2022

# Improving Robotic Decision-Making in Unmodeled Situations

Nicholas Scott Ohi
*West Virginia University*, nohi@mix.wvu.edu

# *Improving Robotic Decision-Making in Unmodeled Situations*

Nicholas S. Ohi

Dissertation submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in
Aerospace Engineering

Yu Gu, Ph.D., Chair
Jason N. Gross, Ph.D.
Guilherme A. S. Pereira, Ph.D.
Natalia A. Schmid, Ph.D.
Gianfranco Doretto, Ph.D.

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia
2022

Keywords: Decision-Making under Ambiguity, Autonomous Robotics

Abstract

# *Improving Robotic Decision-Making in Unmodeled Situations*

Nicholas S. Ohi

Existing methods of autonomous robotic decision-making are often fragile when faced with inaccurate or incompletely modeled distributions of uncertainty, also known as *ambiguity*. While decision-making under ambiguity is a field of study that has been gaining interest, many existing methods tend to be computationally challenging, require many assumptions about the nature of the problem, and often require much prior knowledge. Therefore, they do not scale well to complex real-world problems where fulfilling all of these requirements is often impractical if not impossible. The research described in this dissertation investigates novel approaches to robotic decision-making strategies which are resilient to ambiguity that are not subject to as many of these requirements as most existing methods. The novel frameworks described in this research incorporate *physical feedback*, *diversity*, and *swarm local interactions*, three factors that are hypothesized to be key in creating resilience to ambiguity. These three factors are inspired by examples of robots which demonstrate resilience to ambiguity, ranging from simple vibrobots to decentralized robotic swarms. The proposed decision-making methods, based around a proposed framework known as *Ambiguity Trial and Error* (AT&E), are tested for both single robots and robotic swarms in several simulated robotic foraging case studies, and a real-world robotic foraging experiment. A novel method for transferring swarm resilience properties back to single agent decision-making is also explored. The results from the case studies show that the proposed methods demonstrate resilience to varying types of ambiguities, both stationary and non-stationary, while not requiring accurate modeling and assumptions, large amounts of prior training data, or computationally expensive decision-making policy solvers. Conclusions about these novel methods are then drawn from the simulation and experiment results and the future research directions leveraging the lessons learned from this research are discussed.

# Contents

# List of Figures

# List of Tables

# Acknowledgments

I have been very fortunate to receive a tremendous amount of support and mentorship throughout my graduate career and I would like to thank a few of the key people who have helped me along the way.

First, I would like to thank **Dr. Yu Gu** , for all of his help and support over the years. From encouraging me to go to graduate school, to our inspiring brainstorming discussions, and much more, your support as my advisor has been invaluable. I would also like to thank **Dr. Jason Gross** for giving me the opportunity to get involved in research as an undergraduate and supporting my transition from undergraduate into graduate school.

In addition, I would like to thank **Dr. Guilherme Pereira** , **Dr. Natalia Schmid** , and **Dr. Gianfranco Doretto** for their support, instruction, and insightful advice.

And finally, I would also like to acknowledge the **National Science Foundation Graduate Research Fellowship Program (NSF GRFP)** , the **NASA West Virginia Space Grant Consortium (WVSGC)** , the **NASA Established Program to Stimulate Competitive Research (EPSCoR)** , and the **West Virginia Higher Education Policy Commision (WV HEPC)** for funding the research presented in this dissertation.

# Contributing Papers

Portions of the discussion and results presented in this document are originally from previous publications. The publications and the chapters to which they contribute are listed below.

**Ch. 3** Y. Gu, **N. Ohi** , K. Lassak, J. Strader, L. Kogan, A. Hypes, S. Harper, B. Hu, M. Gramlich,R. Kavi, et al., "Cataglyphis: An autonomous sample return rover," *Journal of Field Robotics*, vol. 35, no. 2, pp. 248–274, 2018.

**Ch. 3** Y. Gu, J. Strader, **N. Ohi** , S. Harper, K. Lassak, C. Yang, L. Kogan, B .Hu, M. Gramlich, R. Kavi, et al., "Robot foraging: Autonomous sample return in a large outdoor environment," *IEEE Robotics  Automation Magazine*, vol. 25, no. 3, pp. 93–101, 2018.

