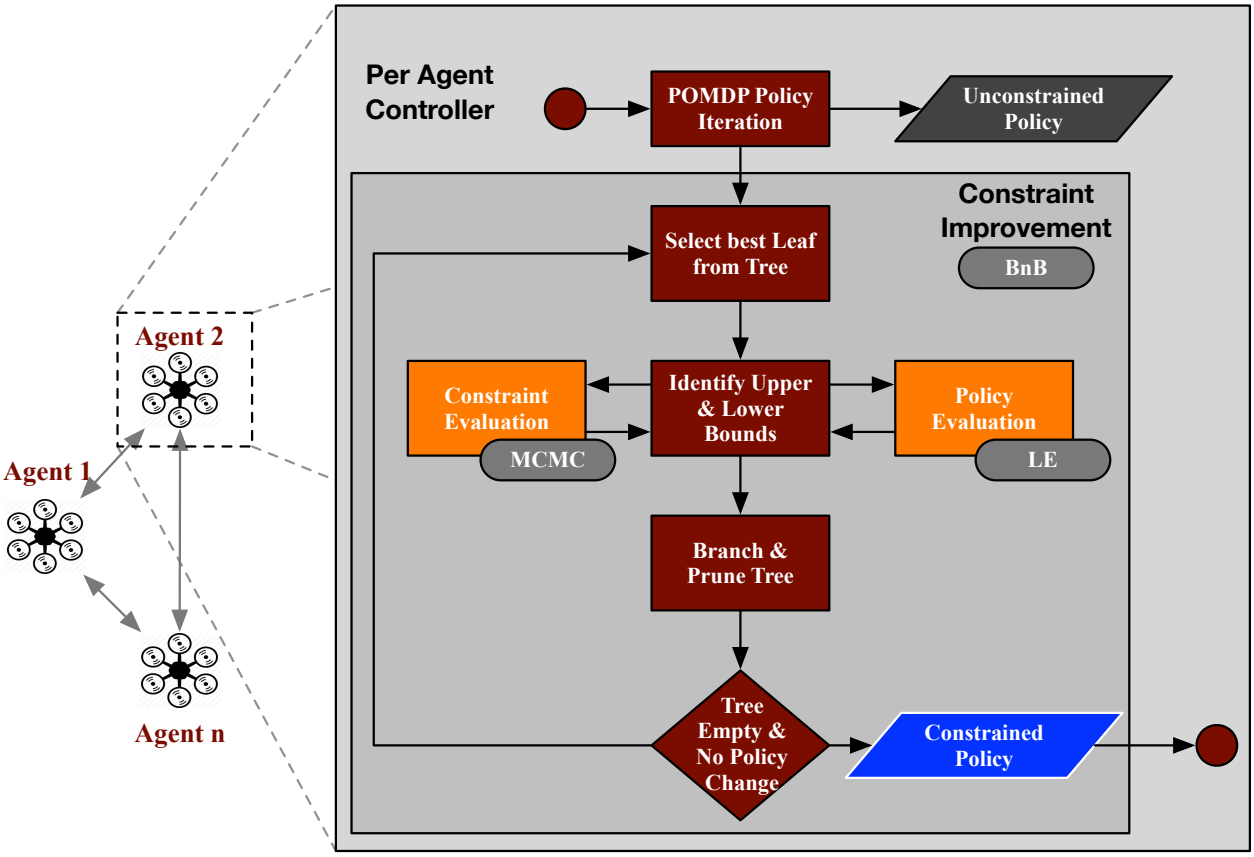


Figure 4.1: Generic logic diagram of the CA-POMDP methodology indicating the points different techniques are integrated into the process.

The goal is to adapt an infinite-horizon finite state controller to probabilistically guarantee not to violate the soft constraints within a desired period of time. Policies are represented as a finite-state controller (FSC), since an FSC is a cyclic graph that represents an infinite-horizon POMDP well [83]. An infinite-horizon policy allows the controller to run continuously without needing to restart back at the initial horizon point of finite horizon. It also can accelerate the convergence of an appropriate FSC, since an infinite horizon policy in value iteration is not guaranteed to converge [17].

The FSC is a directed cyclic graph, $\mathcal{G}$, with the vertices, $\mathcal{A}$, representing *machine states* consisting of actions and edges, $\mathcal{E}$, representing transitions to another machine state determined

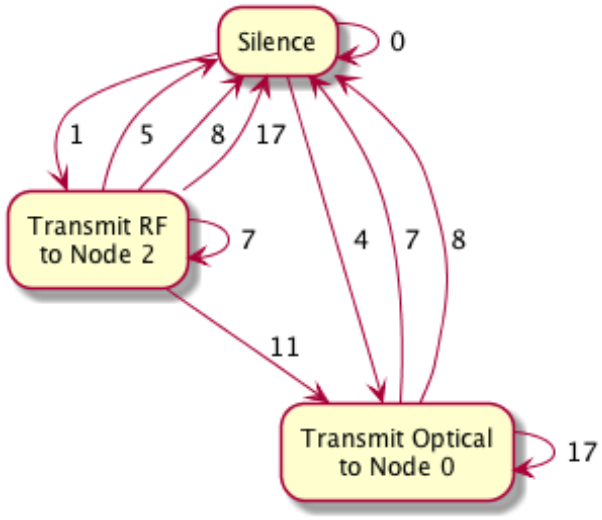


Figure 4.2: An example of a resulting Finite-state Controller for a constrained-action POMDP where the vertices represent actions and the edges are transitions based upon environmental observation. Edge Transitions are represented as numbers since the quantity of observations requires long descriptors.

by the observation seen after executing the action of the vertex. Figure 4.2 shows a simplified optimal policy, π^*, represented as an FSC with actions at the indices (e.g., Silence) and an observation indices labelling transitions, $\mathcal{A} \times \mathcal{A}$, which represent a specific observation. Constraints are applied to the entire agent and can consist of a set of constraints, $\mathcal{C}$, (e.g., power and bandwidth) within a specified time period or epoch. The resource utilization of each one of the constraints, $c \in \mathcal{C}$, is defined on a per actions basis as a Gaussian distribution, $c(a) \leftarrow \mathcal{N}(\mu, \sigma)$. The FSC representing the optimal policy will have machine states introduced into the controller, *constraint states*, through discrete optimization to bring the controller within a probabilistic constraint satisfaction while minimizing the impact on the optimal value function.

4.1 Constrained-Action POMDP Model

Our goal is to adapt an infinite-horizon FSC of a POMDP policy π from policy iteration [27] to probabilistically stay within soft constraints over a desired period of time T, i.e., $p(\sum_{t=0}^{T} u_h(a_t) \leq c_h) \geq \eta_h$, for all constraints $h \in \mathcal{H}$, with resource limits $c_h \in \boldsymbol{C}$, where $u_h \in \boldsymbol{U}$ is resource utilization used by the action a_t. Policies π are represented as a finite-state controller (FSC), since an FSC is a cyclic graph that represents an infinite-horizon POMDP well [83]. An infinite-horizon policy allows the controller to run continuously without needing to restart back at the initial horizon point of a finite horizon. The FSC representation can also accelerate the convergence to a policy, since an infinite horizon policy in value iteration is not guaranteed to converge [17]. The CA-POMDP can be described formally by the tuple

$$< \boldsymbol{S}, \boldsymbol{O}, \boldsymbol{B}, \boldsymbol{A}, \boldsymbol{T}, \boldsymbol{R}, \boldsymbol{\Pi}, \mathcal{H}, \boldsymbol{C}, \boldsymbol{U}, \boldsymbol{E} > \tag{4.1}$$

where $\boldsymbol{S}$, $\boldsymbol{O}$, $\boldsymbol{B}$, $\boldsymbol{A}$, $\boldsymbol{T}$, $\boldsymbol{R}$, and $\boldsymbol{\Pi}$ are the same as defined in a POMDP, $\boldsymbol{\Sigma}$ is the alphabet of communications $\sigma_i \in \boldsymbol{\Sigma}$ that agent i can transmit to a neighbor $n \in \mathcal{N}$ as the action $a_{i,n,\sigma} \in \boldsymbol{A}$, $\mathcal{H}$ is the set of resources being used by the agents, $\boldsymbol{C}$ is the matrix defining the constraints for each resource, $\boldsymbol{U}$ defines the utilization of resources for an action $a \in \boldsymbol{A}$ and its variation or uncertainty, and $\boldsymbol{E}$ is the matrix of edge observation probabilities for transitioning from a machine state i to another j in an FSC based upon action observation histories (AOH).

The constraint model consists of a matrix of soft constraints $\boldsymbol{C}$ (e.g., power and bandwidth), the utilization of resources $\boldsymbol{U}$ per time epoch Δt (a decision is made each epoch of time), and the matrix of observation probabilities $\boldsymbol{E}$. Specifically, $\boldsymbol{C} \in \Re^{|\mathcal{H}| \times 2}$ is a matrix representing the desired soft constraints within a specified time period T where each row $h \in \mathcal{H}$ of $\boldsymbol{C}$ is the vector $[c_h, \eta_h]$, with c_h the desired upper limit for resource h, and $0 < \eta_h < 1$ is the

probabilistic constraint satisfaction for that resource based upon operational scenarios. The resource utilization $\boldsymbol{U}$ defines for each resource $h \in \mathcal{H}$ a matrix of Gaussian probability distribution parameters for all actions, i.e., $u_h(a) \leftarrow \mathcal{N}(\mu_h^a, \sigma_h^a)$. The matrix $\boldsymbol{E} \in \Re^{|\pi| \times |O|}$ consists of edge observation probabilities $p(o|a_{i \in \pi})$ for encountering an observation $o \in \boldsymbol{O}$ while at a machine state i of the finite state controller policy $\pi \in \Pi$. Edge observation probabilities are tied to the FSC policy representation as a means to track the stochastic nature of the environment causing observation transitions from one machine state to another. Their probabilistic distribution is initially assumed to be uniform and the true distribution is learned online, see Section 4.5.

Our core mechanism in CA-POMDP is the FSC representing the optimal policy calculated using Hansen's POMDP PI algorithm [27], along with discrete optimization which acts to constrain the policy. During the discrete optimization phase, CA-POMDP will introduce *constraint states* j into an unconstrained controller to bring the controller within a probabilistic constraint satisfaction η (analyzed via MCMC) while minimizing the impact on the optimal unconstrained value function V^*. To effectively alter the controller with constraint states, the non-dominant alpha vectors per action, defined as set Ω, are maintained during the dynamic programming updates for unconstrained policy optimization. These alpha vectors are not dominant at any dynamic programming update, but may utilize fewer resources and therefore should be considered for altering the controller for probabilistic constraint satisfaction.

4.2 Random Sampling of Markov Processes

To analyze the probabilistic constraint satisfaction of an FSC policy from an *unconstrained* POMDP, the Markov chain Monte Carlo (MCMC) sampling methodology is applied. An

FSC representation of a POMDP Policy when being randomly sampled is a Markov process. The Markov process transitions are driven by observation probabilities to new states based upon the current state. A pure Monte Carlo approach is insufficient since the samples are not independent and identically distributed (iid) random variables, whereas MCMC is a well known general technique for stochastic simulation methods with Markov processes.

A MCMC is a particle-based approximate inference sampling technique that uses a sequence of samples to iteratively approach a desired posterior distribution [90]. For CA-POMDPs, the MCMC provides a mechanism to continuously sample an FSC policy seeking a posterior that represents its resource utilization. When samples are initially likely to be from the prior distribution, the sampling of a sequence of samples will successively approach the posterior, whereas likelihood based approaches are unlikely to account for this fact and place undue weighting on earlier samples. Another gain in utilizing MCMCs in determining the posterior distributions is the versatility in the type of distribution that can be inferred without being bound to Gaussian distributions or Gaussian approximations.

The Metropolis-Hastings algorithm to solving MCMC was selected because of the difficulty of direct sampling from the probability distribution of a cyclic controller and calculating the normalizing factor of the distribution. The algorithm uses a random walk approach that either accepts or denies a proposal rather than trying to track importance weights. The acceptance ratio reduces simply to

$$\mathcal{A} = \frac{P(x|\mu)P(\mu)}{P(x|\mu_0)P(\mu_0)} \tag{4.2}$$

of the proposed posterior distribution, $P(x|\mu)P(\mu)$, over the current posterior distribution, $P(x|\mu_0)P(\mu_0)$. The most difficult part of calculating the posterior with the Bayes formula, the evidence $P(x)$, is common to both the proposed and current posterior and therefore

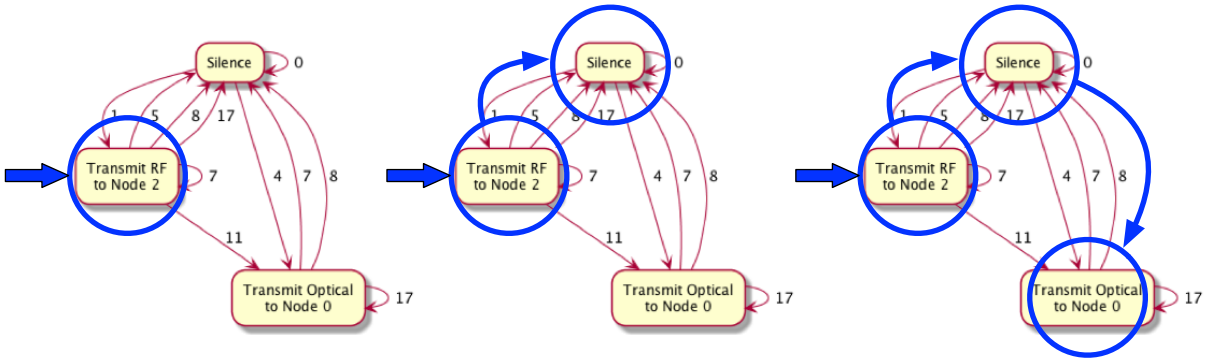
conveniently cancels out.

If a random number from $[0, 1]$ is lower than the acceptance ratio (4.2), then the proposed posterior is accepted. In cases where the proposed distribution is larger than the current distribution, then the acceptance ratio will be greater than one $\mathcal{A} \geq 1$ and therefore the proposed distribution will always be accepted. On the contrary when the acceptance ratio is less than one $\mathcal{A} < 1$, then there is a uniform probability that it will accept the proposed distribution.

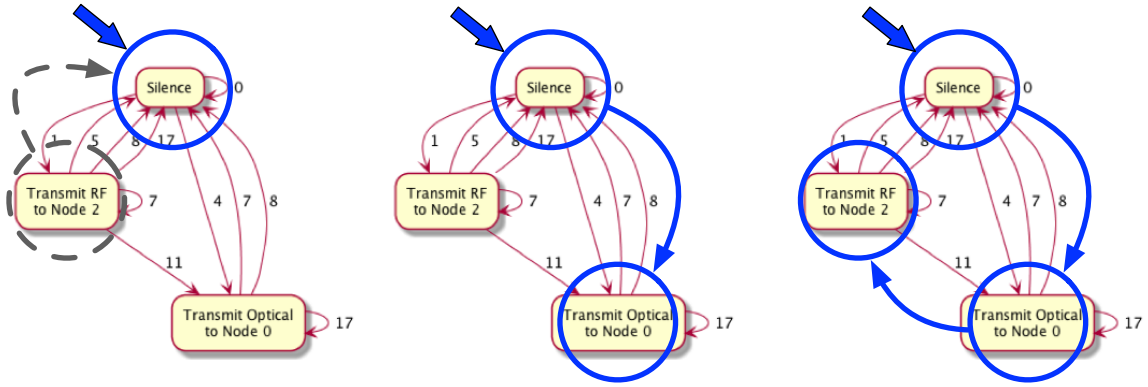
4.3 Constraint Satisfaction Evaluation of an FSC Policy

Evaluating constraint satisfaction, shown in Figure 4.4, is accomplished by random sampling of the finite state controller (FSC) to estimate the resource utilization through a Markov chain Monte Carlo (MCMC) Metropolis-Hastings algorithm. An FSC representation of a POMDP Policy when being randomly sampled is a Markov process. The Markov process transitions are driven by observation probabilities to new states based upon the current state. A pure Monte Carlo approach is insufficient since the samples are not independent and identically distributed (iid) random variables, whereas MCMC is a well known general technique for stochastic simulation methods with Markov processes.

Using the length of time or epochs over which the constraints are being applied T, the finite state controller is sampled $T/\Delta t$ times, where an action is taken every Δt. Starting from a random initial machine state, this sampling process follows the FSC edge transitions around the controller and records the resource utilization per action by sampling from distributions $\mathcal{N}(\mu_h^a, \sigma_h^a)$, for each resource $h \in \mathcal{H}$ as shown in Figure 4.3a. The samples are combined into



(a) Initial step through an FSC with $T = 3$.



(b) Second mixing step starting from a random new state along the observation probabilities.

Figure 4.3: Example of the MCMC sampling process for constraint satisfaction evaluation with (a) showing an initial starting point that is than randomly sampled multiple times until mixed at which point (b) the sampling starts at another random point along an observation edge.

an aggregate sample x that represents the total mean and deviation of resource utilization for the entire time period T. This is repeated from the same initial machine state to gather a sequence of samples $\boldsymbol{x}(\alpha)$ until the chain has approached a representative posterior $P^T(\alpha)$ of the FSC resource utilization for that particular initial machine state, where α represents the distribution of sample data. The posterior has been considered "mixed" when the variational distance is within epsilon [90] (Line 13)

$$\mathbf{D}(P^{(T)}; \boldsymbol{x}(\alpha)) = \max_{\alpha} |P^{(T)}(\alpha) - \boldsymbol{x}(\alpha)| \leq \epsilon_D. \tag{4.3}$$

procedure CONSTRAINTEVAL($\pi^*, \boldsymbol{U}, \boldsymbol{A}, \boldsymbol{C}, \boldsymbol{B}, l$)
  $\boldsymbol{P}^{(T)} \leftarrow \emptyset$
  repeat
    $b_0 \leftarrow$ RANDOM($b \in \boldsymbol{B}$)
    $i^* \leftarrow \arg\max_{i \in \pi^*} V_i \cdot b_0$
    $\boldsymbol{x} \leftarrow \emptyset$
    while True **do**
      for $t \leftarrow 1$ to T step Δt **do**
        $\boldsymbol{x} \leftarrow \boldsymbol{x} \cup U(a_{i^*} \in \boldsymbol{A})$
        $i^* \leftarrow$ RANDOMEDGE($p(o|i^*), P^{(T)}(\alpha)$)
      end for
      $\boldsymbol{x}(\alpha) \leftarrow$ GAUSSIANFIT($\boldsymbol{x}$)
      if $D(P^{(T)}; \boldsymbol{x}(\alpha)) \leq \epsilon_D$ **then** ▷ Equation (4.3)
        break
      end if
      if RANDOM$(0,1) \leq \mathcal{A}$ **then** ▷ Equation (4.2)
        $P^{(T)}(\alpha) \leftarrow \boldsymbol{x}(\alpha)$
      end if
    end while
    $\boldsymbol{P}^{(T)} \leftarrow \boldsymbol{P}^{(T)} \cup P^{(T)}(\alpha)$
  until $Cov\left[\boldsymbol{P}^{(T)}[m]; \boldsymbol{P}^{(T)}[m+l]\right] < \epsilon_{cov}$ ▷ Equation (4.4)
  return $\int_{-\infty}^{\eta \in \boldsymbol{C}} \boldsymbol{P}^{(T)} dx$
end procedure

Figure 4.4: Algorithm for Evaluating a Finite State Controller for its Probabilistic Constraint Satisfaction.

To determine when the algorithm needs to stop taking samples from random machine states along the observation transition probabilities of the current start point per Figure 4.3b, the algorithm continues until the autocovariance of the lagged MCMC distributions, a generalization of the central limit theorem to Markov chains, has converged [90] to within ϵ_{cov}

$$Cov\left[\boldsymbol{P}^{(T)}[m]; \boldsymbol{P}^{(T)}[m+l]\right] \approx \tag{4.4}$$
$$\frac{1}{M-l} \sum_{m=1}^{M-l} (\boldsymbol{P}^{(T)}[m] - \hat{\boldsymbol{E}}(m))(\boldsymbol{P}^{(T)}[m+l] - \hat{\mathbf{E}}(m+l)),$$

where $\mathbf{M}$ is the set of samples collected, l the number of samples to lag, $\boldsymbol{P}^{(T)}$ is the vector of $P^{(T)}(\alpha)$, and $\hat{\mathbf{E}}(\cdot)$ is the unbiased estimator $\frac{1}{M}\sum_{m=1}^{M} P^{(T)}(m)$.

Once the MCMC has terminated, the resource utilization posterior distribution is used to determine the probability that the current FSC satisfies the soft resource constraints η.

4.4 Constraint Satisfaction Improvement of an FSC Policy

Improving the constraint satisfaction of an ϵ-optimal unconstrained policy is solved using a branch and bound (BnB) algorithm variant of [91], a well-known technique for discrete and combinatorial optimization. It is basically a state space search algorithm of rooted tree, where the root has the set of all candidate solutions S. The tree is divided into subsets through a branching function and upper and lower estimated bounds are used in making decisions about "prunning" a branch from further investigation. This pruning uses the estimated bounds to remove branches that are proven not to contain an optimal solution. The underlying goal of the BnB in CA-POMDP is to determine a set of constraint states, our previously stored set of alpha vectors Ω, to "inject" into the optimal controller which utilize fewer resources than the existing machine states in the FSC while trying to minimize the loss in the expected value from ϵ-optimal.

Remark 4.1. By calculating the optimal policy first, we also have an initial optimal policy to recalculate from as resource utilization and observation edge probabilities are learned online (Section 4.5).

The algorithm (Figure 4.6) initializes with the ϵ-optimal FSC policy π^* calculated from policy iteration which is immediately checked for probabilistic constraint satisfaction (Line

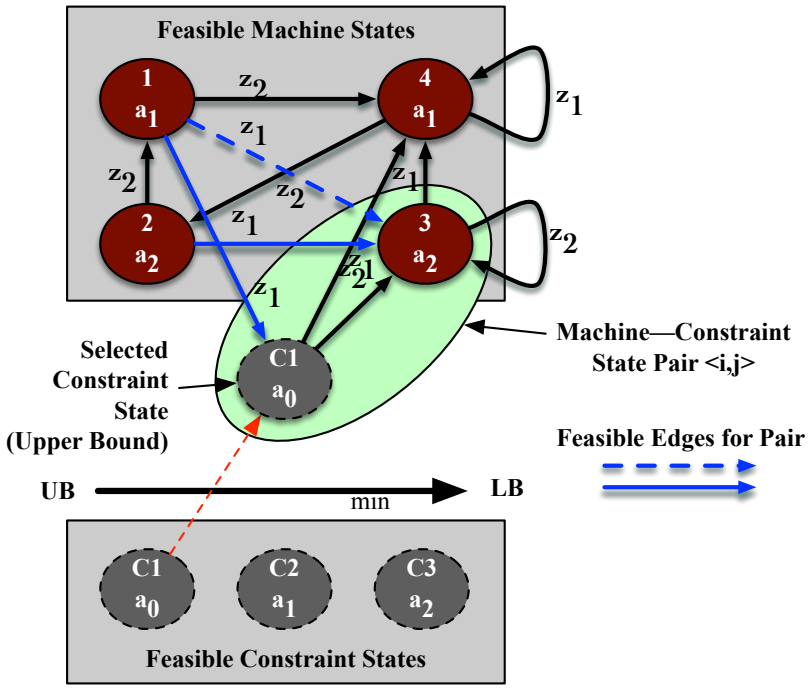


Figure 4.5: An example of the feasible regions for a finite state controller for use in the Branch and Bound Discrete Optimization Algorithm.

3 of Figure 4.6). If the policy is non-compliant, the root of a BnB Tree Q (Line 10) is initialized with the *feasibility space* of modifications that can be made to the unconstrained policy π^*, i.e., an ordered combinatorial set of all of machine states, constraint states, and edge redirection variables, see Figure 4.5.

The first variable in the feasibility region is the set of existing machine states i in the policy $\hat{\pi}_{t-1}$, shown in the upper gray box of Figure 4.5, which is initially π^*. Each machine state's dominant region $V_i b_i^*$, where b_i^* is the region in the belief space b that machine state i has dominance over all other machine states, is used in creating an ordered set of machine – constraint state pairs, marked in Figure 4.5.

The set of alpha vectors Ω saved from the policy iteration algorithm define the second set of variables in the feasibility region. Each alpha vector becomes a potential constraint state j, shown in the lower gray box of Figure 4.5, that can be introduced into the controller

```
1: procedure ConstraintImp(C, U, E, π_0)
2:     π*, V*, Ω ← POMDP Policy Iteration(π_0)
3:     N_0 ← ConstraintEval(π*, C, U, E)
4:     if N_0 ≥ η then
5:         return π*, N_0
6:     end if
7:     π̂_0 ← π*;   t ← 0
8:     repeat
9:         t ← t + 1
10:        Π' ← AllFeasibleRegions(π̂_{t−1}, Ω)
11:        π_min ← min_π π ∈ Π'
12:        g(Π') ← ‖ V*− PolicyEval(π_min) ‖
13:        Q ← Π'
14:        repeat
15:            Π' ← arg min_{q∈Q} g(q)
16:            π_max ← max_π π ∈ Π'
17:            f(Π') ← ‖ V*− PolicyEval(π_max) ‖
18:            N ← ConstraintEval(π_max, C, U, E)
19:            R ← BranchController(π', U)
20:            for R ∈ R do
21:                π_min ← min_π π ∈ R
22:                g(R) ← ‖ V*− PolicyEval(π_min) ‖
23:            end for
24:            Q ← Q ∪ R
25:            Q.Pop(Π')
26:            Prune(Q)
27:        until Q = ∅
28:        π̂_i ← arg max_{π∈Q} g(π)
29:    until π̂_i = π̂_{i−1}
30:    return π̂_i, N(π̂_i)
31: end procedure
```

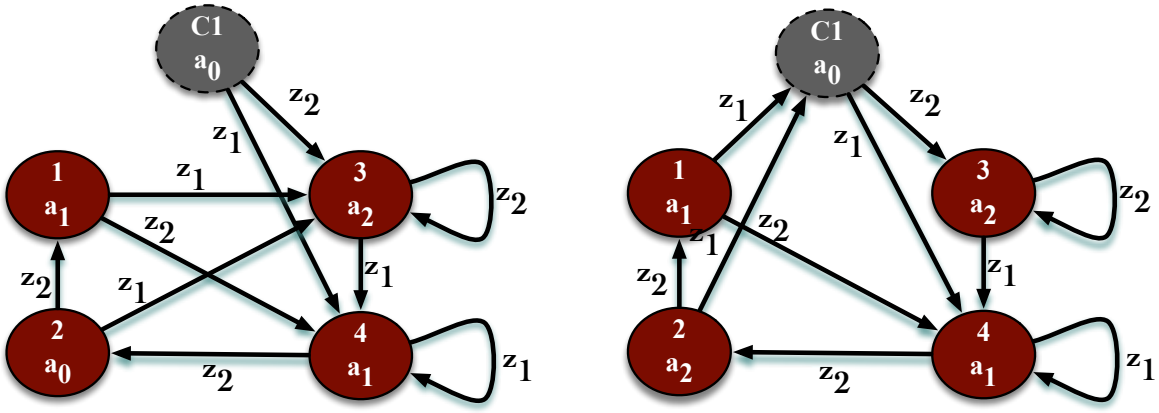
Figure 4.6: Algorithm for performing constraint improvement of an optimal finite state controller.

and will result in a loss of value depending on the value of the new alpha vector and the set of possible edge redirections (the third feasibility region variable). A feasible constraint state j is an alpha vector whose action a_j utilizes less resources $U(a_j) < U(a_i)$ than machine state i. The set of solutions are ordered based on a heuristic involving the loss in value

function that the selection of a constraint state j will have on the overall value. This is calculated mathematically as the distance between the constraint state's alpha vector $V_j b_i^*$ and the machine state's $V_i b_i^*$ for the machine state's dominant region b_i^*, defined by the norm: $\|V_i b_i^* - V_j b_i^*\|$.

The third and final variable in the feasibility region is the set of viable edge redirection probabilities. During the branch and bound, the feasible constraint states for a particular machine state pair $< i, j >$ are injected into the controller and the edges leading to this new constraint state from other machine states already in the controller are selected as either maintaining their transition to the existing machine state i or redirecting to the new constraint state j. The edge redirections are solved for last and until then a simple assumption is made for the calculation of upper and lower bounds for machine–constraint state pairs. The upper bound function assumes that none of the edges are redirected which results in a controller that has no value impact as shown in Figure 4.7a. On the contrary in Figure 4.7b, the lower bound assumes all edges are redirected to the constraint state, which causes the greatest impact on the value function as the original machine state that dominated has been completely substituted or replaced by the constraint state that maximizes $\|V_i b_i - V_j b_i\|$.

The blue lines in Figure 4.5 indicate feasible edges that could be redirected or maintained for the $< 3, C1 >$ machine – constraint pair. The constraint state inherits the outgoing edges of the original machine state i since the outgoing edges already transition to dominant machine states after a belief update, and any constraint state introduced into the controller has already been determined to be non-dominant. Optimally redirecting an edge from another machine state to the new constraint state is not trivial due to the combinatorial explosion of redirection options, which drastically reduces computational efficiency. Instead, we allow the BnB to select from a finite set of the probabilities of redirecting an edge from an original machine state to the new constraint state, i.e., a vector of ordered probabilities



(a) An upper bound edge redirection with no edge redirection.

(b) The lower bound edge redirection with full edge redirection.

Figure 4.7: Upper and lower bound edge redirections heuristics for brand and bound discrete optimization for FSC transformations for CA-POMDP.

$\overrightarrow{P} = [p_0, p_1, \ldots, p_n]$ where $p_0 < p_1 < \cdots < p_n$ and $0 < p_i < 1$. The BnB algorithm searches for the correct probability $p_i \in \overrightarrow{P}$ of edge redirection, yielding a solution that satisfies probabilistic constraints while minimizing the impact to the value of the constrained controller. As the edge redirection probability increases, more edges from an existing machine states k in the controller to the machine state i will be redirected to the constraint state j.

Formally, the objective of the Branch and Bound algorithm of Figure 4.6, is to minimize the impact on the constrained controller value $V(\hat{\pi})$ compared to the optimal controller V^* while ensuring that the probabilistic constraint satisfaction $N(\hat{\pi})$ is greater than or equal to the soft constraints $\eta \in \boldsymbol{C}$. That is,

$$\begin{aligned} \textit{Objective}: &\quad \min_{\hat{\pi}} \|V^* - V(\hat{\pi})\| \\ s.t.: &\quad N(\hat{\pi}) \geq \eta \quad \eta \in \boldsymbol{C}. \end{aligned} \tag{4.5}$$

During each step of the algorithm, a set of feasible solutions Π' is removed from the tree Q based on a priority metric (best-first search) provided by the lower bound function $g(\Pi')$ (Line 12). The branching function (line 18) simply divides the region of feasible solutions

in half, similar to constrained integer programming [92], creating two new nodes Π' in the tree Q until there is no region to divide creating a "leaf" node. The probabilistic constraint satisfaction N of the upper bound policy $\pi_{\max}$ of a node $\Pi' \in Q$ is evaluated via MCMC (Line 17), described in the next section, since we are interested in the best satisfaction the worst controller can provide. The pruning function (Line 19) removes any node in the tree where: (1) the lower bound of that node is greater than the upper bound of any other $g(\Pi'_i) < f(\Pi'_j) \quad \forall (i,j) \in Q_{i \neq j}$; or, (2) the upper bound does not meet the constraint satisfaction $N_i < \eta \quad \forall i \in Q, \eta \in \boldsymbol{C}$.

There are two loops in the constraint improvement algorithm with the inner loop terminating when there are no further nodes to consider in the tree Q (Line 20). The outer loop (Line 22) introduces a single constraint state during each iteration until the desired probabilistic constraint satisfaction has been achieved. Afterwards, a constraint state can be introduced into the finite state machine with an objective to increase the value function without violating the constraint. Once there are no changes to the controller to improve constraint satisfaction or value function, the function returns the new controller.

Figure 4.8 provides an example of a constraint improvement step with the original optimal FSC shown on the left with machine states 1 through 4. Notice that the machine states 1 and 4 have the same action, a_1, which is an indicator that the same action during policy iteration was dominant in two different belief regions. During the constraint improvement, the branch and bound is analyzing the possibility of a constraint state $C1$ concurrent with machine state 3 utilizing action a_0, which utilizes fewer resources than action a_2. $C1$ was initially selected because $C1$ has the greatest value in the same belief region as 3, since in a best-first search it minimizes $\|V_3 b_3 - V_{C1} b_3\|$ and can be seen in the red line of the right graph of Figure 4.8. If $C1$ fails to satisfy the constraints, then the algorithm will examine using $C2$ assuming $C2$ utilizes fewer resources then $C1$.

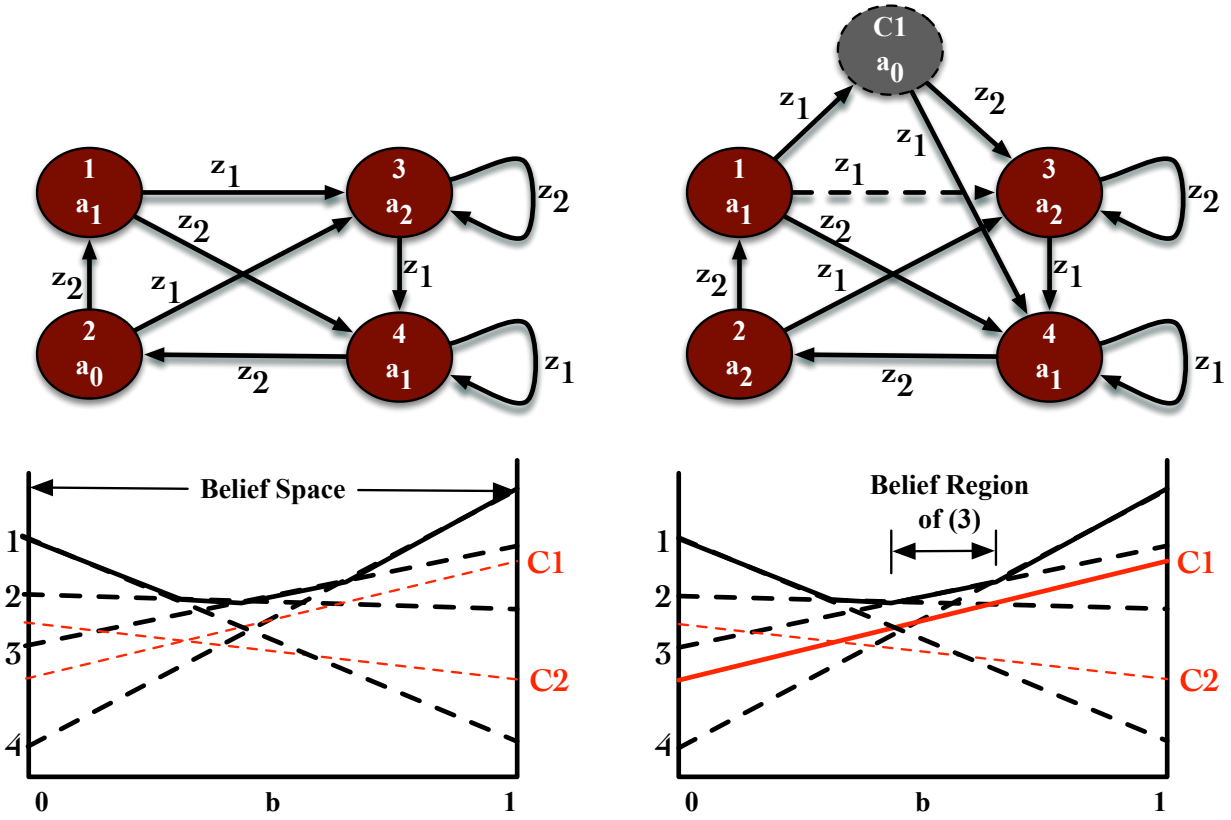


Figure 4.8: The constraint $C1$ has been injected into the controller to reduce the resource utilization of machine state 3. The edge redirection function has selected the $1 \rightarrow 3$ to be reconnected as $1 \rightarrow C1$ and $C1$ inherits the outgoing edges of 3. Notice the loop for observation z_2 at 3 becomes a transition of $C1 \rightarrow 3$

The edge redirection function has selected to redirect the edge $(1, 3)$ to constraint state $C1$ and $C1$ inherits the outgoing edges from machine state 3. It is important to note that the loop at machine state 3 for observation z_2 in the original controller does not create a loop in $C1$ for z_2 but is redirected to the original machine state since 3. This occurs since machine state 3 is the dominant machine state for the belief update of z_2, thus the BnB procedure avoids a non-optimal loop in the constrained controller.

4.5 Learning Edge Probabilities and Resource Utilization

When sampling the controller during the evaluation of constraint satisfaction, an observation edge o is randomly chosen during each of the samples with a probability $p(o|a_i)$ depending on the action a_i of machine state i. This Monte Carlo sampling of observation edges through the controller is initially a uniform *a priori* distribution, but the actual conditional probability of an observation given an action during operation needs to be learned so that the controller can be updated to ensure proper constraint satisfaction as the environment changes. Observations o given an action a are described by a categorical distribution (aka, a generalized Bernoulli or multinoulli distribution), which is used to represent the likelihood of some finite set of possible observations. The conjugate prior of a multinoulli distribution is a Direchlet distribution, which is also a Jeffrey's Prior for an N-sided die with biased probabilities, $\overrightarrow{\gamma} = (\gamma_1, \ldots, \gamma_N)$. The closed form solution for calculating the posterior distribution for the categorical distribution and a Direchlet prior is

$$p(o|A) = \frac{c(o,a) + \alpha(o,a)}{\sum_{j=1}^{|c(j)|} c(o,j) + \sum_{j=1}^{K} \alpha(o,j)}, \tag{4.6}$$

where c is the *a priori* count for an observation o with state action a, α the observed occurrence of observation o, and K the total number of occurrences seen. This method allows a deployed system to track the relationship between actions and observations over time and then recalculate a constrained policy to maintain constraint satisfaction or maximize the value of the controller if it is utilizing fewer resources than expected.

Another consideration in adapting the controller for constraint satisfaction is validating the *a priori* resource utilization models are still applicable during operation. Situations in the

field change over time and the *a priori* models used for resources will not be valid during the entire operational life of the autonomous system. As an example, as the battery of an autonomous system is drained during its operation, an action may consume more battery power than when the battery is fully charged. To track the resource utilization, a simple Bayes estimator with Gaussian priors and likelihood is used to track the current resource utilization per action.

It is unrealistic to continuously update the controller with every edge observation or resource distribution change due to limitations in computational resources and to prevent short-term instabilities or unpredictability in the controller. Platform limitations are the primary driver for the recalculation of a policy from the variation of information, so that a computer is capable of calculating a policy well before the probability of a new policy needs to be recalculated again to prevent any race conditions and processor saturation. However, we still need to adapt online to ensure constraint satisfaction or improve the FSC value. To define an appropriate trigger for recomputing a new controller, the information-theoretic concept of Variation of Information is utilized. When the variation of information

$$d(X,Y) = H(X|Y) + H(Y|X) \tag{4.7}$$

$$H(X|Y) = -\sum_{i,j} p(x_i, y_i) \log \frac{p(x_i, y_i)}{p(y_i)} \; \forall i \in X, j \in Y \tag{4.8}$$

exceeds a desired threshold, the algorithm will recompute a constrained controller from the precomputed optimal controller, where X is either the *a priori* probability distribution of edges $\boldsymbol{E}$ or resource utilization $\boldsymbol{U}$ and Y is the associated probability distribution of the learned distribution. When the learned distributions are significantly different than those previously used to compute a constrained FSC, we recompute the controller with the new learned probability distributions.

Chapter 5

Intelligent Knowledge Distribution

Intelligent Knowledge Distribution (IKD), in this paper, is applying a Constrained-Action POMDP to control communications between multiple independent agents that must stay within quality of service limitations. The goal of IKD is to answer the questions: (i) What information should we send, (ii) when should we send it, and (iii) to whom should we send it to? The other agents within the multi-agent distributed network are the destinations for the information being observed, but to calculate whom to communicate among the other agents grows exponentially, $\frac{1}{2}|A|(|A|-1)$. To reduce the number of communication channels and improve the feasibility of solving for the communication model, the coordination graph used in ND-POMDPs is utilized, which can be different for each asset being tracked by the agents.

The objective of the model is to determine what information needs to be shared with whom at an appropriate time. Therefore, actions are simply to transmit the information learned, $\iota \in I$, to agent $n \in N$ designated as a_ι^n. As the number of agents grows in the system, it is necessary to reduce the complexity of the model by determining an appropriate coordination graph. In a homogeneous system, the coordination graph can be formulated geographically as those operating in the area of the current agent which can assist or needs to know information. On the contrary in heterogeneous systems, the coordination graph can focus more on collaborative capabilities and interoperability.

The state of the system is the discrete variable that another agent has relevant and timely

information that an agent has obtained, which is not directly observable. Therefore the belief state $b(s)$ is a probability distribution that indicates the confidence that an agent has that a collaborating agent $n_{-i} \in \mathcal{N}_i$ is "up to date" with the relevant information to the overall objectives of the system.

The controller begins with an initial belief state, b_0, which is used to determine the starting point in the controller. The associated action that maximizes the $\arg\max_a b_0 V^\pi$ is taken and the environment observed. The observation drives the transition from an action to the next action indicated as index numbers on the edges.

5.1 CA-POMDP Model Formulation for Intelligent Knowledge Distribution

5.1.1 CA-POMDP Action & State Model for IKD

The IKD Model builds on the CA-POMDP formulation by extending the CA-POMDP tuple to include $< \mathbf{\Sigma}_i, \mathcal{I}, \mathcal{N}_i, \boldsymbol{F} >$, where we assume each agent independently runs CA-POMDP. Each agent i has a set of neighbors $\mathcal{N}_i$ in the environment that it can actively collaborate with based upon proximity, abilities, et cetera. $\mathbf{\Sigma}_i$ is the alphabet of communications $\sigma_k \in \mathbf{\Sigma}_i$ of information elements k that agent i can transmit to a neighbor $j \in \mathcal{N}_i$. $\boldsymbol{F}$ is a set of states $s \in \boldsymbol{S}$ that place mission objectives at risk of failure, which we will use to drive information relevance. We define the set of actions $\boldsymbol{A}_i$ for an agent i that indicate the decision to transmit the kth element of information to agent $j \in \mathcal{N}_i$, denoted as action a_k^j related to the information σ_k:

$$\mathbf{A_i} = \left\{ \cup_{\sigma_k \in \mathbf{\Sigma}_i, j \in \mathcal{N}_i,} \quad a_k^j \rightarrow \{0,1\} \right\} \cup \varnothing \tag{5.1}$$

along with the single $\varnothing$ (or Silence) action to not transmit at all.