**Ch. 3** **N. Ohi** , K. Lassak, R. Watson, J. Strader, Y. Du, C. Yang, G. Hedrick, J. Nguyen, S. Harper, D. Reynolds, et al., "Design of an autonomous precision pollination robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pp. 7711–7718, IEEE, 2018.

# 1

# Introduction

## 1.1 Goal and Motivations

The goal of this research is to investigate new approaches to autonomous robotic decision-making that do not rely on many of the assumptions required by most existing decision-making frameworks. Of primary interest is relaxing assumptions about the accuracy of the robot's prior knowledge about the distributions of uncertainty involved in the decision-making problem. This is in an effort towards making robot autonomy more generalizable to real-world situations, as opposed to highly structured and controlled environments custom designed to cater to the robot's limitations. Most existing methods of autonomous robotic decision-making rely on both accurate perception of all relevant information to the decision-maker, as well as complete, accurate models of the robot, its environment, and the associated distributions of uncertainty. Alternatively, if such models are not known, then it is assumed that they can be learned accurately, and it is assumed that the structure of the learned model will accurately capture all necessary parameters to fully describe the decision-making problem and the distributions of uncertainty involved. Then, traditional "rational," top-down decision-making can be performed to provide *optimality*. It is the author's assertion, however, that the real-world is far too complex to model every source of uncertainty accurately and thereby assuming that all relevant information needed to make rational, top-down decisions is available. Some portions of the problem can likely be modeled a priori and other portions may be able to be learned online, but for many real-world tasks, it is unlikely the full problem can be accurately modeled. There will always be unhandled edge cases and lack of complete and accurate knowledge about the distributions of uncertainty. This is a concept that may be referred to as second-order uncertainty, Knightian uncertainty, or *ambiguity* [1, 2]. This inability of most existing decision-making frameworks to perform reliably when faced with ambiguity limits the applicability of autonomy to many real-world situations. Explicitly designing decision-making policies to account for every possibility in complex real-world situations is infeasible. New types of frameworks are needed that do not fully rely on these "traditional" approaches in order to provide better "resilience to ambiguity."

As a starting point to properly motivate the specific objectives of this research, inspiration is taken from various robotics topics, some very different than robotic decision-making, which demonstrate special properties hypothesized by the author to be key in working towards the goal of this research. The first example, while a very different subject than robotic decision-making, is "bristle-

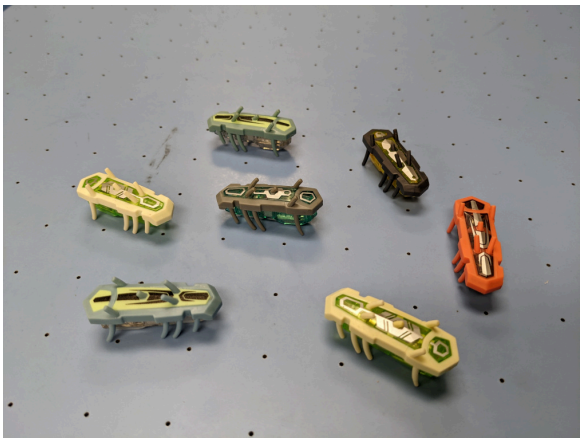


**Figure 1.1.1:** Examples of vibrobots.

bots" or "vibrobots," an example of which is shown in Fig. 1.1.1. These small robots move around by using a vibrator in combination with springy bristles on the bottom of the robot. The motion can be directed by designing the shape of the springy bristles to favor a particular direction. While typically used for nothing more than simple children's toys, these robots display a remarkable resilience to getting stuck and becoming unable to move, even in very cluttered and irregular environments. They have no programming or high-level reasoning to control their behavior, other than their single mode of vibrational motion and no prior knowledge or remote sensing of their environments, but nevertheless are able to move about almost any flat but cluttered environment, full of obstacles using one simple design that makes use of **physical feedback** from the environment. This offloading of computing to the interaction between the body and the environment, instead of a centralized "brain" or controller, is often referred to as morphological computing [3]. This ability to rely primarily on physical feedback and generalize to many other environments, using designs that do not encode vast amounts of prior knowledge, is highly inspiring for this research.

Another example that is often resilient to ambiguity is distributed robotic swarms. As they move around the environment and encounter each other, swarm agents may be influenced by **local interactions** with neighboring agents. These local interactions, however simple or complex they may be, result in an exchange of information between swarm agents and can create "emergent intelligence" or "emergent behaviors" that are not explicitly designed into any of the swarm agent's decision-making algorithms.