The states of the IKD model are defined by levels of relevance S_r of locally observed information for each agent i (e.g., $S_r = \{LOW, MEDIUM, HIGH\}$), and levels of collaboration S_c for each agent i with its neighbors $j \in \mathcal{N}_i$ (e.g., $S_c = \{LOW, MEDIUM, HIGH\}$). Relevance is described by a set of discrete states that indicate how important information is to global mission objectives, and is calculated per-agent based on local observations only. The confidence of an agent in its current collaboration with its neighbors $j \in \mathcal{N}_i$ is a set of discrete states indicating a level of confidence that the current level of communication will maintain the ability for the agents to achieve global mission objectives by sharing local observations. Formally, the set of states for an agent i in the IKD model is:

$$\mathbf{S_i} = \{s_r^{k=1}, \ldots, s_r^{k=|\Sigma_i|}, s_c^{j=1}, \ldots, s_c^{j=|\mathcal{N}_i|}\} \tag{5.2}$$

with relevance states $s_r^k \in S_r$ for each of information element k and a set of collaboration states $s_c^j \in S_c$ with a neighbor $j \in \mathcal{N}_i$ indicating whether the neighbor is "up to date" with relevant information. The states in an agent's state set therefore becomes a representation of how important information is and how timely the agent's information is for neighboring agents.

5.1.2 CA-POMDP Reward Model for IKD

The reward functions for IKD need to be defined specifically by the model designer or learned online; what follows are simply basic guidelines for reward formulation. The reward for relevance $\hat{R}_r(s, a)$ is a normalized reward based upon the product of 1. the likelihood $\mathcal{L}$ that the next state s' is not a critical state $f \in \boldsymbol{F}$ where global mission objectives are at risk when taking action a in state s, which is either known *a priori* or learned online, and 2. the

information-theoretic metric on the maximal value of the information $\rho(k)$ an agent could convey to a neighbor:

$$\hat{R}_r(s,a) = \mathcal{L}(s' \notin \boldsymbol{F}|a,s) \max_{\sigma_k \in \Sigma_i} \rho(k). \tag{5.3}$$

The basic concept is to increase the reward for communications when the mission objectives are at risk and the value of the local information is high.

Remark 5.1. The utilization of a reward function with the components ρ and $\mathcal{L}$ by its nomenclature appears similar to belief-dependent rewards [28], however we argue here why this is not the case. The reward functions are not tied to a state belief but instead are calculated for each state of the IKD model and assume a linear function between states, as seen with any belief-state MDP [6, 14]. The reward functions can be learned online and the FSC policy recalculated but they will remain piecewise linear and convex within the hyperplane of belief space and do not change as the belief does.

Remark 5.2. Defining an information theoretic metric for ρ can be achieved via multiple methods. In our experimental model we use an information matrix based upon sensor deviations κ_a for $\rho(k) = 1/\kappa_a$, but mutual information is a very common approach in the literature often utilizing KL-Divergence.

Collaboration is a normalized reward $\hat{R}_c(a,s)$ based upon the product of the proximity of a quantized state to approaching the heuristic bound where collaboration will diverge and become unbounded $\mathcal{Q}(\cdot)$, as with intermittent communication in controls with a Kalman filter's estimation error covariance [93], and the maximal value of information that can be shared with a particular neighbor:

$$\hat{R}_c(a,s) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathcal{S}(\mathcal{Q}(\lambda_j|a,s)) \cdot \max_{\sigma_k \in \Sigma_i} \rho(k), \tag{5.4}$$

where $\mathcal{S}(\cdot)$ is the sigmoid function, $\mathcal{Q}(\lambda_j|a,s)$ is the probability that the communication action a with neighbor $j \in \mathcal{N}$ will approach the proximity of an unbounded heuristic $\lambda \in \mathbf{\Lambda}$, $\mathbf{\Lambda}$ is the limits of the upper $\bar{\lambda}$ and lower $\underline{\lambda}$ bound heuristic of a critical probability λ_c that is dependent on the model (See Section 6.3.1), and $\max_{\sigma_k \in \Sigma_i} \rho(k)$ is the best value of information $\rho(k)$ that an agent $j \in \mathcal{N}$ could receive from i. Thus, as the lack of communications drives a global objective towards an unbounded heuristic (e.g., a diverging Kalman filter estimate due to intermittent communications) and the value of information for neighboring agents increases, the reward for communication increases. The final reward $R(s,a)$ is the sum of the collaboration and relevance rewards with one exception. If the action is not to transmit ($a = \varnothing$), the reward is zero for all states:

$$R(s,a) = \begin{cases} \hat{R}_r + \hat{R}_c & if\ a \neq \varnothing \\ 0 & if\ a = \varnothing \end{cases}. \tag{5.5}$$

This at first appears counter intuitive, because the action to not transmit will be dominated by all other actions which is intentional. If the agent stops transmitting then it always is going to lead to information loss and therefore provides no benefit not to continuously transmit. It is the constraints to preserve resources that will introduce a do not transmit action or one that utilizes fewer resources into the controller as a viable constraint state to satisfy soft resource constraints.

5.1.3 CA-POMDP Transitions & Observations

As with the rewards model, transitions and observations are divided into relevance and collaboration components. Transition probabilities from one state s to another state s' based upon an action a are assumed to be independent to reduce computational complexity

and therefore are the product of the independent transitions of the components

$$T(a, s, s') = \prod_{i=1}^{n} \prod_{\sigma_k \in \Sigma_i} T(a, s_r^k, s_r^{k'}) \cdot \prod_{j \in N_i} T(a, s_c^j, s_c^{j'}) \tag{5.6}$$

where T is the transition matrix, and s_r and s_c are the relevance and collaboration state components. The component transition functions $T(a, s_r, s'_r)$ and $T(a, s_c, s'_c)$ are determined by the knowledge being shared and are discussed for our use case in Section **??**.

The observation probabilities are similarly constructed as the transition probabilities with relevance and collaboration components as follows:

$$O(a, s, o) = O(a, s_r, o_r) \cdot \prod_{j \in \mathcal{N}_i} O(a, s_c^j, o_c^j) \tag{5.7}$$

where O the observation matrix, and o_r and o_c are the relevance and collaboration components of observations. [1]

Remark 5.3. Though the relevance of the information and the collaboration between agents are conditionally dependent in reality, the formulation of the model as a decentralized POMDP reduces the dependence with the focus on local observations allowing to validate the approach before addressing a conditional dependence transition and observation model.

As discussed in related work, the observation model is key in restructuring the POMDP to avoid belief-dependent rewards and its associated solution methods. In the construction of an IKD model, an observation model should be designed to provide the ability to map observations to the appropriate belief space for relevance and collaboration. Any mapping υ from observations in the environment to an observation state is the mapping of a continuous observation space to discrete observations through cluster-based techniques, such as

[1] In future work, this assumption will also be relaxed as with the transition assumption of independence.

K-medoids [94] or DBScan. See Section 5.3.1 for spatial clustering techniques and Section **??** for how an observation model was constructed to address IKD with asset localization and Kalman Filters.

5.1.4 Value of Information

There are many techniques available to determine the *value of information* that is being observed by an agent to inform the intelligent knowledge distribution communication model. Some information models come with their own heuristics on the value of information like research in Kalman filters looking at the probability of arrival of missing observations will cause the filter to diverge and become unrecoverable [93]. In this section, we cover a broad information theoretic approach for evaluating the value of information.

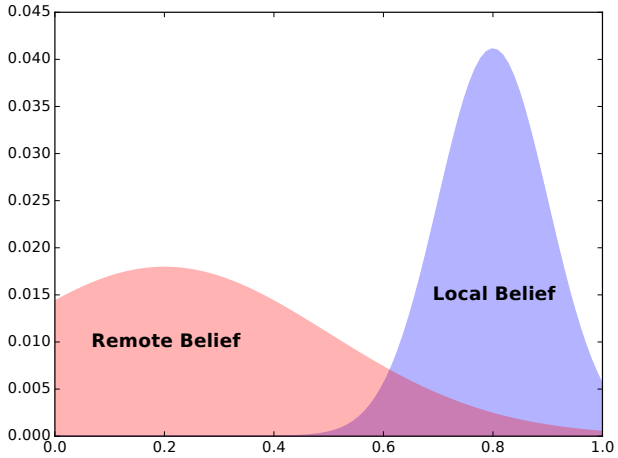


Figure 5.1: Comparison between the value of information of the local agent's belief state (blue) to the belief state of one of the collaborative agents (red).

The rewards for the POMDP in an information theoretic setting are determined by calculating the value of the information based upon the known belief of information to the agent in comparison to what the agent believes the information will improve the belief state of a neighboring agent. The relative entropy metric provides the framework for determining

the information theoretic information gain between the expected entropy of another agent and the actual entropy of the local agent. Relative entropy [95, 96], using Kullback-Liebler divergence

$$D_{KL}(P||Q) = \sum_{x} p(x) \log \frac{p(x)}{q(x)} \geq 0, \tag{5.8}$$

is based on the distribution of the agent's beliefs, which can easily be calculated from the agent's existing belief state on the target and its estimated belief state of the another agent's belief state.

The KL Divergence is dependent on $|\mathcal{N}|$ probability distributions for each local agent, which required a time dependent belief update mechanism for both the local agents belief in the information it has and in predicting the belief states of the other agents. As the local agent receives new information about an item, it can update its own belief states depending on the noise and accuracy of the sensor input. At the same time, it updates the belief state of the other agents [6]

$$b'(s') = \frac{p(z|s') \sum_s p(s'|a,s) b(s)}{\sum_{s,s''} p(z|s'') p(s''|a,s) b(s)} \tag{5.9}$$

as it performs a *NoOp* action without any observation to improve accuracy (e.g, $\mathcal{N}(\mu, \sigma)$ for a Kalman Filter).

For a multiagent system and their belief states, Figure 5.1 shows two distributions with the belief of the local agent and its estimated belief of another neighbor agent in the coordination graph. To determine the reward for communicating what the local agent knows, it needs to understand what information the other agent would gain from receiving an update. Kullback-

Leibler Divergence, D_{KL}, provides the ability to determine the information gain

$$V_\iota = D_{KL}(b'_0(s) \parallel b'_n(s)) \tag{5.10}$$

$$= \sum_{s \in S} b'_0(s) \log \frac{b'_0(s)}{b'_n(s)} \tag{5.11}$$

obtained from the local agent sharing its local beliefs, b'_0, in comparison to what it believes the other agent contains, b'_n. Many times information theoretic comparisons can be computed offline and be available *a priori* to the model. This allows for rewards to be defined ahead of schedule for the state space and utilized in an observation model.

5.1.5 Applicable Constraints

The constraints placed on the actions are to prevent an impact on the quality of service of the sensor on its limited constraints. Every action to share information with another agent to improve collaboration has significant impacts on the limited resources available to sensor nodes. This can include insuring the data traversing the wireless communication system minimizes its utilization of bandwidth or minimizes the use of battery power to maximize the operation time of autonomous vehicles. Common constraints under consideration in performing Intelligent Knowledge Distribution to maintain constraint satisfaction include:

- **Power** – The power consumed to communicate collaboration information between agents should not exceed a defined amount of the battery to ensure the power is available for flight operations, computational processing, and sensor operation.
- **QoS** – The bandwidth consumed to communicate collaboration information between agents should not exceed a specified percentage of the available bandwidth to ensure key sensor information is being fed to operators.

- **CPU Time** – The amount of processing capability being used to prepare and transmit the data is not consuming processing power needed to collect and analyze the sensor data.

- **Memory** – The amount of memory being utilized to store useful information is not exceeding a predefined memory allocation. This particular constraint is contrary on the others in that previous active collaborative actions consumed resources, but in this case the act of not collaborating will consume memory resources to maintain the information until is believed it is needed.

5.2 Concurrent Decentralized POMDP

The Concurrent Decentralized (CoDec) POMDP architecture consists of four concurrent decision-making processes that provide an agnostic framework for the key components of a fully autonomous multi-agent system. At the most rudimentary level, the individual allocation model handles the task or action that has been assigned to the agent, as with motion planning for unmanned vehicles. The Task Allocation algorithm is responsible for evaluating which task is the most appropriate for the agent with respect to its capabilities and those of the other agents. In many scenarios, there is an independent agent that the system is monitoring, e.g. first responders, survivors, victims, and other drones, which is the purpose of the asset monitoring function to learn and track this behavior so that the individual MDP and the Task Allocation MDP are able to account for it. The final concurrent algorithm is the key component for Intelligent Knowledge Distribution (IKD), which decides on what information from task allocation and asset tracking should be sent to whom and when. The CoDec POMDP model is a convergence of the POMDP and CMDP models and consists of

the tuple

$$< \mathbf{S}, \mathbf{O}, \mathbf{B}, \mathbf{A}, \mathbf{T}, \mathbf{R}, \mathbf{\Pi}, Q_t, \mathbf{d}, \mathcal{B}, \mathcal{G} >$$

where **S** are the states available but not observable, **O** is the observation model that ties observations to beliefs, **B** is the belief states probabilities across states, **A** is the set of actions, **T** is the transition probabilities, **R** is the rewards, **Π** is the set of feasible policy trees π, Q_t is a matrix of decision variables, $\mathcal{B}$ is the belief model that maps concurrent MDP model information to CoDec belief states b, $\mathcal{G}$ is the coordination graph, and **d** is the upper bounds on the state and transition probabilities.

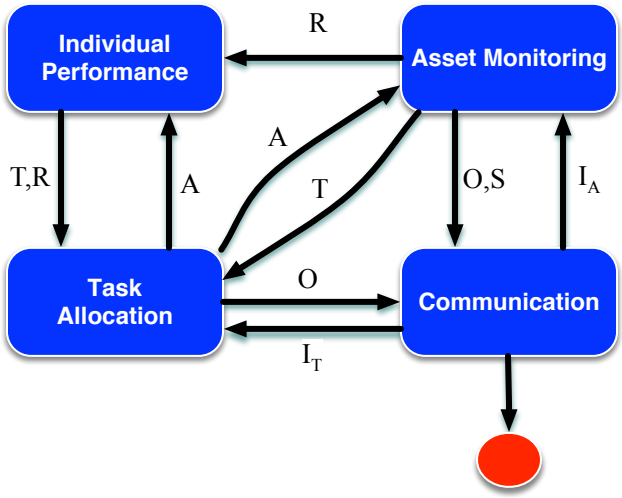


Figure 5.2: A generic Concurrent Decentralized MDP interdependencies between the Individual Performance, Task Allocation, Asset Monitoring, and Communication decision engines.

The Individual MDP needs to understand the actions and states available to it from the Task Allocation MDP to properly determine the transitions and rewards for that particular tasking along with the behavioral models from the asset monitoring, which is learning the behavior. The Asset Monitoring informs the Individual MDP into the probabilities and probabilistic rewards based upon its behavior, for example: (i) Is the asset moving to fast or slow for the agent? (ii) Is the asset moving away from the agent? (iii) Is the asset too maneuverable for the agent to maintain monitoring?

The rewards and transitions from each one of the associated tasks drive the Task Allocation to select an appropriate assignment for the current state of autonomous system, e.g., is it close to one of the assets or not. The task Allocation is also driven by all the current or belief of the task states of the other agents. It also feeds the current tasking to the Asset Monitoring to start a new model or switch to an existing stored model.

The Communication CA-POMDP determines what information needs to be sent to whom and when. The information it receives from Task Allocation can consist of the current tasking and the value functions, but the transmission of just observations can reduce the state space explosion for performing all the various available tasks. It also needs to determine if the information being learned about the asset needs to be transmitted to the other agents. Information about the different assets will assist those assets in determining their ability to track and monitor that particular asset.

5.2.1 Agent Control Concurrency Interaction Models

First, we address the interaction of the non-communication components of the CoDec POMDP originally envisioned in [13], who originally formalized the interaction between the task allocation and the Individual MDP. We expand the CMDP paper work of Girard [13] to include the concurrent MDPs for Asset Monitoring and Communications. The communication interaction between the concurrent MDPs and the Communication MDP is covered in the next section because its relationship with the other concurrent MDPs is different than in the standard CMDP construction utilized here. The Individual Performance, Task Allocation, and Asset Monitoring algorithms are dependent on each other as concurrent MDPs where:

- The *Individual Performance* MDP needs to know the possible task or asset that could been assigned to it to determine the rewards for performing the task from information

provided by the Asset Monitoring;

- The *Task Allocation* MDP on the other hands needs to be aware of the individual agents ability and capabilities to perform an assigned task from Individual Performance and the likelihood of succeeding from Asset Monitoring; and,
- The *Asset Tracking* MDP is responsible for understanding the behavior of the agents being tracked, wether an enemy drone, human team members, or rescue victims, and providing the statistical information needed to the other components to determine performance.

The Individual Performance MDP needs to understand two primary things: (i) the actions from the Task Allocation to calculate the rewards associated for that particular task along with (ii) the behavioral models from Asset Monitoring, which is learning the asset's behavior including transitions, predicted states, et cetera. The Asset Monitoring informs the Individual MDP into the probabilities and probabilistic rewards based upon its behavior.

The Individual Performance relies on the Task Allocation in that for every state $s_i \in S_i$ there is assumed to be a subset $e_{IT} \in s_i$ whose evidence value e is dependent on Task Allocation. For Individual Performance that involve motion planning, as in the Disaster Monitoring experiment in Section 6, the Individual Performance provides Task Allocation an understanding of the costs or rewards R_T^* in monitoring a particular asset that it might be assigned to. It is also dependent on Asset Monitoring for the reward function $R_{TA}(s_{TA}, a_t, s'_t)$ that is a probability distribution determined by a predictive algorithm, like Kalman filters or deep neural networks (DNN). The Individual MDP is therefore assigned a sub-task $t \in T_{TA}$ from Task Allocation which defines a subset of an MDP model that the agent progresses towards defined by the tuple $< S_t, A_t, T_t, R_t >$.

The rewards and transitions from each one of the associated tasks drive Task Allocation to

select an appropriate task $t \in T_{TA}$ for the state s_i (MDP) or current belief b_i (POMDP) of the autonomous system, e.g., is it close to one of the assets. The Task Allocation is also driven by all the current states or beliefs of the tasks of the other agents $i \in \mathcal{N}$. It also feeds the current tasking to the Asset Monitoring to start a new model or switch to an existing stored model. Conversely, the Task Allocation MDP is dependent on the reward function $R_{IP}(s_{IP}, a_t, s'_t)$ and the transition function $T_{IP}(s_{IP}, a_t, s'_t)$ of the Individual Performance of an agent for an action $a \in A_i \in \boldsymbol{A}$. The Asset Monitoring MDP provides information on how well an individual agents is capable of performing its job against an asset it is responsible for through statistical information on an assigned tasks uncertainty.

Asset Monitoring is an online learning or state predictive algorithm responsible for the construction of a model of the asset being monitored. This asset can be either an enemy or friendly drone, a human team member that the drone is meant to support in the field, or a victim that is the target for search and rescue. The transition probabilities of the asset or target and the predicted reward mechanism driving its behavior informs the Task Allocation of the uncertainty T_{AM} over a finite horizon and Individual Performance as to the ability to maintain awareness T_{IP} and the reward R^*_T for doing so. In the Disaster Monitoring experiment of Section 6, this informs the agent on how well its is able to track and maintain observation over an asset that might have varying speed and rotational capabilities.

5.2.2 Agent Communication Concurrency Models

The Communication CA-POMDP determines what information needs to be sent to whom and when. The previous section covered the non-communicating components of the concurrent model, which have a different relationship between each other than they do with the Communication MDP, because the other MDPs are not dependent on the communication

model for solving optimal policies but instead it updates their current observations $o \in O$ or states $s \in S$. The Communication MDP's model is dependent on the data the other components wish to transmit and the number of neighboring agents $i \in \mathcal{N}$ that are a part of the coordination graph $[G]$, but its formulations is independent of the behavior of the other three.

The information it receives from Task Allocation is on the current tasking and the value functions for performing all the various available tasks. It also needs to determine if the information being learned about the asset needs to be transmitted to the other agents. Information about the different assets will assist those assets in determining their ability to track and monitor that particular asset.

Task Allocation and Communications are bound in ensuring that communications are maintained to ensure mission objectives are being met. Section 6.4 shows how the communication model was constructed to provide the agnostic intelligent communications. The data being transmitted between multiple agents from each of the concurrent MDPs drives the action, state, transition, observation space of the CA-POMDP model used for intelligent communications.

For multi-agent systems, the collaboration of assignments between agents to efficiently meet overall mission objectives is the responsibility of task allocation, sometimes referred to as mission orchestration or management. The task allocation decision process τ can have multiple pieces of data $\iota^\tau \in \mathcal{I}$ that needs to be "registered" with the communications component which can consist of the current task allocation that the agent has been assigned a_t^τ, the task allocation state s_t^τ (MDP) or belief b_t^τ (POMDP), observations o_t^τ, action-observation histories, and the estimated reward R_t for that task at time t. These registered data messages of σ^τ of the information ι^τ makes up part of the alphabet of information Σ^τ.

Asset monitoring α is driven by maintaining bounded estimates of the observations of an asset $k \in K$. These observations can lead to tracking the state s^{α_k} as would be done with state estimation algorithms like Kalman filters, utilized in the experiment covered in Section 6.3.2. It can also consist of communicating learned parameters of the asset as in transition probabilities $T_t^{\alpha_k}$ and learned Q values $Q_t^{\alpha_k}$ at time t.

The Individual MDP maintains the current state of the drone itself and therefore that information could be beneficial for other agents in determining an optimal policy. In more complex scenarios where there is desire for transfer learning in improving the execution of tasks that others may not have been exposed to, that information may also be registered with communications model. Also, information relevant from the Individual MDP is often incorporated into the task allocation MDP and can be communicating from that model in lieu of a connection from the individual MDP.

5.3 Learning Multi-agent Collaboration Strategies

Learning the collaboration strategies of the agents involved in the multi-agent system will use probabilistic graphical models to learn these collaboration strategies. The collaboration strategies can depend on many system and environmental factors. In homogeneous systems, the geographical location or clustering of agents could determine their collaboration or the communication channel. It is the goal to concentrate more on heterogenous systems where the collaboration strategies will depend not only the location information, but also on tasking and services. As an example, a drone with complex infrared thermal imaging could work in collaboration with a drone with precision chemical sensors to locate survivors for rescue personnel.

Initial development into this area will start with statistical matching algorithms before ex-

amining more complex models including:

1. Markov Network or Markov Random Field, which is a undirected cyclic graph
2. Conditional Random Fields, which are a subset of Markov Random Fields where random variables may also be conditioned upon a set of global observations.

A coordination graph (CG) is used to graphically represent the coordination requirements necessary to achieve a joint optimization with minimal coordination between agents. Figure 5.3 shows an example coordination graph with a joint utility function. The goal of a CG is to identify an unsupervised clustering strategy that maximizes a joint utility while minimizing computational complexity. There are three techniques that have been identified that meet various circumstances that are worth covering for Intelligent Knowledge Distribution:

1. **Spatial Clustering** – In spatial clustering, the designer has determined that grouping assets for collaboration in the field is based upon geographical location $[x, y]$. This is often seen in proximity operations, like drone swarms;
2. **Relational Learning** – An ontological approach to coordination that learns relationships between data types to develop semantic knowledge of the data the agents are communicating between each other; and,
3. **Factored MDPs** – A more complex Q-Learning approach to coordination based on probabilistic graphical models (PGM), which are learned online.

5.3.1 Spatial Clustering

Spatial clustering is a group of unsupervised machine learning techniques that group types by their proximity within a feature space, in our case geographical location $[x, y]$. There

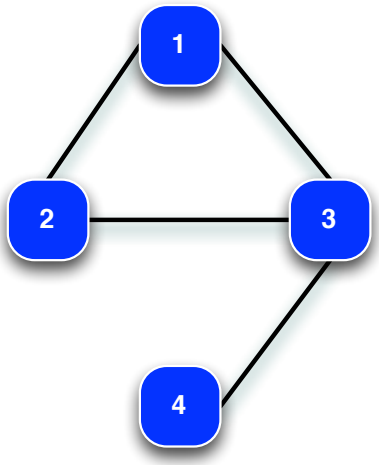


Figure 5.3: Coordination graph used in networked distributed POMDP (ND-POMDP) to factor the reward dependence between agents and reduce combinatorial explosion with multi-agent coordination.

are two primary algorithms that are easily available to perform spatial clustering: K-means clustering [97] and Density-based spatial clustering of applications with noise (DBSCAN) [98]

The first, K-means clustering, is a popular data clustering algorithm that partitions the data space into Voronoi cells. The objective

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in S_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2 \tag{5.12}$$

is to find a partition of n observations into sets $\boldsymbol{S} = \{S_1, S_2, \ldots S_k\}$ as shown in Figure 5.4. This is often performed through an iterative refinement approach that alternates between two steps: assignment and update. In the assignment step, the data is assigned to a centroid, randomly initiated, that has the closest least-squared Euclidean distance. The centroids are recalculated based upon the geometric mean of the data assigned to the current centroid

cluster

$$m_i^{t+1} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j. \tag{5.13}$$

The algorithm terminates when there has been no change to the centroids of each cluster. The K-means algorithm is not guaranteed to converge on an optimal solution and therefore initial states and better Euclidean solutions can be achieved through k-medians [99] or k-medoids clustering [100].

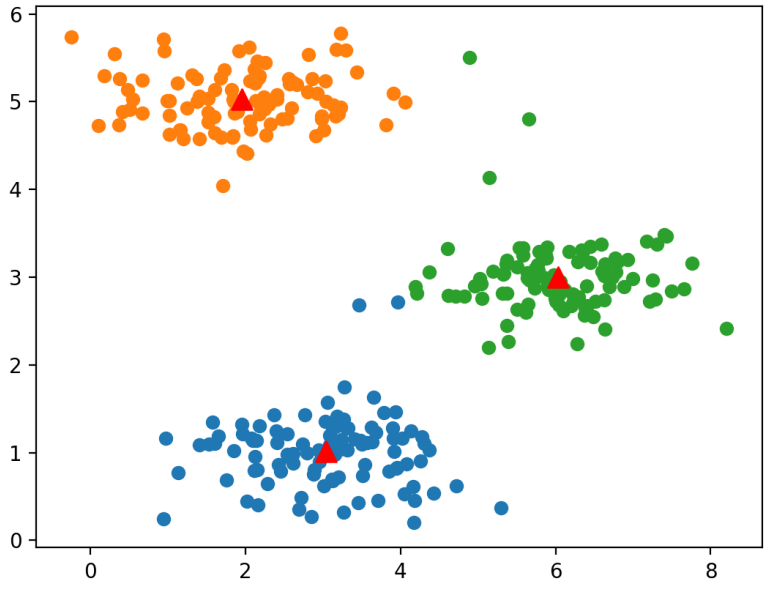


Figure 5.4: Clustering performed on an example data set through the K-means algorithm (K=3).

DBSCAN is another popular clustering algorithms in the scientific community that clusters data that are closely packed together, density-based. Some of the advantages of DBSCAN over K-means are the number of clusters, K, does not need to be defined *a priori* and that it can find arbitrary shaped clusters, as shown in Figure 5.5. The algorithm can be summarized

in three steps [101]:

1. Find the points in the ϵ neighborhood of every point, and identify the core points with more than `minPts` neighbors.

2. Find the connected components of core points on the neighbor graph, ignoring all non-core points.

3. Assign each non-core point to a nearby cluster if the cluster is an ϵ neighbor, otherwise assign it to noise.

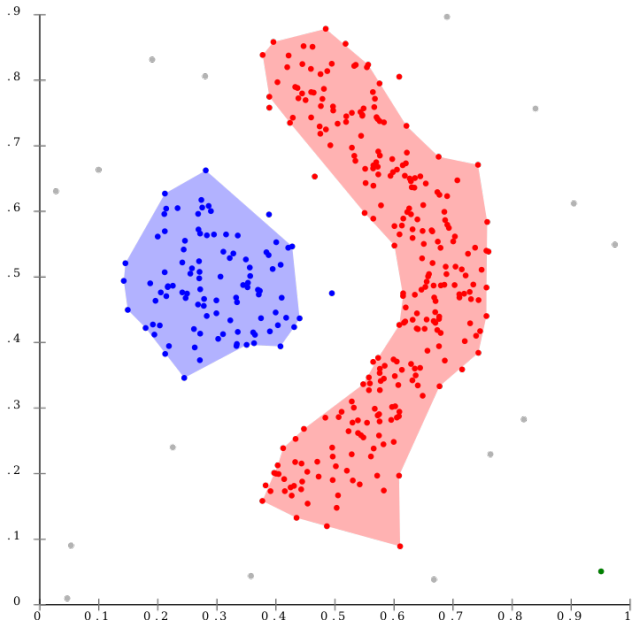


Figure 5.5: Clustering performed on an example data set through the DBSCAN algorithm [8].

5.3.2 Relational Learning

Another key technique for determination coordination graphs is to understand the semantic knowledge that needs to be exchanged between the agents. Semantic knowledge can be

expressed a system of relations, which cognitive scientists and AI researchers attempt to discover or learn utilizing "domain theories" [102]. We are interested in the kinds of entities that exist in each set, and the possible or likely relationships between them. Infinite Relational Model (IRM) is a framework to learn structure in relational data by partitioning each type into clusters and using those partitions to infer relationships between them [103]. For example, the authors in [103] have a single type *people* and multiple relationships defined over the same domain, referred to as a predicate, that consists of $people \times people$: $likes(\cdot,\cdot)$, $admires(\cdot,\cdot)$, $respects(\cdot,\cdot)$, and $hates(\cdot,\cdot)$ defined as a ternary relation $applies(i,j,p)$ which is true if predicate p applies to the pair (i,j).

IRM is interested in the posterior distribution $P(R^1,\ldots,R^m,z^1,\ldots,z^n)$ of the generative model for the relations and the cluster assignments

$$P(R^1,\ldots,R^m,z^1,\ldots,z^n)=\prod_{i=1}^{m}P(R^i|z^1,\ldots,z^n)\prod_{j=1}^{n}P(z^j), \tag{5.14}$$

where the observed data has m relations involving n types, R^i is the ith relation, T^j is the jth type, and z^j is a vector of cluster assignments for T^j.

The prior $P(z^j)$ is based on a nonparametric Bayesian model [104] with a distribution over partitions set by the Chinese Restaurant Process (CRP) [105] so that IRM can discover the number of clusters in type T. With CRP, a new member is attracted to an existing cluster in proposition to its size

$$P(z_i=a|z_1,\ldots,z_{i-1})=\begin{cases}\frac{n_a}{i-1+\lambda} & n_a>0\\ \frac{\lambda}{i-1+\lambda} & a\ is\ a\ new\ cluster\end{cases} \tag{5.15}$$

where n_a is the number of objects already assigned to cluster a, and λ is a parameter. The

complete generative model for generating relations from the clusters then becomes

$$z|\lambda \sim CRP(\lambda) \tag{5.16}$$

$$\eta(a,b)|\beta \sim Beta(\beta,\beta) \tag{5.17}$$

$$R(i,j)|z,\eta \sim Bernoulli(\eta(z_i,z_j)) \tag{5.18}$$

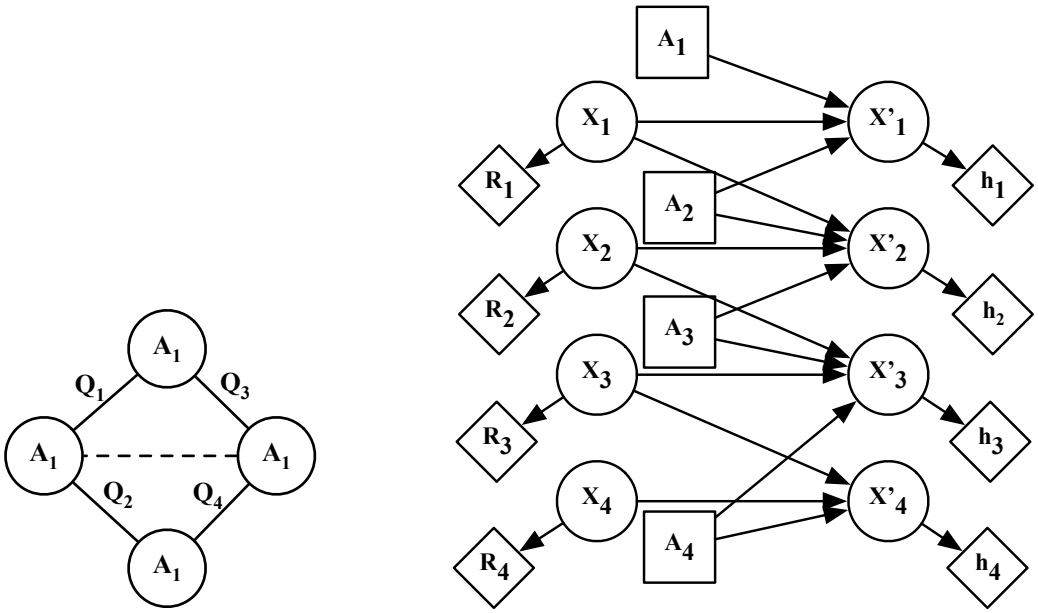
MCMC inference to sample the posterior on cluster assignments $P(z|R) \propto P(R|z)P(z)$ [106] to find the best partition of z by repeatedly running a hill climbing algorithm from an initial configuration. To represent the system or relations graphically for use a coordination graph, we utilize η to identify category pairs that are strongly linked and the predicates that connect them [107] to create a CG representation.

5.3.3 Factored MDPs

Factored MDPs [108] allow for large state or action spaces in an MDP to be represented as a dynamic Bayesian Network (DBN) to address multi-agent cooperative action selection [109]. The objective is to select a joint action $\boldsymbol{a}$ that maximizes $\sum_j Q_j(\boldsymbol{a})$, where Q_j is the local utility function of agent j.

$$\begin{aligned} Q^* = \max_{a_1,a_2,a_3,a_4} [&Q_1(a_1,a_2) + Q_2(a_1,a_3) \\ &+Q_3(a_2,a_3) + Q_4(a_3,a_4)]. \end{aligned} \tag{5.19}$$

The strategy for determining the optimal joint action $a^* = \arg\max_{\boldsymbol{a}} Q(\boldsymbol{a})$ is solved either using non-serial dynamic programming [110] or through variable elimination [86], similar to variable elimination in Bayesian networks, which is more commonly seen in current literature.



(a) Coordination Graph (b) Deep Belief Network (DBN)

Figure 5.6: Coordination graph and DBN for a 4-agent problem.

The reader should reference Kok [111] for implementing the max-plus algorithm for multi-agent decision making to solve the variable elimination problem.

If the coordination graph can not be determined *a priori*, then the coordination dependencies need to be learned online. The coordination behavior can be learned through a Q-learning variant for CG [112] or by statistical analysis [113]. The Utile Coordination of [113] provides a statistical analysis technique that is based on actions and states that learns where the agents need to coordinate. To determine the benefits of coordination, at each state s a sample of the combined return $\hat{Q}_I(s, a_I)$ across the set I of agents after a joint action a is maintained

$$\begin{aligned} \hat{Q}_I(s, a_I) &= \sum_{i \in I} \bar{Q}_i(s, a) \\ &= \sum_{i \in I} [R_i(s, a) + \gamma Q_i(s', a^*)] \end{aligned} \tag{5.20}$$

where $Q_i(s', a^*)$ is their individual contribution to the maximum global Q-value. The expected combined return is estimated by computing the mean $\hat{Q}_I(s, a_I)$ over the last M samples. A statistical t-test [114]

$$t = \frac{\max_{a_I} \bar{Q}_I(s, a_I) - \bar{Q}_I(s, a_I^*)}{\sqrt{(2/M)((M-1)\sigma_{\max}^2 + (M-1)\sigma_*^2)/(2M-2)}} \tag{5.21}$$

measures wether the largest expected combined return $\max_{a_I} \bar{Q}_I(s, a_I)$ with variance $\sigma_{\max}^2$ differs significantly from the expected combined return $\bar{Q}_I(s, a_I^*)$ with variance σ_*^2 when performing the greedy joint action a_I^* with $(2M-2)$ degrees of freedom. The null hypothesis assumes the two groups are equal and the level of significance p is the probability of rejecting that hypothesis. If there is a statistically significant difference ($p < P$) and also sufficiently large ($d > D$) as determined by a standard effect size measure [114]

$$d = \frac{\max_{a_I} \bar{Q}_I(s, a_I) - \bar{Q}_I(s, a_I^*)}{r_{\max} - r_{\min}}, \tag{5.22}$$

then there is a significant benefit of the agents coordinating actions during that state.

Chapter 6

Experiment & Results

The use case for validating the Concurrent Decentralized (CoDec) MDP and the Constrained-Action POMDP (CA-POMDP) for *Intelligent Knowledge Distribution* (IKD) was the aerial monitoring of ground assets during a disaster response to ensure they safely avoid hazards and dangerous situations they may not be aware of from their perspective on the ground. Though a ground vehicle could potentially have accurate location information from GPS, we consider the asset tracking viable, because: (1) There are situations where GPS information in a disaster site is either unavailable or degraded due to obstructions or other factors; (2) The actual location of the hazards are not known and the monitoring drones need to maintain accurate relative positioning of the ground asset; and, (3) The ground asset needs to update its motion planning to optimize for known hazards as they are estimated by monitoring drones.

In this context, our IKD agents are unmanned aerial vehicles (UAVs) performing continuous monitoring of the disaster site while communicating across a first responders mobile ad-hoc network (MANET) as shown in Figure 6.1. In this study to validate the CA-POMDP, CoDec, and IKD concepts, the agents are tracking ground assets and need to determine *what* sensor information needs to be shared between them *when* to ensure they maintain situational awareness of the hazard risks. They also provide a relative position warning to the ground asset so that it can update the edge costs of its own motion planning algorithm appropriately, e.g., MDP or D*-Lite [115].

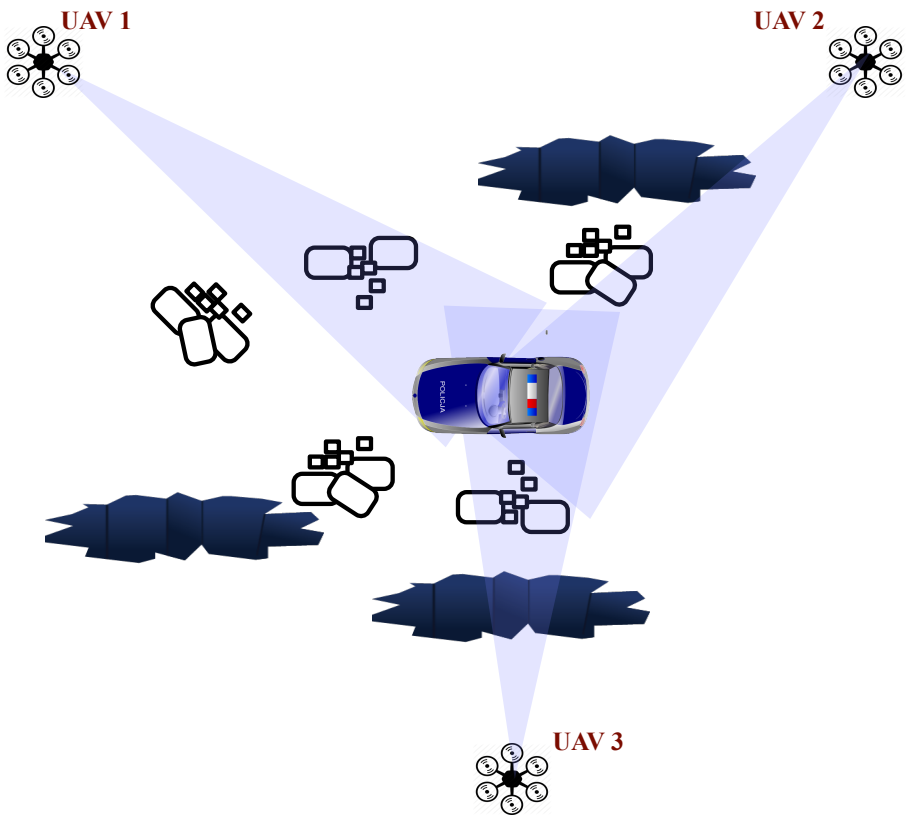


Figure 6.1: The conceptual diagram of the situation being modeled with an autonomous emergency vehicle traversing a disaster site with blocked views of potential hazards (e.g. sinkholes)

As represented in Figure 6.2, the agents have a limited field of view of the area around them and need to ensure that the agents provide satisfactory "visual" surveillance of the ground assets in case of a sudden change in direction or mismatched capabilities, e.g. speed.

The CoDec MDP framework implements multiple concurrent MDPs whose policy solution is dependent on model parameters from other MDPs, as discussed in Section 5.2, to highlight how *fully autonomous independent systems* could utilize the framework. Figure 6.3 shows the four primary decision engines used in the target tracking simulation and scenario. Each of these are described individually below in how they were constructed to perform target tracking, but are basically consisting of a hierarchical MDP (hMDP) to perform Motion

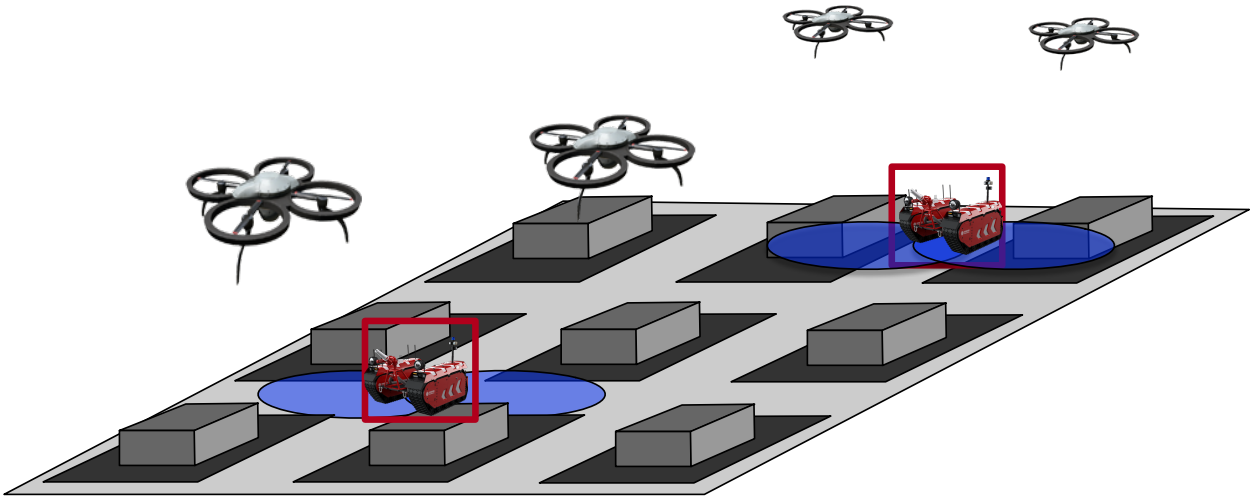


Figure 6.2: The disaster monitoring scenario used to validate the CoDec MDP formulation for Intelligent Knowledge Distribution where multiple ground assets are being monitored by multiple autonomous UAVs.