And finally, another interesting example is the approach taken by Cully, et al. known as Intelligent Trial and Error (IT&E) [4]. Using this approach, walking hexapod robots are able to adapt to new ways of walking if the robot suffers damage by attempting different, **diverse** walking strategies provided as prior knowledge and finding which ones are successful. Offline learning of diverse walking behaviors is provided to the robot, but adjusting its walking behavior is done quickly online and without pre-programmed contingency plans for any specific failure modes. This is done through an informed *trial and error* method, without any attempt to directly model the damage the robot has suffered, which may be infinitely variable and impossible to model accurately in a "catch-all" sense. This ability to be resilient to unknown failure modes and continue operating, even in a limited capacity, without failing outright, is a key innovation. The use of the informed trial and error method is also very inspiring.

## 1.2 Research Objective

It is the author's hypothesis that these three factors: **physical feedback**, **local interactions**, and **diversity** all play key roles towards improving autonomous robotic decision-making under ambiguity. This has been considered somewhat in the context of robotic swarms, though research focused directly on the aspect of resilience to ambiguity is lacking. Additionally, there are even fewer examples of research into resilience to ambiguity for single agents, as opposed to swarms. These gaps in existing research motivate the objective of the research presented in this document, which is the following:

> Investigate autonomous robotic decision-making techniques that are resilient to *ambiguity*, which is defined as a lack of complete and accurate knowledge about the distributions of uncertainty involved in the problem.

The types of ambiguities investigated in this research include stochastic state transition dynamics for which 1) the values of the parameters that define distributions of uncertainty are unknown and 2) the unknown parameters may also vary parametrically with the state (i.e., position and/or time dependent). It is the author's hope that the outcomes of this research will enable further research into the creation of robotic decision-makers that are more resilient to ambiguity than existing frameworks.

## 1.3 Expected Outcomes and Broader Impacts

The expected outcomes of this research are both theoretical contributions to the study of robotic decision-making, as well as more application focused results with prototype frameworks developed to solve a variety of case study problems. In terms of theory, the expected outcomes are to provide insights into the concept of resilience to ambiguity for robotic decision-making. These insights will provide a foundation on which future work on this topic may leverage. On the application side, the expected outcomes are prototype decision-making frameworks and results from case-study problems, focused primarily on robotic foraging, demonstrating the benefits of the techniques presented in this research, in terms of resilience to ambiguity. Specifically, this includes a trial and error based decision-making framework named Ambiguity Trial and Error (AT&E) that can be applied to complex decision-making under ambiguity problems without These case-studies may serve as benchmarks for evaluating future work on these topics, and the frameworks as baseline techniques upon which further improvements can be demonstrated.

The broader impacts of this research relate back to the initial motivations: making robot autonomy more generalizable to real-world environments and situations, instead of only being used with close human supervision in highly controlled environments like laboratories, warehouses, and factories. Many robots outside of these setting are very fragile in terms of their decision-making, and therefore are not entrusted with important or complex tasks, without a human expert actively monitoring the situation. Tremendous amounts of labor-intensive and cost-intensive work is required to make robots capable of performing even very simple tasks autonomously, in real-world settings. This includes detailed modeling of environments robot dynamics, and distributions of uncertainty, as well as testing and validation, which must be performed by large teams of skilled robotics experts. As argued already, it is the author's assertion that no amount of detailed modeling can account for every foreseeable possibility and source of uncertainty in many real-world scenarios. Even very thorough testing and validation is also unlikely to expose every possible failure mode or edge case that could be encountered during operation. Attempting to achieve more complex autonomy through simply trying to apply more explicit modeling of every possibility and more exhaustive testing results in diminishing returns. It is desired that the new approaches explored through this research will lay the foundation for new ways of thinking about and designing robot autonomy that alleviate some of the rigid assumptions of "traditional methods," and enable

autonomous robots to operate more reliably and flexibly in real-world scenarios. The benefits of higher reliability robot decision-making is that autonomous robots will require less human operator supervision, can be trusted to perform more complex tasks on their own, and can be expected to operate autonomously for longer periods of time before receiving new instructions or aid from humans. This will increase the productivity of both robot-only tasks and human-robot collaborative tasks, with less supervision from expert human operators being required. This will allow the use of autonomous robots to be extended to many new domains and will increase the effectiveness of the the robots in the domains in which they are already used.

## 1.4 Challenges and Evaluation of Results

One of the biggest challenges of this research is actually one of human nature and not of robotics at all. Instead, the challenge is that the case-study problems must be designed in such a way that they will present the robotic decision-makers with ambiguities that are known to the human designer, but do not impact the human designer's interpretation of the results. When evaluating the results, the human designer must not allow their knowledge of the nature of the ambiguities to introduce bias and influence their design of the decision-making algorithms in any way that specifically accounts for these ambiguities. To do so would be in direct contradiction with the objectives of this research. This is challenging though, because it is human nature to want to "solve problems" about which we have foreknowledge. And in most cases, for real applications, there is nothing wrong with doing so. In fact, any prior knowledge that can be leveraged to improve the robot's decision-making and help it predict things that can be accurately anticipated can and should be used. The point being made in the motivation for this research, however, is simply that this use of prior knowledge when designing the decision-making algorithms is not enough to provide accurate logic about what the robot should do in every possible situation. The prior knowledge encoded into the decision-maker through its design may successfully handle many situations, but no amount of detailed prior knowledge from the human designers will be sufficient to cover every possibility and source of uncertainty. Returning to the point about this being a challenge for this research, however, care must be taken by both the author, when designing these case-study problems and their solution methods, and by the reader, when interpreting the results and conclusions of these case-studies. The author must not allow their foreknowledge of the nature of the ambiguities built into the case-studies to

bias their design of the decision-making algorithms used by the robots and the reader must not bias their interpretation of the results by knowing that the ambiguities could have been specifically addressed. Demonstrating resilience to ambiguities that are not specifically addressed is the desired outcome. Some provisions must be made to address certain aspects of the problem, however, to make the case studies functional. Therefore, the author must clearly describe which aspects of the problem are known to the decision-making algorithms as prior knowledge and which aspects are not so that the case studies are correctly designed to investigate their intended hypotheses.

Another challenge is determining how to correctly evaluate the case-study results and derive meaningful metrics. There are few examples of works that attempt to generate similar types of results in literature; therefore there is not a well-established basis that is universally understood as "common practice" for evaluating these types of problems. Results are generated through several robotic decision-making case-study problems, both simulated and experimental, the details of which are described in later chapters. While the interpretation of the results is specific to each case study, a primary metric of interest across all of these case studies is "resilience to ambiguity," as defined previously. Resilience is inherently a relative concept and cannot be "measured" in an absolute sense, however, so there must be a point of comparison to consider as a baseline for each case study. Existing decision-making methods, based on more traditional approaches, are implemented to provide a baseline for each case-study. The applicable baseline methods are specific to each case study problem, and are described specifically for each. It is important to keep in mind that neither the prototype methods developed through this research, nor the baseline methods, may provide "optimal" or even "near-optimal" solutions to the decision-making problems to which they are applied. Optimality is a condition that implies both the availability of all necessary information and accurate models that the decision-maker can use to predict likely outcomes and make fully "rational" decisions. Violations of these conditions are the main premises of this research and therefore, this must be kept in mind when interpreting the results.

## 1.5 Scope of Research

It is important to point out the limitations of the scope of this research, however. This research is not intended to work towards solving the problem of general artificial intelligence (AI), where embodied AI (i.e., sentient robots) can reason at the level of human intelligence, perform complex

real-world tasks that currently can only be performed by humans, and communicate and interact with humans as another sentient being. Nor is the intent to enable robots to reason at the level of animals such as mammals, birds, or insect colonies. The potential applications of this research will still be limited to very constrained decision-making under uncertainty problems where the robot(s) can perform only a small set of actions and reason over only a small number of states, likely fewer than necessary to accurately describe the full decision-making problem and all related factors. One key example application, which is used for the case studies presented later, is robotic foraging tasks, where robots must search for food, retrieve it, and return it to a home base. The incremental step this research intends to provide is to reduce the dependence of robotic decision-making problems of this level of complexity on the assumption that it has complete and accurate knowledge about the distributions of uncertainty involved in the decision-making problem. There are still many assumptions and limitations involved with the prototype frameworks developed through this research and it is not presumed that these methods will enable robots to operate reliably with all variations of ambiguity in any general sense. All scientific progress happens incrementally, and it is the author's hope that this research serves as a starting point for future work into this topic, which as of this moment is under-explored.