Planning, a Decentralized MDP (Dec-MDP) to perform Task Allocation, a Kalman MDP (KMDP) to do Target Tracking, and the Constrained-Action POMDP (CA-POMDP) to provide Intelligent Knowledge Distribution (IKD). Just as in [13], the two MDPs on the left are an individual MDP and a Task Allocation MDP, but the individual MDP also has a dependency on the Asset Monitoring KMDP. Though it appears at first glance the Communication CA-POMDP only has dependencies and there are no MDPs dependent on it, when a communication occurs all the models will be effected by the information I.

The IKD model, as described in Section 5, is a combinatorial problem between the neighboring nodes, the *heterogeneous* sensor data, the relevance of the information, and the current level of collaboration. Since an agent can only communicate with a single node at a time due to routing protocols with WiFi Mesh Networking, it needs to determine the risk it believes an asset is under (relevance) and how confident it is in the data it has received so far from other nodes (collaboration) to ascertain the appropriate information to send and to whom to maintain accurate situational awareness, all while not overly consuming the limited resources

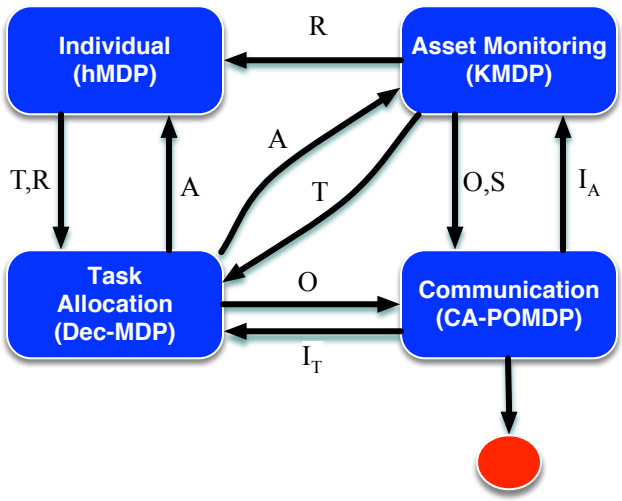


Figure 6.3: Concurrent MDP interdependencies between the Individual, Task, Target, and Communication decision engines with Task Allocation being a Decentralized POMDP and the Communications being a CA-POMDP for the Target Tracking experiment.

of bandwidth in the MANET and power available from the UAV battery.

6.1 Individual MDP

The Individual MDP drives the movement of the UAV agent by determining the best overall direction to travel to quickly acquire or maintain surveillance of the ground asset. It is also the MDP, through reinforcement learning, that is adapting the models to account for agent control and environmental variation (e.g., wind). The model for the Individual MDP is simplistic, but does run into real-time calculation issues because of state space explosion. A simple 10×10 grid has 100 states with the states growing squared with the size of the grid, and is unnecessary when many of the states are not feasible within a specified time horizon. For example, a one meter grid for the one square kilometer simulation area has one million states. This complexity is reduced by combining hierarchical approaches to a factored state states for path planning to provide directional guidance in long paths.

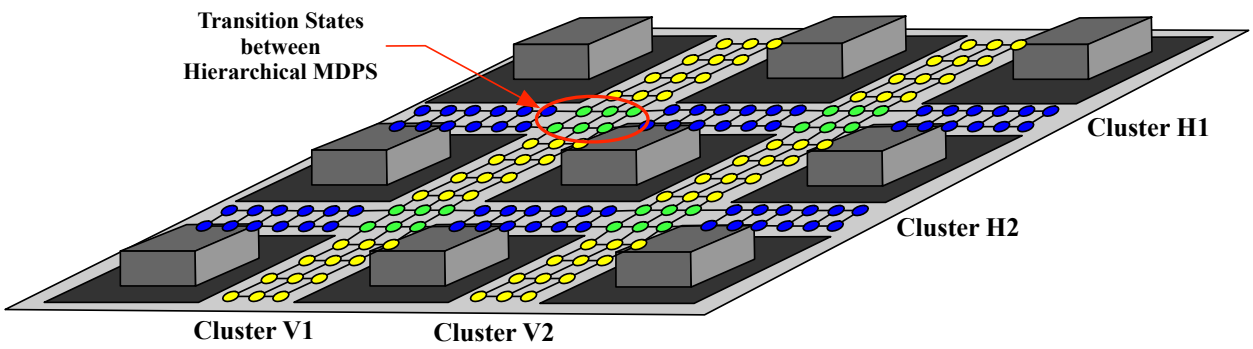


Figure 6.4: An application of a Hierarchical MDP to motion planning in an urban environment where streets can act as clusters with the intersection being transition states that can activate a change to another MDP cluster.

In the application of this effort, a hierarchical MDP (hMDP) of Section 3.2.1 was used to reduce the complexity of the state space when it came to motion planning in an urban environment, where the clustering algorithm works within the context of geographical location. Clustering of geographical spaces is natural in that we don't think of every single step we need to make from one location in a building to another, but we know we need to go from our office to the hallway, then into the next room, and so forth, as depicted in Figure 6.4 for a small portion of the urban environment to provide clarity. The two north-south streets are two independent clusters `Cluster V1` and `Cluster V2` along with two east-west streets also divided into two independent clusters `Cluster H1` and `Cluster H2`. Even though a UAV is capable of flying above buildings and other obstacles, it might not be able to maintain surveillance because of obstructions. Therefore, the state space points were kept to within the confines of the street. At the intersection of each of the streets are transitions states that exist in both clusters that intersect at that location with one of the intersections pointed out in a red circle on Figure 6.4. When a cluster MDP reaches on of the transition states, then the overall root MDP can change the active cluster MDP to the intersecting street, if the motion planning policy and reward are appropriate.

The rewards for the agent are determined by the predicted transition behavior of the Asset

Monitoring KMDP. This posterior modeling aids the agent to maintain surveillance of the target by predicting where it will go and staying within visual range. Figure 6.5 shows an example of a probabilistic reward structure from the Asset Monitoring KMDP on where the target is likely to be located over a finite horizon. For the root MDP rewards in the HMDP, the maximum reward within a sub-MDP cluster was used.

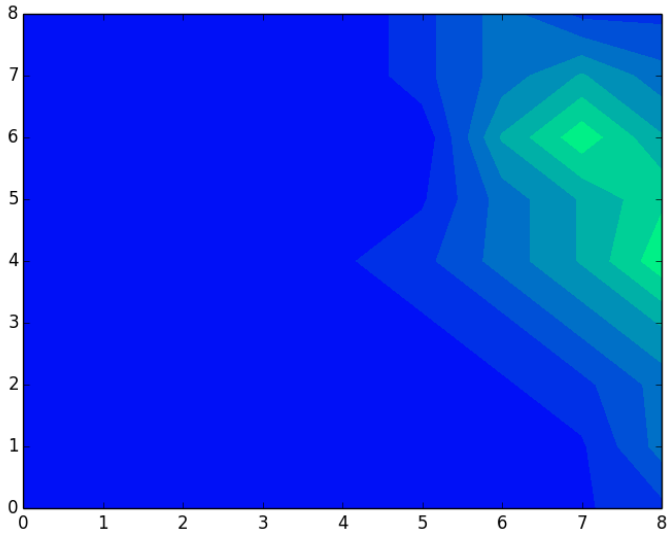


Figure 6.5: Contour Graph of the Individual MDP Rewards provided by the predicted ground asset locations from the Asset Monitoring KMDP over a finite horizon.

The transition probabilities of the Individual MDP simply initially consist of *a priori* motion control. These transition probabilities are updated based upon the agents ability to maintain proper motion control from either internal factors (e.g., voltage variation in rotors) or external environmental factors (e.g., wind). The available actions for the Individual MDP are to simply stay at its current location or move in one of the eight cardinal directions, $< N, NE, E, SE, S, SW, W, NW >$. The same probabilities for motion within a sub-MDP cluster where also used for the transition probabilities for the root MDP in the HMDP.

6.2 Task Allocation Dec-MDP

The Task Allocation Dec-MDP determines the appropriate ground asset $i \in \mathcal{T}$ for the agent to track in coordination with the Individual MDP based upon its capabilities to maintain surveillance on the target. If there are enough agents to track the available assets, the model should allow an agent to explore the area of operation seeking other assets.

In Task Allocation model, the set of actions $\boldsymbol{A}_i$ per agent i consists of two basic decisions, `explore` the environment for unknown targets a_0 or `track` one of the existing targets $i : \cup_{i \in \mathcal{T}} a_i$. Along the same line of the actions, the set of states $\boldsymbol{S}$ has a base state of `undiscovered` while there is a combinatorial set of `uncovered` or `covered` for each target discovered.

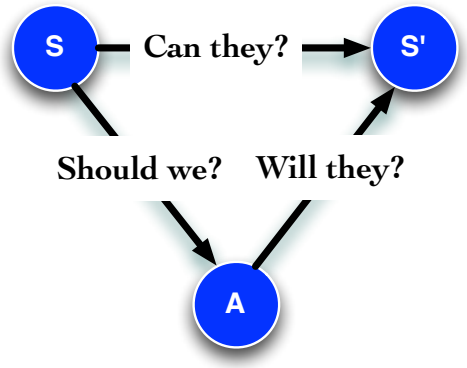


Figure 6.6: Philosophical connections required in understanding the construction of the transition and reward matrixes of the Task MDP.

Calculating the transition probabilities and rewards for task allocation is rather complex due to the rather simplex definition of the action and state space. Figure 6.6 shows the interplay between the current state, the action, and the next state that needs to be represented in the transition and reward matrices.

- $s \rightarrow s'$: *Can they?* – Is the agent even able to reliably change states and monitor an-

other ground asset? With multiple agents and targets with heterogeneous capabilities, an asset may not be the right assignment to track the agent because the target could be too maneuverable for the agent, too fast for it to maintain relative positioning, or vice versa. The Asset Monitoring KMDP learns a predictive model of the target that is utilized to influence the transition probabilities and rewards for this relationship.

- $s \rightarrow a$: *Should we?* – Is it in the best interest of the UAVs to switch task assignments that they have already been assigned? Has the UAV been tracking this ground asset and other assets are within the vicinity of each other? Even though there might be a mismatch in capabilities between the UAVs and its assigned ground asset, the UAV might be in the best position at that point in time to maintain some level of surveillance on the ground asset that could be completely lost track of if it moved to another task.

- $a \rightarrow s'$: *Will they?* – Will another other agent have the same realization that the a new task assignment is globally beneficial? With limited and asyncrhonous communications, the Dec-POMDP allows for the determining an action based upon a global understanding of the other agents, which comes in the form of a stochastic controller, discussed in Section 3.3.2.

The transition probabilities for the task allocation MDP are constructed around the communication and collaboration of the decentralized agents. Except for the task assigned to the local agent itself, the other probabilities have to deal with the likelihood, $\mathcal{L}$, another agent will have the same realization as the local agent on the rewards or even existence of an asset. At this stage of the experiment, this likelihood is kept static and tied to the probability of collaboration between agents, p_c.

$$\mathcal{L}_i(s_i'|s_i, a_i) = \begin{cases} p_c & \text{if } s_i = s_i' \\ (1 - p_c) & \text{if } s_i \neq s_i' \end{cases} \tag{6.1}$$

$$p_i(s'|s, a) = \frac{\mathcal{L}(s_i'|s_i, a_i)}{\sum_{i \in M} \mathcal{L}(s_i'|s_i, a_i)} \tag{6.2}$$

The reward structure is were most of the calculation and influence on the decision making to determine the appropriate assignment of tasks and ensure mission objectives, like coverage of assets. The reward for any particular action is $R(s, a) = R_a + (-R_s) + R_j$ and is a summation of multiple rewards: the reward of the current state $R_s = V^*(s) + (-\Delta(s))$, the reward of the action $R_a = V^*(a) + (-\Delta(a))$, and the joint reward R_j

$$R_j = \sum_i \begin{cases} R_k & \text{if } s_i = a_i \\ -R_{\neg k} & \text{if } s_i \neq a_i \end{cases} . \tag{6.3}$$

The reward calculation for the state and action utilizes the optimal value determined in the Individual MDP and is adjusted by the variety of the states or actions, $\Delta(\cdot) = \psi_{coverage}(\cdot) + \psi_{search}(\cdot)$. The variety ensure that a particular state or action is properly covering the ground assets, $\psi_{coverage}$ and penalizing the reward if there are more than enough UAVs to cover the assets and an agent is not searching for other ground assets, ψ_{search}. The joint reward, R_j, is calculated from the impacts of switching from a state to another by the action as a form of hysteresis penalization, where it is best to permit an agent to continue tracking an asset R_k it has already acquired then reassigning it $R_{\neg k}$, unless the rewards are that much greater for doing so.

6.3 Asset Monitoring KMDP

The Asset Monitoring Kalman MDP (KMDP) is responsible for learning a model of each ground asset, which includes learning the transition probabilities, through reinforcement learning. This model informs the task allocation model for utilization in the transition probabilities of maintaining coverage of a ground asset and the Individual MDP reward function with state prediction. The individual performance MDP allows an agent to proactively proceed to a location that an asset will likely be at over a finite horizon without always reacting after the fact which improves the probability of maintaining surveillance.

The observation of ground assets is a partially observable environment because of the uncertainty inherent in the sensors, but this adds a level of complication to the overall CoDec solution that can be mitigated. Therefore to simplify the model, a fully observable MDP will be used in collaboration with Kalman filters. The observations of the asset are used to update the Kalman filter state estimator and the predicted state is used as the "fully observed" state in the MDP.

A Kalman filter could be utilized in isolation for asset monitoring, but the reinforcement learning of the Markov decision process provides a probability transition model T_{AM} that informs the task allocation as to the fitness of assigning a ground asset to an agent. If the learned transition model indicates a probability to a next state s' that an asset could not maintain, then there is a mismatch in the assignment of an agent to that asset.

To provide the reward function for the individual MDP, the Kalman filter state prediction function is used for a number of finite time steps into the future to create a probabilistic reward function as shown in Figure 6.5. Since a Kalman filter assumes a Gaussian distribution, there should be one global and concave probability distribution that should allow the Individual MDP to converge on a solution.

Remark 6.1. The number of finite time steps is based on the designer and should take into consideration the time for recalculating the individual MDP as the reward function changes and the uncertainty of the Kalman estimator to maintain an accurate model.

6.3.1 Unscented Kalman Filters

The experiment employs Kalman filters to track the position and bearing of the ground assets, which are non-linear models due to the state transition function and triangulation of the ground asset from observation drones. Kalman filters (KF) are a common approach to guidance, navigation, and control of vehicles and robots [116] as a linear quadratic estimator of the state of a system from a series of measurement or observation samples over time. They are a Bayes filter with a Gaussian assumption for state transition noise Q and measurement noise R that performs very well with linear models. The KF has been extended to non-linear systems with Extended Kalman Filters (EKF) that have approximate linear behavior within the time period of updates, but in highly non-linear systems the EKF can perform poorly [117] because the covariance is propagated through linearization of the underlying non-linear model.

The Unscented Kalman filter (UKF) uses a deterministic sampling technique known as an unscented transform consisting of a minimal set of sample points around the mean. The unscented transform passes a set of sigma points $\boldsymbol{x}$ through a nonlinear function $\boldsymbol{y} = f(\boldsymbol{x})$ to yield a new set $\boldsymbol{y}$ of transformed points. The mean and covariance of the transformed points are calculated simply as

$$\mu = \sum_{i=0}^{2n} w_i^m \boldsymbol{y}_i \tag{6.4}$$

$$\Sigma = \sum_{i=0}^{2n} w_i^c (\boldsymbol{y}_i - \mu)(\boldsymbol{y}_i - \mu)^T \tag{6.5}$$

where w is a weighted value from the sampling technique for the mean m and the covariance c for each sample i.

The Van der Merwe's Scaled Sigma Point Algorithm is a deterministic sampling technique for the unscented transform, which provides a good balance between performance and accuracy [118], and is a slight reformulation of the Scaled Unscented Transform [119]. It utilizes three parameters to control how the sigma points are distributed and weighted: α, β and κ In this algorithm, the first sigma point X_0 is always the center of the ellipse μ. The remaining sigma points are calculated as

$$X_i = \begin{cases} \mu + [\sqrt{(n+\lambda)\Sigma}]_i & for\ i = 1 \dots n \\ \mu - [\sqrt{(n+\lambda)\Sigma}]_{i-n} & for\ i = (n+1) \dots 2n \end{cases} \tag{6.6}$$

where $\lambda = \alpha^2(n + \kappa) - n$, i is the i^{th} column vector of the matrix, and n is the dimension of x. The weights for X_0 for the mean and covariance are calculated independently than the rest respectively as

$$W_0^m = \frac{\lambda}{n+\lambda} \tag{6.7}$$

$$W_0^c = \frac{\lambda}{n+\lambda} + 1 + \alpha^2 + \beta. \tag{6.8}$$

Whereas the weights for the remaining sigma points $X_1 \dots W_{2n}$ for the mean and covariance are the same

$$W_i^m = W_i^c = \frac{1}{2(n+\lambda)}. \tag{6.9}$$

There are two occurrences that need to be addressed in the application of Kalman Filters because of intermittent communications and limited line of sight: missing measurements and missing observations. In missing measurements, there is no information at a discrete time step to update the state estimate and the predicted measurement is propagated forward as an observation [93, 120]. For missing observations, there are observations at a given time step but not of all the state information. In the case of geolocation through triangulation, a bearing and distance measurement is not available to accurately pinpoint the robot, but there are enough observations to get a less accurate measurement. With UKF, the unscented transform is performed as usual, because the model can be constructed "on-the-fly" to replicate the variance that would be seen with the lack of observation for a fully accurate triangulation by adjusting the measurement noise, R [121].

6.3.2 Asset Localization & Kalman Filter

An unscented Kalman filter (UKF) was utilized in performing localization of the ground vehicle from multiple drones, where each drone is providing a bearing and distance to the ground vehicle with variances per the capabilities of the sensor packages that are aboard that particular drone. A nonlinear Kalman filter was necessary due to (1) the nonlinearity of the motion model with arcs of motion depending on the angle of the front wheel and (2) the nonlinearity of triangulating the position of the ground vehicle from drones with sensor bearings and distances.

The motion model $f(x, u)$ of the ground vehicle is a standard *bicycle model* with static rear wheel and variable front wheel. The state model $\boldsymbol{x}$ maintains information on the position and orientation of the ground vehicle as the vector $\boldsymbol{x} = [x \quad y \quad \theta]^T$, where (x, y) is the location of the ground vehicle in the disaster site as measured in meters and θ is the bearing in radians. The state transition function $\bar{x}$ is a non-linear motion model $f(x, u)$ with Gaussian white noise $\mathcal{N}(0, Q)$ defined as

$$\bar{x} = x + f(x, u) + \mathcal{N}(0, Q), \tag{6.10}$$

where $\boldsymbol{u}$ is the command input $[v \quad \alpha]^T$ to the ground vehicle control system and is defined by the linear velocity v and the steering angle α of the ground vehicle. The measurement model $\boldsymbol{z}$

$$\boldsymbol{z} = h(\boldsymbol{x}, \boldsymbol{P}) + \mathcal{N}(0, R) \tag{6.11}$$

$$h(\boldsymbol{x}, \boldsymbol{P}) = \begin{bmatrix} \sqrt{(p_x - x)^2 + (p_y - y)^2} \\ \tan^{-1}\left(\frac{y - p_y}{x - p_x}\right) - \theta \end{bmatrix} \tag{6.12}$$

involves the bearing and distance to the ground vehicle's location $[x, y]$ from the current observation location of drone $[p_x, p_y]$, in which the bearing and range measurement noise

$$\boldsymbol{R} = \begin{bmatrix} \sigma^2_{range} & 0 \\ 0 & \sigma^2_{bearing} \end{bmatrix} \tag{6.13}$$

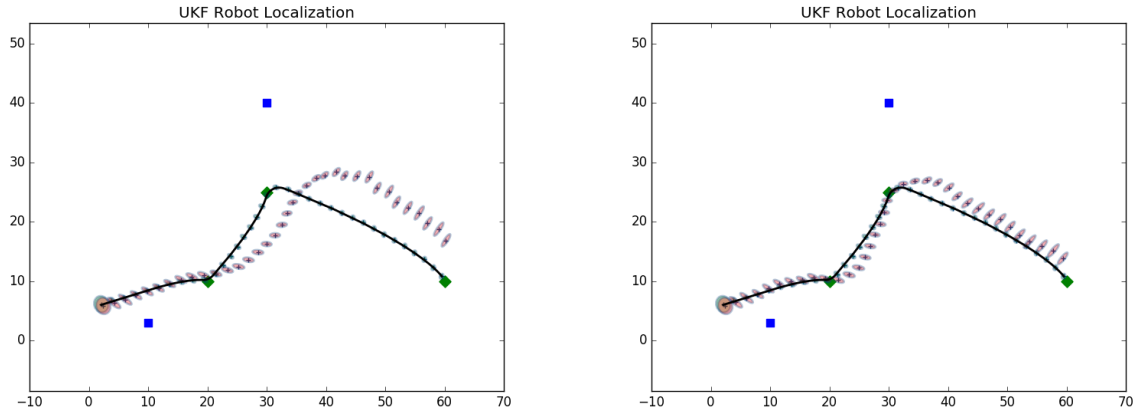
is assumed to be independent and may or may not have line of sight to the target. In cases where a drone does not have a line of sight of the ground asset nor any observation communications from another drone, this is a missing measurement.

For the UKF, the unscented transform uses a particle-based technique, which requires the ability to calculate the mean of the particles. Calculating the mean of the positions is a simple mathematical average, whereas calculating the mean of the bearing is

$$\bar{\theta} = \text{atan2}\left(\frac{\sum_{i=1}^{n} \sin\theta_i}{n}, \frac{\sum_{i=1}^{n} \cos\theta_i}{n}\right). \tag{6.14}$$

There are two occurrences that need to be addressed in the application of Kalman Filters because of intermittent communications and limited line of sight: missing measurements and missing observations. In missing measurements, there is no information at a discrete time step to update the state estimate and the predicted measurement is propagated forward as an observation [93, 120]. For missing observations, there are observations at a given time step but not of all the state information. In the case of geolocation through triangulation, a bearing and distance measurement is not available to accurately pinpoint the robot, but there are enough observations to get a less accurate measurement. With UKF, the unscented transform is performed as usual, because the model can be constructed "on-the-fly" to replicate the variance that would be seen with the lack of observation for a fully accurate triangulation by adjusting the measurement noise, R [121].

Since the UKF utilizes a state transition function rather than a linear state transition matrix to perform state updates, the function requires an understanding of the command sequences being issued for robotic motion to maintain a close estimate in an environment with 90% missing measurements, 8% missing observation, and 2% full observation. Figure 6.7 has an example of Kalman filter sate estimations with and without command estimations, where the ground vehicles is traveling between waypoints (green diamonds) with observations from drones (blue squares). As can be seen in Figure 6.7a, if the state transition function is not provided an estimation of the command of the command input $\boldsymbol{u}$ then the state estimation will differ greatly from ground truth. Whereas using observations that are already being taken, an estimate of the control commands allows us to stay within reasonable estimates of the ground vehicle, Figure 6.7b.



(a) Localization *without* Command Prediction. (b) Localization *with* Command Prediction.

Figure 6.7: Localization of a ground vehicle through Unscented Kalman Filters with and without command estimations.

If the probability of arrival λ of an observation in a Bernoulli process is less than a critical probability $\lambda \leq \lambda_c$ for a Kalman Filter, then the expectation of the estimation error covariance is unbounded [93]. The lower bound $\underline{\lambda}$ of the critical probability λ_c has a closed form solution but the upper bound $\bar{\lambda}$ requires solving a linear matrix inequality (LMI) [93].

Therefore, the authors used a simplified simulation involving fixed observation points to experimentally approximate upper $\bar{\lambda}$ and lower $\underline{\lambda}$ heuristics for $\mathbf{\Lambda}$ of the IKD reward function $\mathcal{Q}$ (5.4) as applied to Kalman filters for use in the full simulation.

6.4 Communication Concurrency Model

In the previous sections, we discussed the primary components of the agent control concurrency interaction models, but now we describe the construction of the agent communication concurrency model that provides the *Intelligent Knowledge Distribution* for fully autonomous systems. Each component of the CoDec POMDP can register data to be communicated with the CA-POMDP IKD model to other agents in the collaboration graph, $\mathcal{G}$. Some of the key information that is viable for transmission includes individual performance online learning for transfer learning of task execution, target tracking observations and online learning of target behavior, task allocation observations, task allocation rewards, and task allocation policy or action.

To reduce drastically increasing the complexity of the model, the concurrent information registered with the CA-POMDP data handler was restricted to two key bits of information that provided an optimal balance between minimizing data while maximizing information content. Therefore, the information being communicated by the system is the environmental observations from the Kalman filter of the Target Tracking Kalman MDP, which allows each agent to update their state estimators and locally update their target behavior model.

Remark 6.2. In this experiment we utilized a fully observable Dec-MDP for task allocation, but if the system needs to utilize a Dec-POMDP, then the system can register environmental observations, which allows the agents to probabilistically achieve joint optimal policy with observation information on tasks assignments being shared.

Due to size, weight, and power (SWAP) restrictions of the UAVs, they are assumed to have heterogeneous sensors with overlapping sensor support. Each UAV has been assigned to carry two sensors on their platform from a total possibility of three: (1) RF geolocation, (2) optical tracking, and (3) laser range finder. Therefore each node has the option of sending the result of a sensor to any one of the nodes that it's connected to. This creates an action space that as previously described combines the sensors' data on-board and the number of neighbors $\mathcal{N}_i$, including the action of not transmitting any information.

Remark 6.3. The combinatorial explosion caused by scaling the number of agents in the system is combatted with the use of collaboration graphs to factor the action space between agents by clustering capabilities, location, etc.

As an example, a UAV connected to two other drones with RF geolocation and optical tracking capabilities, there are five total actions to choose during any time epoch:

1. Do not communicate (Silence or Null)
2. Send RF Geolocation results to Node A
3. Send Optical Tracking results to Node A
4. Send RF Geolocation results to Node B
5. Send Optical Tracking results to Node B

The state information is assumed to be a combinatorial set of the relevance of the data, the risk to the ground asset, and the state of collaboration based on the expected value of information from previous communications and the quality of the data. As defined in Section 5.1 for establishing the CA-POMDP to perform Intelligent Knowledge Distribution, each component that registers data to be handled by the communication component needs

to provide three major heuristics: (i) the maximum value of information $\max \rho(k)$ for both the relevance of the information and the collaborative impact with a neighbor $j \in \mathcal{N}_i$; (ii) the likelihood an IKD CA-POMDP state and action $\mathcal{L}(s' \notin \boldsymbol{F}|a, s)$ might place mission objectives at risk of being achieved; and, (iii) the proximity of a quantized state to approaching heuristic bounds where collaboration will diverge and become unbounded $\mathcal{Q}(\cdot)$.

The rewards are also combinatorial as formulated in IKD with the information theoretic metric ρ based upon the information matrix of the sensor variances $\rho = 1/\kappa_a$. This leads to an updated relevance of

$$R_r(a, s) = \mathcal{L}(s' \notin \boldsymbol{F}|a, s) \cdot \frac{1}{\kappa_a}, \tag{6.15}$$

where $\kappa_a = \min_{j \in \mathcal{N}}(\sigma_i \sigma_j)$, σ_i is the variance of the local sensor information being transmitted, and σ_j is the variance of a collaborative agent $j \neq i \in \mathcal{N}$. This also leads to an updated collaboration reward of

$$\hat{R}_c(a, s) = \frac{1}{|\mathcal{N}|} \sum_{j \in \mathcal{N}} \mathcal{S}(\mathcal{Q}(\lambda_j|a, s)) \cdot \frac{1}{\kappa_a}, \tag{6.16}$$

where the sigmoid function $\mathcal{S}(\cdot)$ has a center point $x_0 = \frac{1}{2}(\bar{\lambda} + \underline{\lambda})$ and k is set to create a switching function beyond the heuristic bounds of $\boldsymbol{\Lambda}$. The function $\mathcal{Q}$ adjusts λ_j based upon the current state s of communication collaboration and the value of information κ_a in the action a to improve the covariance of the Kalman filter.

The transition probabilities for relevance are driven by the relationship between the monitoring UAVs and the ground asset. The ground asset is assumed to be reachable by the UAVs through the MANET or a multi-hop network so that it can be influenced in finding a more conservative route by updating the edge costs between waypoints around the relative position of the hazard. The timeliness and usefulness of the data being transmitted and received

was utilized to calculate the collaboration transition probabilities via repeated simulation.

Observation probabilities are defined through subject matter expertise and repeated simulation. The Kalman filter and a sufficient statistic of observation histories, a Bayesian estimator, were processed through a belief function υ to create the *relevance* and *collaboration* mapping to the observation space $o \in \boldsymbol{O}$. The observation states themselves are a discrete categorization of local observations using a K-means clustering algorithm learned unsupervised from numerous Monte Carlo simulations. In particular, the mapping of observations υ_r is performed solely by cluster classification of the state estimation and its covariance, driven by sensor inaccuracies of the ground asset's proximity to hazards to observation relevance states o_r. Whereas, the observation mapping υ_c for collaboration is based on multiple observations: (i) a Bayes estimation of the current arrival probability $\hat{\lambda}_{ij}$ of messages $\sigma : j \to i$ or $\sigma : i \to j$ used to ascertain proximity to heuristic bounds; (ii) the usefulness of information σ_k recently received from or sent to another agent j according to κ_a; and, (iii) the lagged autocovariance $cov(\boldsymbol{P}[m], \boldsymbol{P}[m+l])$ of the Kalman filter covariances $\boldsymbol{P}$ to evaluate the impact of information exchange over time in maintaining mission objectives involved in asset tracking. A *collaboration* mapping υ_c is the cluster classification of $\hat{\lambda}$, σ_k, and $cov(\cdot)$ to the observation space o_c. The relevance o_r and collaboration o_c observations from the mappings υ are indexed to the observation o in the combinatorial set, which then drives the edge transitions of the finite state controller policy.

If a Dec-POMDP is desired, the Task Allocation component is registering the observational data with the communication component. This simplifies the quantity of data that needs to be decided on while still providing highly relevant information to inform decentralized task assignments. The likelihood $\mathcal{L}$ of $s' \notin \mathcal{F}$ is derived from *a priori* information of the likelihood that the system will loose track of an asset and not be able to reacquire. For each one of the task allocation observation states, this means:

- **Undiscovered** – The model should not penalized the agent for not being on a target, but should highly encourage exploration $a = 0$ of the environment to quickly locate any targets so surveillance can be maintained and mission objectives be meet. Any other action during a fully undiscovered state would lead to mission failure as no targets have been located.

- **Uncovered** – Any action that could leave a ground asset uncovered will cause the mission to likely fail, therefore a large penalty for having an uncovered asset is assigned. A belief $\mathcal{B}$ that an action set $\boldsymbol{A}$ across all the agents might leave a known asset uncovered and put the mission at risk.

- **Covered** – This is an optimal state for a known set of ground assets that have been discovered and an excellent reward is provided but only as long as there is an agent believed to still be exploring and/or there are enough agents covering the ground assets based upon asset model behavior.

In a Task Allocation Dec-POMDP, the maximize value of information $\max \rho(k)$ was calculated using KL-Divergence of the belief $\mathcal{B}$ that all the agents were covering ground assets for both collaboration and relevance rewards but with collaboration having the appropriate focus on the neighbors $\mathcal{N}_i \in \mathcal{G}$. The heuristic bounds $\mathcal{Q}$ for when task allocation will exceed bounds and diverge like the likelihood function is highly focused on maintaining surveillance of the assets and ensuring all assets have been located. The key mission is to ensure the agent's find the ground assets (e.g., exploration) and maintain surveillance of an asset in such away they are not able to reacquire it before it encounters a hazard or risk.

Remark 6.4. There is an issue that needs to be addressed in the agent communication model in that the likelihood is high that they will start off in a mission failure state if no assets are in sight. Therefore, the heuristics should initially favor exploration to maximize

value and transition to mission heuristics, as defined above, over a finite amount of time.

6.5 Constraints on Communication

The constraints being analyzed by CA-POMDP are the utilization of bandwidth (bytes per second) and power (Watt per second) over one second, which is a sampling length of 10 epochs as the UAV makes a collaboration decision every $100ms$. The constraints are considered soft constraints because there is often more available bandwidth in the network than is being allocated to multi-agent collaboration. Also, network protocols like ZeroMQ permit queueing techniques to account for myopic link saturation. The power utilization is considered an average power consumption needed for the communication system to use to ensure the UAV can maintain the required flight time needed to maintain situational awareness of the disaster site.

Resource utilization by the UAVs is dependent on the type of information they are sending. It is assumed that sending optical data requires more data and therefore higher power requirements to achieve the SNR needed for higher bandwidth physical layer protocols, e.g., 64QAM. The data for laser range finding then RF geolocation are assumed to take sequentially less data and power than optical data.

6.6 A Semantic Network for Learning Collaboration Strategies

For the drones to develop a coordination graph $\mathcal{G}$ to reduce the computational complexity involved in large action and state space models, we examined two clustering approaches

for an unsupervised learning of collaboration strategies: spatial clustering and relational learning. To permit collaboration strategies to be formed into a coordination graph from relational learning, the systems needs an ontology that the infinite relational model can leverage. Using the Unified Medical Language System (UMLS) semantic network [122] as a guide, the target tracking scenario constructed a simple semantic network from taxonomies and classifications in Section 2.2.2.

The semantic network is a relation $R: \quad T^1 \times T^2 \times T^3 \rightarrow \{0,1\}$ between a set of objects (T^1), a set of features (T^2), and a set of binary predicates (T^3). The set of objects T^1 consists of the heterogeneous drones that are deployed in the field. The features T^2 are provided by the DoD classifications in Table 2.1, the range and endurance classifications in Table 2.2, and Figure 2.4 on drone service taxonomies. The set of predicates T^3 are the "verbs" relating the objects and features between each other in the semantic network like *supports*, *provides*, *assists with*, *capable of*, *complicates*, *prevents*, *associated with*, *result of*, and *affects*. In Figure 6.8, we show a sample of the sementic network constructed from these objects, features, and predicates for the target tracking scenario where the objects and features are nodes of the graph and the predicates are edges.

6.7 Results

The applicability of Concurrent Decentralized (CoDec) and Constrained-Action (CA) POMDPs to address resource awareness in Intelligent Knowledge Distribution was investigated through a series of Monte Carlo simulations. From these simulations, the probabilistic constraint satisfaction was analyzed between intelligent (IKD), greedy, and naive communication models while confirming their ability to maintain acceptable state estimations. In our experiment, greedy communications assume a constant communication of the best information observed

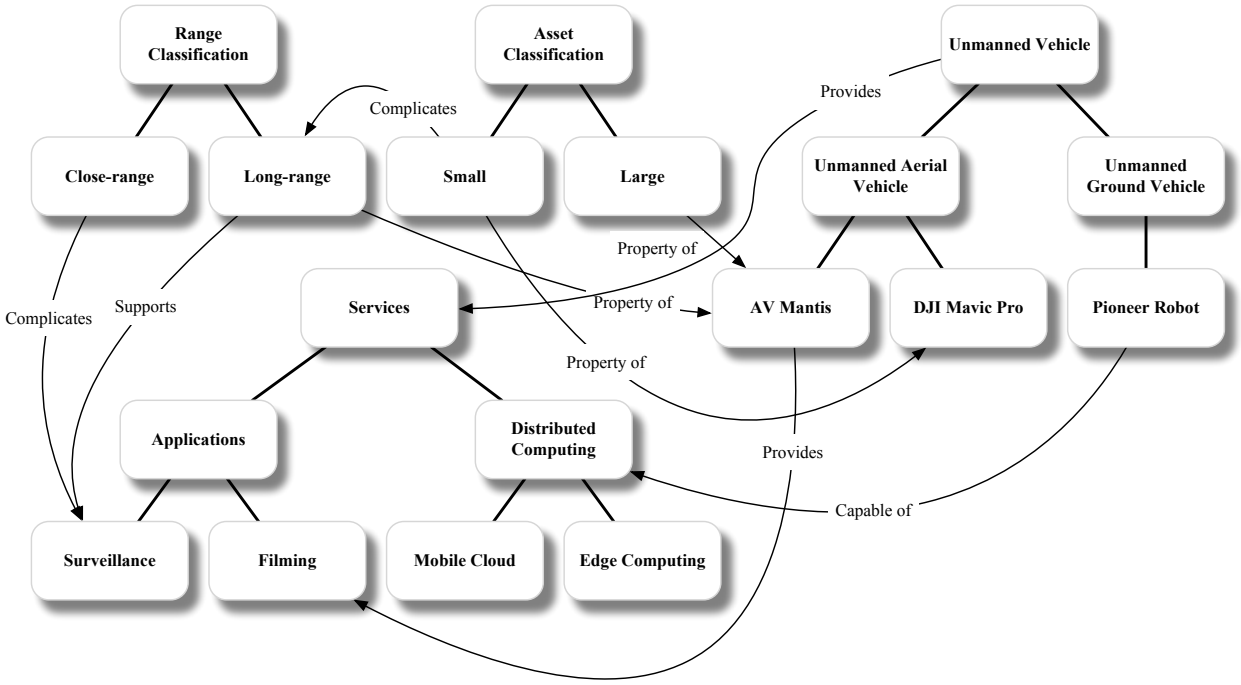


Figure 6.8: A sample of the UAV semantic network constructed from taxonomies and classifications.

at all times alternating the destination between each time epoch among collaborating neighbors. Naive communications applies a probability of communicating depending on the relevance of the information, the estimated proximity to a hazard.

The simulation environment was constructed inside Python utilizing the SimPy discrete event simulator with a one square kilometer "disaster" site. Anywhere between three to five hazards were randomly placed in the environment with a restriction that the randomly placed ground vehicle is not initially placed within 50m of one of the hazards. Otherwise, the ground vehicle would crash into the hazard before monitoring drones were able to establish state estimations. A goal location indicating a point of interest or a victim is randomly selected, but only if it is over 300m from the ground vehicle and at least one hazard is in proximity to the estimated path. The threshold parameters for MCMC utilized generally accepted thresholds from the literature whereas recalculating a constrained FSC from the

variation of information was set based upon the computational capability of the hardware platform. A Monte Carlo of at least 1,000 simulations were run to evaluate that each model was able to consistently meet mission objectives, i.e. not crash.

Remark 6.5. Platform limitations are the primary driver for the recalculation of a policy from the variation of information, so that a computer is capable of calculating a policy well before the probability of a new policy needs to be recalculated again to prevent any race conditions and processor saturation.

6.7.1 Constraint Satisfaction and Optimal Value

The model was run through the Constrained-Action POMDP for multiple values of the resource constraints or limitations for predetermined resource utilizations per action. Figure 6.9 shows the analysis of the data from results of the constraint satisfaction and the value of the finite state controller. For all cases, the CA-POMDP was able to achieve approximately 90% constraint satisfaction for the final FSC. In the worst-case scenario evaluated at a soft constraint of 6 Mbps, the optimal controller was only able to achieve 10% constraint satisfaction and with the addition of constraint states was able to improve it to 90% with the constrained FSC staying within 54% of the optimal value function. The comparison between the optimal controller without any attempt at constraint satisfaction and the the constrained controller meant to met the probability of constraint satisfaction was measured as a percentage of the Frobenius norm between the two value functions solved during policy evaluation as a set of linear equations.

As the optimal unconstrained FSC naturally approaches the constraint satisfaction, then the required constraint states to make it compliant drastically decrease its impact on the value. This is possible because the discrete optimization used in finding a constraint state to meet

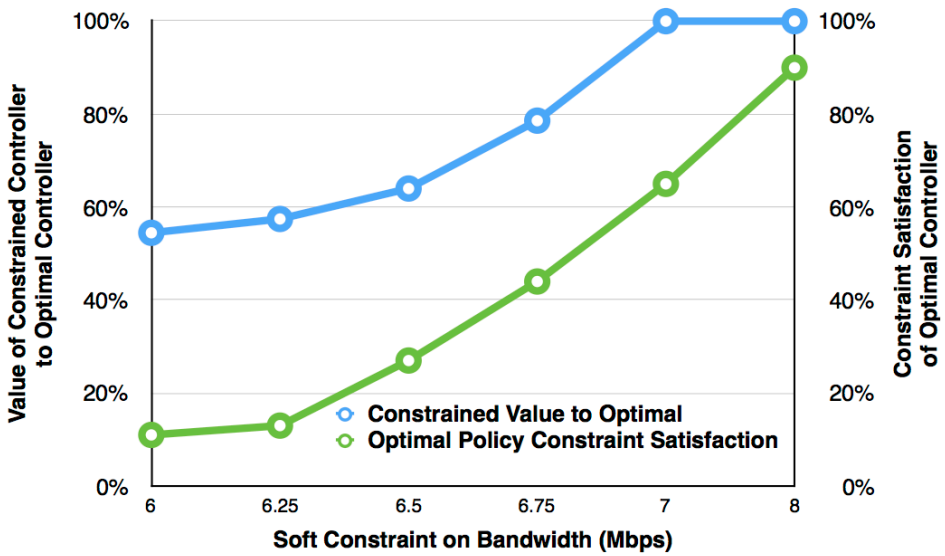


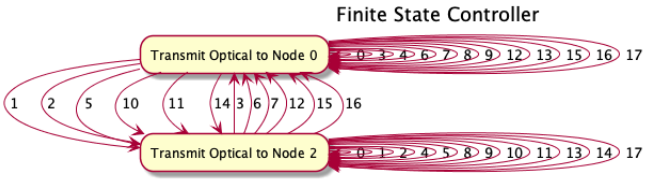
Figure 6.9: The impact of the value function to the optimal controller and constraints satisfaction between the optimal controller without any constraint requirements and the constrained controller versus bandwidth limitations for a system communicating with two collaborative nodes with three different information types.

the constraint requirements is able to find an action to inject into the controller that has higher reward despite a possibly higher resource utilization.