## 1.6 Summary of Research Contributions and Innovations

For ease of reference, the contributions and innovations of this research are summarized here.

- Provide insights into the concept of resilience to ambiguity for robotic decision-making.
- Develop a prototype decision-making framework known as Ambiguity Trial and Error (AT&E).
- Evaluate the AT&E framework by comparing it to baseline methods for a variety of robotic foraging case-study scenarios, for both swarms and single agents, which demonstrate that AT&E has improved resilience to *ambiguity*.
  - *Ambiguity* is defined as a lack of complete and accurate knowledge about the distributions of uncertainty involved in the problem.

## 1.7 Dissertation Outline

The remainder of this document is organized as follows. Literature review is presented in Chapter 2. Preliminaries on autonomous robotic decision-making, summarizing relevant existing methods and presenting examples from the author's prior work is presented in Chapter 3. The proposed methodologies, simulation and experimental results from case studies implementing the proposed methods, and discussion analyzing these results is presented in Chapter 4. And finally, concluding remarks and discussions on future research directions is presented in Chapter 5.

# 2

# Literature Review

Making decisions can be loosely described as the process of evaluating what you know about the current situation and choosing to take a particular course of action that you expect will achieve the best outcome. For humans, we do this almost implicitly for most aspects of our lives, without needing a detailed understanding of the mechanisms by which we do so. Enabling automated machines, computer algorithms, and autonomous robots to make decisions, even about very small, constrained problems, however, has required considerable research over many decades.

## 2.1 Decision-Making Foundations

Much of the basis for modern decision-making theory comes from the field of operations research, which was first recognized as a field of study during the early 20th century, due to the military needs of World War II [5, 6]. Work on operations research was intended to provide a "quantitative basis" for people in executive positions to guide their decisions and choose the best course of action for the inherently complex operations under their control. Examples of such operations include the deployment of military forces [5, 6], planning and management of cities [7], and inventory management and scheduling [8]. Many of the mathematical formalisms about game theory and optimization problems, which are widely used today, originated from works done as a part of operations research. All of this work was focused on providing information to *human* executives to make informed decisions, however. As technology progressed, the concept of computing began to emerge and these formalisms were then taken up by research into automata theory [9, 10], which laid the foundation for much of modern computer science, long before anything resembling modern digital computers existed. Many of the frameworks developed through this research form the basis of most modern computer algorithms. These include finite state machines [11, 12], temporal logic [13, 14], dynamic logic [15], and more.

## 2.2 Robotic Decision-Making and Planning

Moving into the field of early robotics, these frameworks and decision-making formalisms served as the basis for much of early robot autonomy. One prime example of an autonomous mobile robot is the robot Shakey, which was designed to navigate and move boxes in an indoor office environment [16]. Shakey's decision-making capabilities were primarily based on a planner known as STRIPS,

which is a formal "action language" solver used to plan actions to be taken in order to transition through various "world models" (i.e., states) in order to arrive at the desired goal state [17]. The way this was implemented on Shakey is in what is known as the *sense-plan-act* paradigm [18]. Sense-plan-act works by first sensing the environment to determine the current state of the robot and the environment, planning a sequence of actions that will transition the robot through a set of states that will end with reaching the goal state, and then executing this sequence of actions. Another important planning algorithm in this category, used extensively in robotics is Dijkstra's Algorithm [19], which is used to find the shortest path between two vertices in a graph. Dijkstra's Algorithm serves as the basis for many robotic planning frameworks.

The methods for searching through the state space to find the correct sequence of actions to arrive at the goal state range from exhaustive search methods such as bredth-first and depth-first search, to heuristic search methods to improve efficiency. Some key heuristic algorithms are A* [20] and D* [21], which are derivatives of Dijkstra's Algorithm. These heuristic search algorithms may provide optimal solutions, given certain assumptions, but they are generally not guaranteed to be able to find the "globally optimal" solution, with respect to the problem's objective function in all cases.

If the problem can be broken down and solved as a set of recursive sub-problems (i.e., the Bellman equation), then a globally optimal solution, for every possible state can be found, using Dynamic Programming [22–24]. Regardless of which of these planning methods is used, however, in the sense-plan-act paradigm, execution of the plan (the "act" step) occurs blindly, assuming that the sensing step determined the initial state accurately, that the model defining the state transitions is completely accurate, and that there are no dynamic elements in the environment. Additionally, if the state space is large, significant time may be required for the planner to search through large numbers of states and construct a plan, even using more efficient search algorithms. These limitations led to the development of alternative approaches, including *reactive planning*, or *sense-act* [18]. For reactive planning, instead of blindly following a full start-to-end plan, computed once at the beginning of the task, actions are generated directly in response to sensing. This occurs very quickly in response to dynamic or even potentially unknown situations, but this only works for very simple problems, where instantaneous *greedy* decisions are likely to solve the problem well. Many classes of decision-making problems are more complex and require deliberation on possible future states, rather than simply reacting to the instantaneous sensing input. A large number of approaches exist

for combining deliberative and reactive decision-making, including behavior-based robotics [25] and layered architectures [26–29].

## 2.3 Decision-Making under Uncertainty

Many of the methods discussed above (reactive planning being an exception in some cases) rely on one key assumption, however, that is often not valid in many real-world situations. This is that the outcome of any action taken by the decision-maker is *deterministic*, meaning that the same outcome always occurs. In many real-world situations, however, the outcome of actions may be *stochastic*, or random. One of the most important stochastic decision-making paradigms is the Markov Decision Process (MDP). MDPs are an extension of Markov Chains, which represent a stochastic process, where the probability distribution function (or probability mass function for discrete problems) specifying the probabilities of the possible state transitions depend only on the current state and not on any previous state. This "historyless-ness" is known as the Markov Property. Non-Markovian processes exist as well, where the state transition probabilities depend on the full history of prior states. This framework allows for the description of more complex stochastic problems than Markov processes, but due to the need to track the entire state transition history, decision-makers using this framework suffer from the curse of history and quickly become computationally intractable. Further investigations into the theoretical aspects of solving non-Markovian problems are beyond the scope of this research, however.

Returning to MDPs, as an extension of Markov Chains, MDPs add another dimension to the specification of the transition probability, which is an *action* that can be selected by the decision-maker to influence the outcome. The outcome is still stochastic, but it is now conditionally dependent on the action chosen. Additionally, since there is now the possibility of choosing actions, a *reward function* is required to quantify the value of taking every possible action, given every possible state. Therefore, an MDP is fully specified by its state space $S$, action space $A$, transition function $T$, and reward function $R$, as $\langle S, A, T, R \rangle$. Stochastic outcomes, dependent on both the current state and the action, model many classes of decision-making problems well. Therefore, finding solutions to MDPs is a major focus of decision-making research for robotics as well as other fields. Given the full specification of the MDP, action policies (i.e., the decision-making algorithm) that are optimal *in probability* can be found for many classes of MDPs, offline, for every possible state

and action combination, using classical methods such as value iteration [30], policy iteration [31], or other related methods derived from these concepts. Solving for an offline policy, ahead of time, and simply querying a fully specified action policy $\pi(s)$, to determine the next best action to take, is related to the sense-plan-act paradigm discussed earlier for deterministic decision-making; however, a new action must be selected at each step since the state evolution is stochastic. So while the decision-maker is reactive to however the state changes, the policy determining the best action to take from any possible state is fixed and already known. Even if solving for offline policies is possible for some MDPs, it is not always practical to do so. This is especially true if the state space is large, since finding the solutions will be computationally prohibitive. Therefore, for many practical MDP applications, online solvers are used. One of the most important online solution methods is Monte Carlo Tree Search (MCTS), which is an informed sampling search method that focuses on branches of possible decision trees that are the most likely to have high reward payoff [32, 33]. Focusing the search on likely high payoff branches, instead of exhaustively searching the entire space can significantly reduce the computational expense needed to find good solutions, especially for problems where the majority of the state space is not useful and only small segments are of interest.