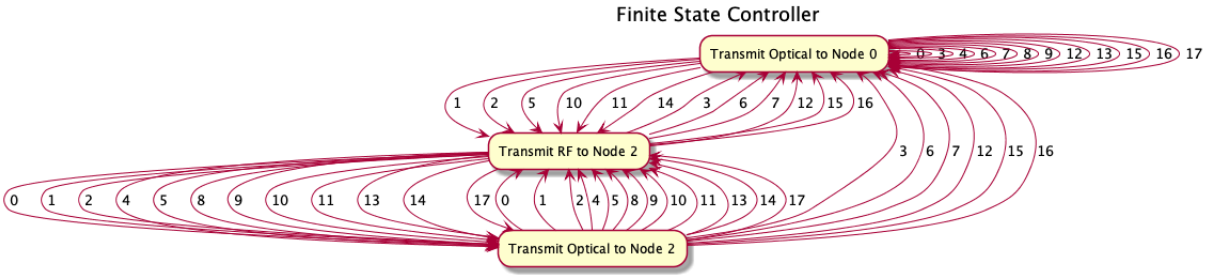
6.7.2 Constraint Impact on Automata

The discrete optimization of the CA-POMDP seeks to introduce a constraint state that brings the constraint satisfaction above the 80% desired behavior while minimizing the impact to the value function of the FSC. The farther away the optimal FSC without any constraint enforcement is from the desired constraint satisfaction, the more likely the CA-POMDP will assign constraint states that utilize the least resources which will negatively impacts the multi-agent collaboration. As the unconstrained FSC approaches but does not meet the desired behavior, an action that utilizes more resources, though less than matched machine state, and has a greater value to the collaboration of the system is chosen. An example seen

during this simulation is the inclusion of *Transmit Optical*, Figure 6.10, as the constraint state when the constraint satisfaction was naturally low and the inclusion of the new action was selected since it requires less energy and bandwidth then the optical data.



(a) Optimal Controller



(b) Constrained Controller

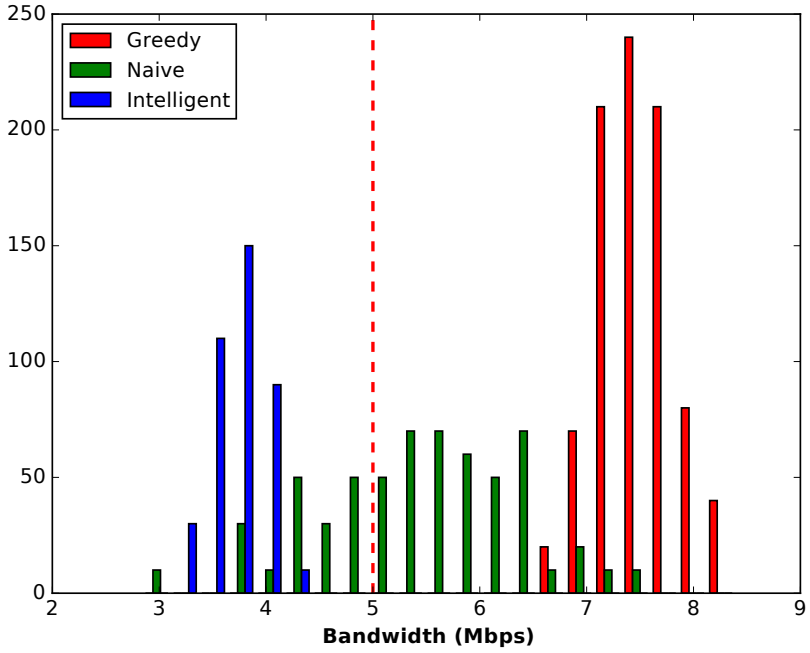
Figure 6.10: Comparison of (a) an Optimal Controller with only 10% constraint satisfaction and (b) a Constrained Controller showing the addition of an action to bring the system within resource limits of the UAV. Nodes represent actions in the FSC and the edges are observation edges. See Figure 4.2 for a clearer example.

One interesting result seen in Figure 6.9 is the point where the constrained FSC value is the same as the optimal FSC despite the unconstrained FSC being below the 80% desired behavior. This was determined to be due to the fact that it introduced into the FSC a constraint state that was exactly the one it replaced but optimized the edge transitions to achieve the desired constraint satisfaction behavior.

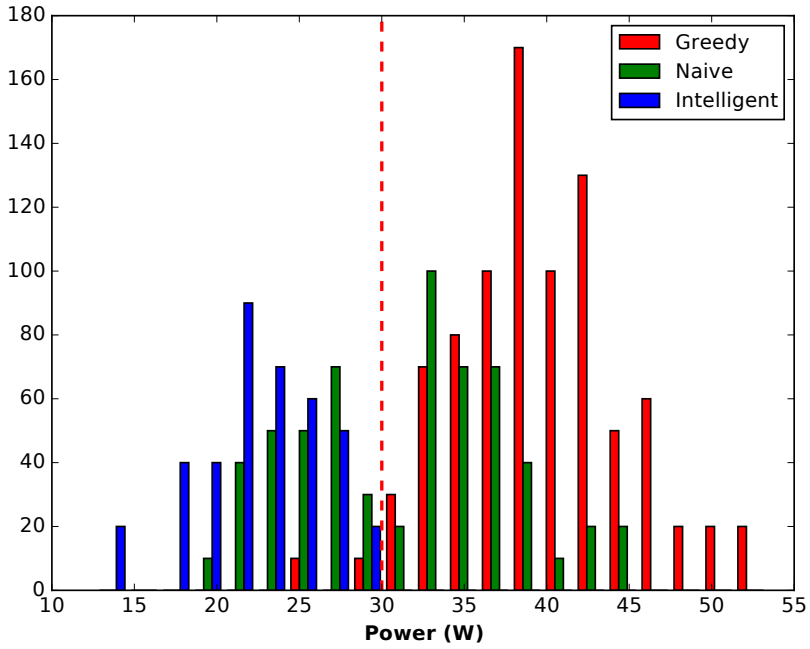
6.7.3 Constraint Satisfaction of Intelligent, Naive, and Greedy Communications

The probabilistic constraint satisfaction between the three communication models was analyzed in a Monte Carlo simulation where the simulation would randomly determine a start location for the ground vehicle, a goal location for the ground vehicle, placement of hazards in the environment, and an initial belief state for the controller. The UAV monitoring drones had a predefined search pattern that was consistent between all simulations. During the IKD simulations, the CA-POMDP model was allowed to recalculate a new constrained policy during the execution of a simulation when the variation of information exceeded a threshold to ensure continued compliance to constraints.

The utilization of resources per second by each agent during all these simulations was graphed as histograms in Figure 6.11 for the two limiting resources along with their soft constraints (dotted red line). The probabilistic constraint satisfaction to these soft limits of both bandwidth and power were substantially improved for intelligent knowledge distribution compared to those of both naive and greedy communication models. This confirms that the CA-POMDP substantiation for *Intelligent Knowledge Distribution* was able to find a controller that complied with probabilistic constraints against soft limits and adapt the controller as needed. During execution, the constraint state "Transmit RF to Node 2" of Figure 6.10b would be taken at appropriate times to reduce resource usage. This is due to the fact that the CA-POMDP algorithm redirected a portion of the observation transitions of an existing optimal machine state of the FSC to the new constraint state that utilizes less resources. The utilization of resources per second by each agent during all these simulations was graphed as histograms in Figure 6.11 for the two limiting resources along with their soft constraints (dotted red line). The probabilistic constraint satisfaction to these soft



(a) Bandwidth



(b) Power

Figure 6.11: Comparison of constraint satisfaction from three Monte Carlo simulation scenarios involving drones performing either Greedy, Naive, or Intelligent Communications.

limits of both bandwidth and power were substantially improved for intelligent knowledge distribution compared to those of both naive and greedy communication models. This confirms that the CA-POMDP substantiation for *Intelligent Knowledge Distribution* was able to find a controller that complied with probabilistic constraints against soft limits and adapt the controller as needed. During execution, the constraint state "Transmit RF to Node 2" of Figure 6.10b would be taken at appropriate times to reduce resource usage. Power was not was not a limiting constraint in this simulation as can be seen in Figure 6.11b, so the bandwidth was the primary constraint that drove the behavior of the controller.

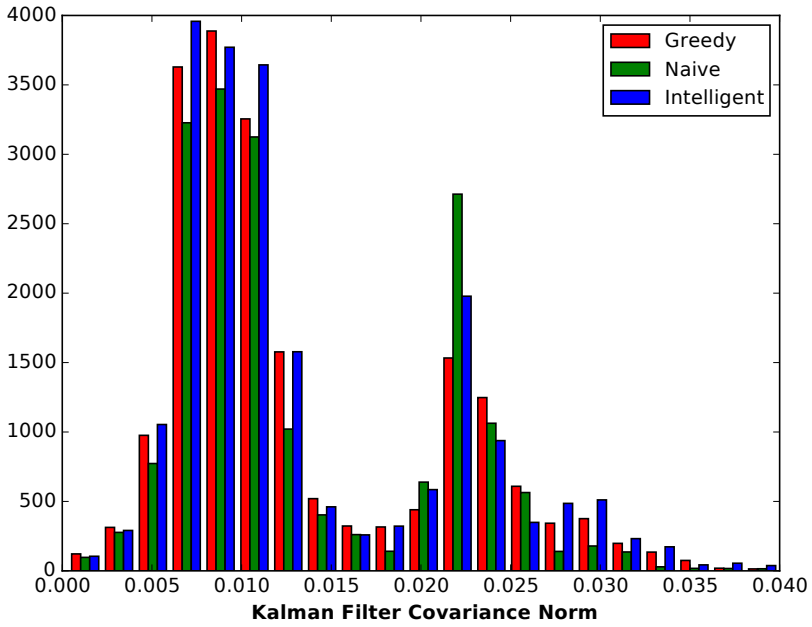


Figure 6.12: Histogram of the norm of the Unscented Kalman Filter covariance matrix showing that despite the constrained actions of IKD, the intelligent controller was still able to maintain accurate estimates of the ground vehicle.

As expected, the naive communication model with its probabilistic approach to making communication decisions on data relevance alone utilized less resources than the purely greedy communication model. Despite the improvement shown with the naive approach, our IKD approach was compliant to the soft constraints while not negatively impacting the state estimation, determined by the norm of the UKF covariance in Figure 6.12. The two

distributions in the graph are indicative of the times when the filter has full observational data and when the filters have missing observations and measurements. Figure 6.12 also validates that the three communication models maintained similar distributions during the Monte Carlo simulation.

The first two columns of Table 6.1 further validates our results showing the probabilistic constraint satisfaction, or CDF below soft constraints, to the bandwidth and power limits. The accuracy of the state estimations are shown in the last column and it is of interest to note that IKD maintained better estimations than naive. It is hypothesized that the improvement results from IKD sending the most relevant data to ensure bounded performance of the Kalman filter, whereas naive does not have that level of contextual understanding.

Table 6.1: Comparison of Simulation Metrics between Greedy, Naive, and Intelligent Communications.

Model	**BW**	**Power**	**Cov**
Greedy	0%	2%	0.0132
Naive	32%	40%	0.0135
Intelligent	97%	97%	0.0133

Remark 6.6. When the probabilities of the naive approach were decreased, i.e. less frequent communication, in an attempt to match the resource utilization of the intelligent model, the ground vehicle would collide into hazards approximately half the time in the Monte Carlo simulation. This is the basis of our hypothesis that the intelligent model for an agent was able to send the correct information to the right neighbor at the right time to ensure the state estimations could timely warn the ground vehicle of potential hazards, whereas the naive approach made poor communication decisions.

Remark 6.7. It is interesting to note that during the construction and analysis of the system that if the optimal controller meets probabilistic constraint satisfaction less than 15% of the time, then there was an unacceptable level of probability that the the resulting

constrained controller would not meet mission objectives. Future work can investigate the driving mechanism behind mission success and techniques to compensate like adaptive soft constraints to ensure mission objectives.

6.7.4 Constraint Learning

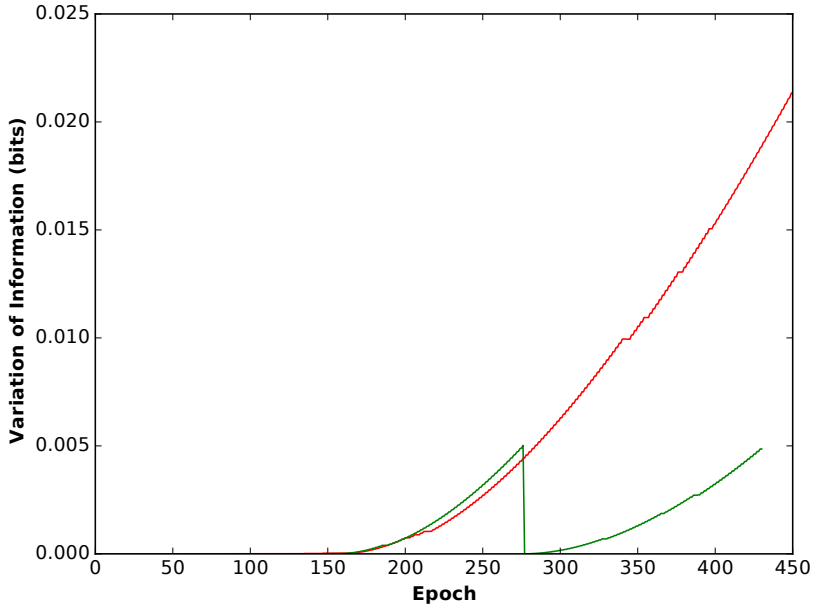


Figure 6.13: Variation of Information during a simulation with (Green) and without (Red) adaptation of the controller where a trigger caused the system to adapt the controller around 270 epochs or 27 seconds into the simulation.

The system adapted to the two learned parameters: the actual resource utilization of actions by the system over time especially as the battery deteriorated and the actual edge transition probabilities to properly evaluate a controller's constraint satisfaction. The threshold for recalculating a controller is based upon the current variation of information which is shown in Figure 6.13 without the adaptation trigger in red. In Figure 6.13 the trigger for adapting the controller was set to 0.005 which causes the sawtooth behavior in the graph, shown in green, which is the point at which IKD recomputed a FSC with the latest learned information.

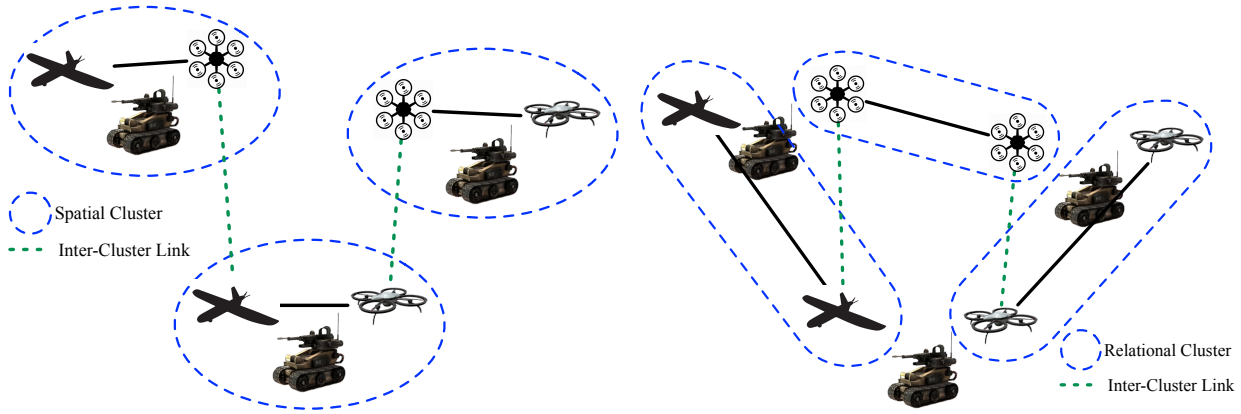
Returning back to the graphs in Figure 6.11, UAV 1 (green) which is performing the IKD has a slowly decaying utilization of resources around 27 seconds. At that point in time, the variation of information in Figure 6.13 also triggered the IKD to recompute a FSC and the drone immediately increases its usage of resources. This is due to the fact that a constraint state previously injected in the controller had more conservative resource utilization and with the realization that it is not using as many resources changes to a higher resource intensive action. In this case, the controller removed the *No Transmit* action with a *Communicate Optical Information to UAV 0*, which improves the overall collaboration between agents and improves the value function.

6.7.5 Learning Collaboration Strategies

It is the objective of the simulation to show the system learning collaboration strategies without any predefined *a priori* coordination graph. Each one of the UAVs in the simulation were all homogeneous in their service, providing surveillance capabilities, but are heterogeneous in their performance criteria: speed, position maintenance, and maneuverability. With the understanding of their own capabilities, they learn to coordinate with appropriate peers based upon the available assets.

In Figure 6.14, we see how the coordination graphs between the two clustering techniques differ in their creation of a coordination graph. When communication is established geographically with spatial clustering per Figure 6.14a, the coordination graph is more focused on highlighting communication between drones that are in close proximity. They have to be forced to establish communication outside their cluster to ensure information about other targets and areas of the environment are being shared. On the other hand, Figure 6.14b shows how relation learning causes the coordination graph to establish communications within fea-

tures. The strength of a link with a probability η assists in aiding the coordination graph to find mismatched clusters that are useful. This becomes useful to have connections outside of a cluster to another group of drones that could aid in communication rather than just assuming they should be connected as we have to make with spatial clustering. In both situations once the clusters have been established, it was necessary to determine which agent within a cluster should have a connection in the coordination graph outside of the cluster. To prevent exacerbating the combinatorial problem of multi-agent coordination, it was desired that only a single agent within a cluster should have a connection to another cluster. For the spatial clustering, a random agent within a cluster was selected to communicate to another cluster. We have the benefit in relational learning of having more information about different agents strength or proximity to other clusters. Therefore, the agent with the strongest proximity to another cluster was selected and if there was more than one agent that qualified, one was randomly selected.



(a) Spatial clustering (b) Relational learning

Figure 6.14: Comparison of coordination graphs constructed from (a) spatial clustering and (b) relational learning.

The semantic network needs to be structured correctly and the results of relational learning properly analyzed to ensure the relationships are established correctly for a specific mission. In the disaster monitoring scenario, the relationships were defined by grouping drones with

similar capabilities together for them to cover a possible ground assets that has similar capabilities. This is not always the desire for the coordination graphs. In some disaster response scenarios, you might want a drone with thermal cameras to coordinate with another drone that has another sensor modality like EO/IR or pheromone detection, because coordinating with another drone that has a thermal camera does not provide any improvement in the information being gathered. In another example, if one of the ground assets is fast moving or highly unpredictable then having a drone as part of the collaboration group that has long-range tracking capabilities like electronic sensing would be beneficial.

It might also be desirable to have the technique be progressive. Initially, a spatial clustering technique could be used to organize the coordination graph by proximity. As the drones share service capabilities, then the clustering can move towards relational learning. Finally, as the system has been fielded for the necessary amount of time for statistical tests to converge, the more complex factored MDPs with an understanding of the true value of information can facilitate the construction of a coordination graph.

Chapter 7

Conclusion

7.1 Summary

The Monte Carlo simulations confirmed two major innovative contributions in decision making under uncertainty for multi-agent autonomous systems: (i) The ability to develop a partially observable Markov decision process that can handle action-based constraints on resources; and, (ii) The eficacy of utilizing Constrained-Action POMDPs to perform Intelligent Knowledge Distribution.

The Constrained-Action POMDPs provide a level of guarantee of probabilistic constraint satisfaction to a desired operational behavior while still allowing short-term bursts of critical information. It also validates the concept of using Markov chain Monte Carlo analysis to evaluate an infinite-horizon policy represented as a finite state controller for its probability of satisfying soft constraints and a combinatorial discrete optimization for achieving desired constraint behavior of a policy while minimizing impact on the controller's value.

Secondly, the results indicate that the incorporation of Intelligent Knowledge Distribution (IKD) which determines what information is needed to whom and when under soft constraints was successful in a multi-agent system. The approach provides an *agnostic* "plug-and-play" framework for IKD by constraining the actions of a POMDP to only transmit information to a collaborative agent when the value of that information warrants the com-

munication. The disaster monitoring scenario utilized a simplified Concurrent Decentralized POMDP to validate the CA-POMDP, but the target tracking scenario highlights the construction of a model that provides for the integration of critical components of a fully autonomous system.

During the simulation, the communication links and their usage was not accurately known *a priori* until the system was in a realistically simulated dynamic environment and the resource utilization was learned online. Also, the edge determination initially assumed a uniform distribution when randomly selecting edges to follow in the controller and improving the constraint satisfaction of the controller was achieved by learning online the actual observation edge probability transitions. The coordination graph that dictated the neighbors in which an agent collaborated with was not known *a priori*, but calculated online utilizing both spatial clustering and relational learning.

7.2 Future Work

A lot of work has been done in the development and validation of *Intelligent Knowledge Distribution* through Concurrent Decentralized POMDPs and Constrained-Action POMDPs, but there is still further research is needed to ensure that it can run on small embedded systems and provide a fully agnostic plug-in-play capability during field implementation. The author concludes with a list of those improvements that has been identified so far during the execution of the work. Though not listed in the future work below, it is intended to monitor the research in named data networking as a tool that can increase the efficiency and dissemination of information in the network.

7.2.1 Anytime Performance Improvements

For IKD in real-time scenarios, the agents available to collaborate and coordinate may not be known until they are actively in the field, therefore learning heterogeneous interactions and constructing a coordination graph online is necessary. The above approach is currently actively being integrated into a fully collaborative multi-agent target tracking simulation using Concurrent Decentralized Markov Decision Processes (CoDec MDPs) to evaluate its effectiveness in Monte Carlo scenario analysis. This includes the ability to distributively learn observed behavior for improved communications in a collaboration environment, including the value of information and collaboration strategies within this CoDec MDP framework.

The algorithms for calculating the observation edge projections in policy iteration and the MCMC mixing in constraint evaluation are computationally inefficient preventing them from being utilized dynamically on a embedded platform. Therefore, the system needed to develop more computationally efficient algorithms to allow embedded computer use. There are two techniques under investigation to address computational efficiencies. The first is to utilize a point-based policy iteration algorithm that has been well researched for POMDPs in the literature. The second is more complex and requires developing an anytime algorithm for policy iteration that is capable of handling large state spaces necessary in large collaboration networks, like drone swarms. This work has already been extended to value iteration with the popular Determinized Sparse Partially Observable Tree (DESPOT) algorithm [123].