While MDPs cover a large class of decision-making problems, they still make an assumption that is often not valid in many situations. This is that the robot always has accurate knowledge of the states, which is often difficult to assume in real-world environments. This leads to an extension of the MDP, known as the Partially Observable Markov Decision Process (POMDP). As the name implies, the states are now partially observable, meaning they cannot be known with certainty. They can, however, be estimated, with uncertainty, based on *observations*. POMDPs augment the definition of MDPs by adding an observation function $O$ and the observation space $\Omega$ and the decision-maker must maintain a *belief* $b(s)$ over all possible states. The observation function provides a likelihood estimate of the occurrence of an observation event, conditioned on the posterior state and the action chosen. This can then be used to update the state belief, every time a new observation occurs. Therefore, a POMDP is fully specified by its state space $S$, action space $A$, transition function $T$, and reward function $R$, observation space $\Omega$, and observation function $O$ as $\langle S, A, T, R, \Omega, O \rangle$. Under certain conditions, optimal policies for POMDPs can be found [34–36], but generally these problems are computationally intractable and approximate solution methods must be used. These methods include point-based approximate solvers [37–41], and Monte-Carlo

planning [42, 43].

The "classic" structure of MDPs and POMDPs, as described above, covers many classes of problems, however, it is often necessary to extend the definition of certain elements to describe more complex types of problems. Methods for solving "classic" MDPs and POMDPs have been well understood for decades, but solutions to extensions of these frameworks tend to be more challenging and are a very active focus of decision-making research. One important class of problems is time-dependent MDPs. In these problems, time is usually considered as a state and actions have a duration. The duration may be known or unknown, but the outcomes of the state transition model may be time dependent, the reward function may be time dependent, or both. Many methods exist for solving time-dependent MDPs [44–47], as well as for problems with time-dependent uncertainty [48, 49]. However, these methods require accurate knowledge of the time-dependent parameterization of the problem, so that predictions based on time can be made, for all possible times. Having accurate models of the time-dependence may be possible for some problems, but many real-world scenarios are too complex to model accurately in this way. Existing methods assume the time-dependence model is accurate and they do not have the capability to reason outside of this assumption.

## 2.4 Decision-Making under Ambiguity

Another class of problems that differs from the traditional decision-making under uncertainty structure is that of decision-making under ambiguity [50–53]. Whereas "uncertainty" represents a lack of complete information about the values of the states, "ambiguity" represents a lack of knowledge about the distributions of uncertainty or models that describe the decision-making process [1, 2]. Some works exist which incorporate ambiguity into MDPs [54, 55] and POMDPs [56, 57], but these works tend to assume that the structure of the models is accurate and ambiguity only represents a lack of knowledge of the values of the parameters of the models. Whether considering a lack of knowledge about model parameters, or model structure, or both, there is a clear lack of existing research into this field of "decision-making under ambiguity." Ambiguity is in fact a very common situation encountered by robotic decision-makers in the real-world, however, since fully accurate modeling of the environment and all possibilities cannot be guaranteed.

The work described by Ross, et al. [56] connects back to another family of approaches to solv-

ing decision-making problems known as Reinforcement Learning (RL) [58], where the rewards and models defining the problems are initially unknown. Contrary to "supervised learning" approaches, RL does not learn the model from a set of training data with labeled truth. RL also differs from "unsupervised learning" approaches, which are generally intended to learn the unknown structure and parameters of a model as accurately as possible. Instead, most RL implementations attempt to learn the *value function*, describing the long-term rewards, by taking actions and interacting with the environment in real-time. The goal of RL is simply to maximize the long-term expected reward, as actions are taken, as is generally the fundamental objective of any decision-making process. The difference is that an RL-based decision-maker has knowledge that its mapping between states, actions, and rewards are initially unknown and it must take this into account when selecting actions. While not necessary, RL implementations may also attempt to learn the unknown models that describe the environment in which they are acting. *Model-free* RL decision-makers are simply trial-and-error learners that leverage prior experience to choose the highest-value actions, whereas *model-based* RL decision-makers attempt to learn the model, as well as the value function to perform more informed predictive decision-making [58]. RL has a wide range of applications to autonomous robotic decision-making, including learning new skills [59–62], dexterous manipulation [63, 64], navigation [65–69], control [70–73], and more. Also in RL-based decision-making, there is a trade-off between exploration (i.e., gathering new information about the unknown model) and exploitation (i.e., leveraging what information has already been gathered to make decisions with high expected reward) that must be considered for every action that is chosen. Some methods focus on incorporating this "exploration vs. exploitation" trade-off directly into the action policy by weighting actions by balancing their expected reduction in uncertainty (exploration) and their expected reward payoff (exploitation) [74]. The challenge with RL and other machine learning methods for robotic decision-making is that significant quantities of data are required. Additionally, in order for these methods to produce reliable solutions, this data must usually represent a diverse sampling over the space of possible outcomes. Model-based RL attempts to build a model to describe *why* outcomes occur the way they do and then use traditional "top-down" reasoning to plan, based on these learned models. While this can be successful in many cases, it is generally not possible to model every possibility, even through learning. These learned models also tend to "converge" to assuming a particular model is true and significant counterexamples are generally needed to "unlearn" a highly converged upon model. Many sources of uncertainty in the real-world are