Once the policy iteration algorithm has been completed for increasing the speed of calculating a policy, the discrete optimization algorithm will be tackled to improve the tree search algorithm for faster probabilistic convergence. It is expected that providing better heuristics that are faster to calculate along with an improved branching algorithm that splits the domain more intelligently than the current approach.

7.2.2 Online & Reinforcement Learning

The initial model for Intelligent Knowledge Distribution assumed that the transition and observation models used to track relevance and collaboration in the CA-POMDP were independent. In reality there is conditional dependence between these relevance and collaboration belief states to the extent determined by the coordination graph.

The objective of this work is to learn the transition and observational conditional dependencies using probabilistic graphical models (PGMs). Initial development into this learning engine will focus on two techniques:

1. Bayes network structure learning to model and learn the conditional dependence with an MCMC search algorithm; and,

2. Autoencoders to learn a representation of the features, which are the transitions and observations, that reduces the dimensionality of their dependence.

7.2.3 Adversarial AI

Another targeted research area for pursuit after this work is looking at adversarial situations. Markov Decision Processes are considered a stochastic game with a single player, which means the analytical framework of the system can be generalized to Partially Observable Stochastic Games (POSG) to account for adversarial behavior. With the division of the system into concurrent decision making algorithms, the only adjustment would need to be in the asset monitoring or target tracking algorithm to provide more complex predictive algorithms for adversarial behavior.

7.2.4 Learning the True Value of Information & Collaboration

One of the key abilities for this approach to be feasible in the field on embedded systems is to ensure that the autonomous agents are truly learning how best to collaborate with each other. This requires that the agents actually learn the true value of information they are receiving so they can improve their understanding of how often communication needs to happen and to whom. They also need to learn who they are collaborating with is key to mission success. We initially presented three approaches to learning collaboration strategies but only evaluated two: spatial clustering and relational learning. These techniques work very well in many situations, especially if the objectives can be clearly defined *a priori*. Still more work is needed in how to construct a coordination graph for various mission profiles from relational learning.

In a truly ad hoc situations where multi-agent systems are first learning of each other during field implementation, the Factored MDP approach to learning collaboration strategies would be highly beneficial. They also need to balance exploration and exploitation of information being learned, so constructing task allocation and asset monitoring algorithms that can handle belief-dependent rewards would provide an intelligent approach to balancing the two, instead of common techniques of randomly introducing actions based upon an annealed probability

Bibliography

[1] (2019, November). [Online]. Available: https://www.darpa.mil/news-events/2018-11-30a

[2] M. Alwateer, S. Loke, and A. Zuchowicz, "Drone services: issues in drones for location-based services from human-drone interaction to information processing," *Journal of Location Based Services*, pp. 1–34, 01 2019.

[3] E. Yanmaz, M. Quaritsch, S. Yahyanejad, B. Rinner, H. Hellwagner, and C. Bettstetter, "Communication and coordination for drone networks," in *Ad Hoc Networks*. Springer, 2017, pp. 79–91.

[4] O. K. Sahingoz, "Networking models in flying ad-hoc networks (fanets): Concepts and challenges," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, pp. 513–527, 2014.

[5] L. Zhang, A. Afanasyev, J. Burke, and V. Jacobson, "K. claffy, p. crowley, c. papadopoulos, l. wang, and b. zhang,"named data networking," sigcomm comput," *Commun. Rev*, vol. 44, no. 3, pp. 66–73, 2014.

[6] M. J. Kochenderfer, C. Amato, and H. J. D. Reynolds, *Decision making under uncertainty: theory and application*. MIT press, 2015.

[7] D. S. Bernstein, C. Amato, E. A. Hansen, and S. Zilberstein, "Policy iteration for decentralized control of markov decision processes," *Journal of Artificial Intelligence Research*, vol. 34, no. 1, p. 89, 2009.

[8] Chire. (2011, October) Dbscan cluster analysis. [Online]. Available: https://en.wikipedia.org/wiki/File:DBSCAN-density-data.svg

[9] M. E. Dempsey and S. Rasmussen, "Eyes of the army–us army roadmap for unmanned aircraft systems 2010–2035," *US Army UAS Center of Excellence, Ft. Rucker, Alabma*, vol. 9, 2010.

[10] G. Tuna, B. Nefzi, and G. Conte, "Unmanned aerial vehicle-aided communications system for disaster recovery," *Journal of Network and Computer Applications*, vol. 41, pp. 27–36, 2014.

[11] D. Akselrod, A. Sinha, and T. Kirubarajan, "Collaborative distributed sensor management for multitarget tracking using hierarchical Markov decision processes," pp. 669 912–669 912–14, 2007.

[12] S. Omidshafiei, A.-a. Agha-mohammadi, C. Amato, and J. P. How, "Decentralized control of partially observable Markov decision processes using belief space macro-actions," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5962–5969.

[13] J. Girard and M. R. Emami, "Concurrent Markov decision processes for robot team learning," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 223 – 234, 2015.

[14] F. A. Oliehoek, C. Amato *et al.*, *A concise introduction to decentralized POMDPs*. Springer, 2016, vol. 1.

[15] S. Seuken and S. Zilberstein, "Improved memory-bounded dynamic programming for decentralized POMDPs," *arXiv preprint arXiv:1206.5295*, 2012.

[16] J. Capitan, M. T. Spaan, L. Merino, and A. Ollero, "Decentralized multi-robot cooperation with auctioned POMDPs," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 650–671, 2013. [Online]. Available: http://ijr.sagepub.com/content/32/6/650.abstract

[17] O. Sigaud and O. Buffet, *Markov decision processes in artificial intelligence: MDPs, beyond MDPs and applications*. London: John Wiley & Sons, 2010.

[18] P. Hintjens, *ZeroMQ: messaging for many applications*. " O'Reilly Media, Inc.", 2013.

[19] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[20] A. Beynier and A.-I. Mouaddib, "An iterative algorithm for solving constrained decentralized Markov decision processes," in *AAAI*, vol. 6, 2006, pp. 1089–1094.

[21] M. El Chamie and B. Acikmese, "Finite-Horizon markov decision processes with state constraints," *arXiv preprint arXiv:1507.01585*, 6 Jul. 2015.

[22] S. Feyzabadi and S. Carpin, "HCMDP: A hierarchical solution to constrained Markov decision processes," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 3971–3978.

[23] ——, "Risk-aware path planning using hirerachical constrained Markov decision processes," in *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*. IEEE, 2014, pp. 297–303.

[24] E. A. Hansen, "Finite-memory control of partially observable systems," Ph.D. dissertation, University of Massachusets Amherst, 1998.

[25] J. D. Isom, S. P. Meyn, and R. D. Braatz, "Piecewise linear dynamic programming for constrained POMDPs," in *AAAI*, 2008, pp. 291–296.

[26] A. Undurti and J. P. How, "An online algorithm for constrained POMDPs," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3966–3973.

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

[28] S. A. Williamson, E. H. Gerding, and N. R. Jennings, "Reward shaping for valuing communications during multi-agent coordination," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 641–648.

[29] M. Araya, O. Buffet, V. Thomas, and F. Charpillet, "A pomdp extension with belief-dependent rewards," in *Advances in neural information processing systems*, 2010, pp. 64–72.

[30] S. Williamson, E. Gerding, and N. Jennings, "A principled information valuation for communications during multi-agent coordination," in *Proc AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains*, 2008, pp. 137–151.

[31] M. T. Spaan, T. S. Veiga, and P. U. Lima, "Decision-theoretic planning under uncertainty with information rewards for active cooperative perception," *Autonomous Agents and Multi-Agent Systems*, vol. 29, no. 6, pp. 1157–1185, 2015.

[32] H. Fargier and J. Lang, "Uncertainty in constraint satisfaction problems: a probabilistic approach," in *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*. Springer, 1993, pp. 97–104.

[33] A. Mesbah, S. Streif, R. Findeisen, and R. D. Braatz, "Stochastic nonlinear model predictive control with probabilistic constraints," in *American Control Conference (ACC), 2014*. IEEE, 2014, pp. 2413–2419.

[34] T. Schiex, "Possibilistic constraint satisfaction problems or "how to handle soft constraints?"," in *Uncertainty in Artificial Intelligence, 1992*. Elsevier, 1992, pp. 268–275.

[35] T. Walsh, "Stochastic constraint programming," in *ECAI*, vol. 2, 2002, pp. 111–115.

[36] L. Ng and M. R. Emami, "Concurrent individual and social learning in robot teams," *Computational Intelligence*, vol. 32, no. 3, pp. 420–438, 2016.

[37] I. F. Akyildiz and X. Wang, "A survey on wireless mesh networks," *Communications Magazine, IEEE*, vol. 43, no. 9, pp. S23–S30, 2005.

[38] D. Outreach, "Darpa seeks clean-slate ideas for mobile ad hoc networks (manets)," *Defense Advanced Research Projects Agency*, April 2013. [Online]. Available: https://www.darpa.mil/news-events/2013-04-30

[39] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati, "Cloud of things for sensing as a service: sensing resource discovery and virtualization," in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015, pp. 1–7.

[40] F. Jiang and A. L. Swindlehurst, "Dynamic uav relay positioning for the ground-to-air uplink," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*. IEEE, 2010, pp. 1766–1770.

[41] O. K. Sahingoz, "Multi-level dynamic key management for scalable wireless sensor networks with uav," in *Ubiquitous Information Technologies and Applications*. Springer, 2013, pp. 11–19.

[42] J. G. Manathara, P. Sujit, and R. W. Beard, "Multiple uav coalitions for a search and prosecute mission," *Journal of Intelligent & Robotic Systems*, vol. 62, no. 1, pp. 125–158, 2011.

[43] G. York and D. J. Pack, "Ground target detection using cooperative unmanned aerial systems," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 473–478, 2012.

[44] S. Zhu, D. Wang, and C. B. Low, "Ground target tracking using uav with input constraints," *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1-4, pp. 417–429, 2013.

[45] 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. 1-4, pp. 533–548, 2012.

[46] I. Maza, F. Caballero, J. Capitán, J. R. Martínez-de Dios, and A. Ollero, "Experimental results in multi-uav coordination for disaster management and civil security applications," *Journal of intelligent & robotic systems*, vol. 61, no. 1-4, pp. 563–585, 2011.

[47] H. Xiang and L. Tian, "Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (uav)," *Biosystems engineering*, vol. 108, no. 2, pp. 174–190, 2011.

[48] L. P. Root, "Squad x core technologies (sxct)," *Defense Advanced Research Projects Agency*, vol. https://www.darpa.mil/program/squad-x-core-technologies, 2018.

[49] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[50] (2019, June). [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS45213219

[51] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[52] N. Koshizuka and K. Sakamura, "Ubiquitous id: standards for ubiquitous computing and the internet of things," *IEEE Pervasive Computing*, vol. 9, no. 4, pp. 98–101, 2010.

[53] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, and T. C. Schmidt, "Riot os: Towards an os for the internet of things," in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*. IEEE, 2013, pp. 79–80.

[54] S. Kraijak and P. Tuwanut, "A survey on iot architectures, protocols, applications, security, privacy, real-world implementation and future trends," 2015.

[55] K. Sood, S. Yu, and Y. Xiang, "Software-defined wireless networking opportunities and challenges for internet-of-things: A review," *IEEE Internet of Things Journal*, vol. 3, no. 4, pp. 453–463, 2016.

[56] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke, "Ieee 802.11 s: the wlan mesh standard," *Wireless Communications, IEEE*, vol. 17, no. 1, pp. 104–111, 2010.

[57] M. Dixit, R. Kumar, and A. K. Sagar, "Vanet: Architectures, research issues, routing protocols, and its applications," in *2016 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 2016, pp. 555–561.

[58] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying ad-hoc networks (fanets): A survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[59] K. Yang, J.-f. Ma, and Z.-h. Miao, "Hybrid routing protocol for wireless mesh network," in *Computational Intelligence and Security, 2009. CIS'09. International Conference on*, vol. 1. IEEE, 2009, pp. 547–551.

[60] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: https://doi.org/10.1145/1355734.1355746

[61] S.-H. Yun and H. Dempo, "Clout networking solutions: Openstack and openflow," NEC, Presentation, October 2011.

[62] (2020, April). [Online]. Available: https://www.openstack.org

[63] I. T. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: A survey and taxonomy," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2713–2737, 2016.

[64] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[65] L. Silva, P. Gonçalves, R. Marau, P. Pedreiras, and L. Almeida, "Extending openflow with flexible time-triggered real-time communication services," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2017, pp. 1–8.

[66] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT

'09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 1–12. [Online]. Available: https://doi.org/10.1145/1658939.1658941

[67] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, Jul. 2014. [Online]. Available: https://doi.org/10.1145/2656877.2656887

[68] Y. Zhang, A. Afanasyev, J. Burke, and L. Zhang, "A survey of mobility support in named data networking," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, Apr. 2016. [Online]. Available: https://doi.org/10.1109/infcomw.2016.7562050

[69] J.-H. Cha, J.-H. Choi, J.-Y. Kim, Y.-H. Han, and S.-G. Min, "A mobility link service for NDN consumer mobility," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–8, Jul. 2018. [Online]. Available: https://doi.org/10.1155/2018/5149724

[70] H. Khelifi, S. Luo, B. Nour, H. Moungla, Y. Faheem, R. Hussain, and A. Ksentini, "Named data networking in vehicular ad hoc networks: State-of-the-art and challenges," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2019. [Online]. Available: https://doi.org/10.1109/comst.2019.2894816

[71] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 3, p. 62, Jun. 2012. [Online]. Available: https://doi.org/10.1145/2317307.2317319

[72] J. V. Torres, I. D. Alvarenga, R. Boutaba, and O. C. M. Duarte, "An autonomous and efficient controller-based routing scheme for networking named-data mobility," *Computer Communications*, vol. 103, pp. 94–103, May 2017. [Online]. Available: https://doi.org/10.1016/j.comcom.2017.02.001

[73] O. Sigaud and O. Buffet, *Markov decision processes in artificial intelligence: MDPs, beyond MDPs and applications*. London: ISTE ; Hoboken, NJ : Wiley, 2010.

[74] Q. Hu and W. Yue, *Markov decision processes with their applications*. Springer Science & Business Media, 2007, vol. 14.

[75] J. Pineau, G. Gordon, S. Thrun *et al.*, "Point-based value iteration: An anytime algorithm for POMDPs," in *IJCAI*, vol. 3, 2003, pp. 1025–1032.

[76] B. Bakker, Z. Zivkovic, and B. Krose, "Hierarchical dynamic programming for robot path planning," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 2756–2761.

[77] M. J. Kochenderfer, C. Amato, and H. J. D. Reynolds, *Decision making under uncertainty: theory and application*. MIT press, 2015.

[78] F. A. Oliehoek, "Decentralized POMDPs," in *Reinforcement Learning*. Springer, 2012, pp. 471–503.

[79] S. P. Singh, T. Jaakkola, and M. I. Jordan, "Learning without state-estimation in partially observable markovian decision processes," in *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 284–292.

[80] D. Szer and F. Charpillet, "An optimal best-first search algorithm for solving infinite horizon dec-pomdps," in *European Conference on Machine Learning*. Springer, 2005, pp. 389–399.

[81] C. Amato, B. Bonet, and S. Zilberstein, "Finite-state controllers based on mealy machines for centralized and decentralized pomdps." in *AAAI*, 2010.

[82] C. Amato, D. S. Bernstein, and S. Zilberstein, "Optimizing fixed-size stochastic controllers for pomdps and decentralized pomdps," *Autonomous Agents and Multi-Agent Systems*, vol. 21, no. 3, pp. 293–320, 2010.

[83] D. S. Bernstein, E. A. Hansen, and S. Zilberstein, "Bounded policy iteration for decentralized pomdps," in *Proceedings of the nineteenth international joint conference on artificial intelligence (IJCAI)*, 2005, pp. 52–57.

[84] H. Ahmadzadeh and E. Masehian, "Fuzzy coordination graphs and their application in multi-robot coordination under uncertainty," in *Robotics and Mechatronics (ICRoM), 2014 Second RSI/ISM International Conference on*. IEEE, 2014, pp. 345–350.

[85] Y. Li, K. Gupta, and S. Payandeh, "Motion planning of multiple agents in virtual environments using coordination graphs," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 378–383.

[86] N. Vlassis, R. Elhorst, and J. R. Kok, "Anytime algorithms for multiagent decision making using coordination graphs," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 1. IEEE, 2004, pp. 953–957.

[87] C. V. Goldman and S. Zilberstein, "Optimizing information exchange in cooperative multi-agent systems," in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '03. New York, NY, USA: ACM, 2003, pp. 137–144. [Online]. Available: http://doi.acm.org/10.1145/860575.860598

[88] S. A. Williamson, E. H. Gerding, and N. R. Jennings, "A principled information valuation for communications during multi-agent coordination," in *Proceedings of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains*, 2007, pp. 137–151.

[89] M. Roth, R. Simmons, and M. Veloso, "Reasoning about joint beliefs for execution-time communication decisions," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*. ACM, 2005, pp. 786–793.

[90] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[91] J. Clausen, "Branch and bound algorithms –principles and examples," *Department of Computer Science, University of Copenhagen*, pp. 1–30, 1999.

[92] B. Taylor, "Integer programming: The branch and bound method," *Introduction to Management Science*, 2009.

[93] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," *IEEE transactions on Automatic Control*, vol. 49, no. 9, pp. 1453–1464, 2004.

[94] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[95] W. R. Ashby and J. Pierce, "An introduction to cybernetics," *Physics Today*, vol. 10, no. 7, pp. 34–36, 1957.

[96] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[97] D. Pelleg and A. Moore, "Accelerating exact k-means algorithms with geometric reasoning," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '99. New York, NY, USA: Association for Computing Machinery, 1999, pp. 277–281. [Online]. Available: https://doi.org/10.1145/312129.312248

[98] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[99] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[100] L. RDUSSEEUN and P. J. KAUFMAN, "Clustering by means of medoids," 1987.

[101] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: Why and how you should (still) use dbscan," *ACM Trans. Database Syst.*, vol. 42, no. 3, Jul. 2017. [Online]. Available: https://doi.org/10.1145/3068335

[102] E. Davis, *Representations of commonsense knowledge*. Morgan Kaufmann, 2014.

[103] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda, "Learning systems of concepts with an infinite relational model," in *AAAI*, vol. 3, 2006, p. 5.

[104] C. E. Rasmussen, "The infinite gaussian mixture model," in *Advances in neural information processing systems*, 2000, pp. 554–560.

[105] J. Pitman *et al.*, "Combinatorial stochastic processes," Technical Report 621, Dept. Statistics, UC Berkeley, 2002. Lecture notes for …, Tech. Rep., 2002.

[106] S. Jain and R. M. Neal, "A split-merge markov chain monte carlo procedure for the dirichlet process mixture model," *Journal of computational and Graphical Statistics*, vol. 13, no. 1, pp. 158–182, 2004.

[107] C. Kemp, J. B. Tenenbaum, S. Niyogi, and T. L. Griffiths, "A probabilistic model of theory formation," *Cognition*, vol. 114, no. 2, pp. 165–196, 2010.

[108] 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.

[109] C. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored mdps," in *Advances in neural information processing systems*, 2002, pp. 1523–1530.

[110] U. Bertele and F. Brioschi, *Nonserial dynamic programming*. Academic Press, 1972.

[111] J. R. Kok and N. Vlassis, "Using the max-plus algorithm for multiagent decision making in coordination graphs," in *Robot Soccer World Cup*. Springer, 2005, pp. 1–12.

[112] ——, "Sparse cooperative q-learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 61.

[113] J. R. Kok, E. J. Hoen, B. Bakker, and N. Vlassis, "Utile coordination: Learning interdependencies among cooperative agents," in *EEE Symp. on Computational Intelligence and Games, Colchester, Essex*, 2005, pp. 29–36.

[114] K. A. Pituch, T. A. Whittaker, and J. P. Stevens, *Intermediate statistics: A modern approach*. Routledge, 2013.

[115] S. Koenig and M. Likhachev, "D^* lite," *Aaai/iaai*, vol. 15, 2002.

[116] H. Musoff and P. Zarchan, *Fundamentals of Kalman filtering: a practical approach*. American Institute of Aeronautics and Astronautics, 2009.

[117] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, vol. 3068. International Society for Optics and Photonics, 1997, pp. 182–194.

[118] R. Van Der Merwe, "Sigma-point kalman filters for probabilistic inference in dynamic state-space models," 2004.

[119] S. J. Julier, "The scaled unscented transformation," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 6. IEEE, 2002, pp. 4555–4559.

[120] D. J. Thomas, K. M. Nastasi, K. Schroeder, and J. T. Black, "Autonomous multi-phenomenology space domain sensor tasking and adaptive estimation," in *2018 21st International Conference on Information Fusion (FUSION)*, July 2018, pp. 1331–1338.

[121] K. M. Nastasi and J. Black, "An autonomous sensor management strategy for monitoring a dynamic space domain with diverse sensors," in *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 0890.

[122] M. Alexa T., "An upper-level ontology for the biomedical domain." *Comparative and Functional Genomics*, no. 1, p. 80, 2003. [Online]. Available: http://login.ezproxy.lib.vt.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edsdoj&AN=edsdoj.92686ed3a327469691a5aebc426290c6&site=eds-live&scope=site

[123] N. Ye, A. Somani, D. Hsu, and W. S. Lee, "DESPOT: online POMDP planning with regularization," *CoRR*, vol. abs/1609.03250, 2016. [Online]. Available: http://arxiv.org/abs/1609.03250