non-stationary and such convergence, while beneficial in cases of controlled environments with limited sources of uncertainty, can be detrimental in more complex scenarios where the sources of uncertainty may vary parametrically. Another challenge is that many problems, except for very simple problems, suffer from the curse of dimensionality and solving these problems may be computationally challenging. One strategy for addressing this in the context of RL is the incorporation of deep neural networks to more efficiently handle high-dimensional state spaces [75]. While this "deep reinforcement learning" has proven to be an effective method at solving many problems, even more so than other RL methods in some cases, it often requires a large amount of training data, which is prohibitive in many situations where such quantities of data are not easily available [76]. Overall, RL is a very powerful tool, but it is not well suited for all types of problems. Other ways of approaching these challenges are still needed.

Another approach to handling this concept of ambiguity or "model uncertainty" are multi-model methods, also called ensemble methods. These methods are often used in the context of machine learning "prediction" and "classification" [77–79] and in stochastic estimators when fusing data from multiple sources that may have different uncertainty models [80–84]. Another related concept is the principle of Minimum Description Length (MDL) which can be used to interpret stochastic information and discriminate between competing models to describe the information [85, 86]. Multi-model methods are also used in "decision-making," [87, 88] but many of the underlying principles of these concepts, in both paradigms, are generally very similar. In the context of decision-making these methods, different models of the problem are used, based on the assumption that one model is not sufficient to describe the complexity of the problem. An "arbitration" step is then used to combine their outputs to result in one final output. Weighted voting is one of the most common approaches to arbitrate ensemble methods [89]. Other applications of these methods include information management to support human decision-making [90, 91], predictions about difficult to diagnose medical conditions [92, 93], economics [94–96], and more. Turning the focus back to autonomous robotic decision-making, however, a number of methods exist that incorporate these concepts [97–99], but the majority of them use these methods mainly to improve their perception performance or to learn models of the decision-making problem and then use traditional decision-making methods to carry out actions. Some examples exist of using multiple models directly in the decision-making process [100], but this is an underexplored area of robotic decision-making research.

A similar concept, more closely related to decision-making, however, is adaptive control, which assumes that the parameters that define a system are non-stationary and not initially known. The controller must estimate the parameters during execution and adjust the control law to adapt to the present best estimate of the parameter values [101]. A contrasting method, robust control, handles uncertainty by using a fixed control scheme that is known to maintain the desired performance as long as the uncertainties stay within prescribed bounds [102]. While applicable to many problems, both of these methods hinge upon a set of strong assumptions about the nature of the problem that are often violated in real-world scenarios. Adaptive control assumes that the set of parameters which define the problem are known ahead of time and it is just their values that are unknown. For constrained problems in controlled environments, this may be a valid assumption, but this can be very difficult if not impossible to assume for more complex real-world scenarios. And for robust control, the assumption that the uncertainty is bounded also is difficult if not impossible to guarantee outside of very controlled conditions. These methods also typically apply to "control" problems, as opposed to "decision-making" problems. While related, control problems are generally more focused on controlling a set of physical states whose dynamics can be described by continuous differential equations, whereas decision-making problems can extend to more abstract problems where the description of the dynamics may not be as succinct.

## 2.5 Reducing Dependence on Prior Knowledge

The overall paradigm of most of the topics discussed thus far is that some kind of *model* that describes the dynamics and/or uncertainties of the problem is either provided as prior knowledge, or can be learned through training data or experience. In other words, these methods operate based on the assumption that the reason *why* outcomes happen the way they do can be explained and therefore reasoned over in a logical manner. A course of action is then selected as the one that is believed to best achieve the objective, given the model that explains why that course of action is the best. Taking a step back and thinking about the broader purpose of autonomous robotic decision-making, however, raises the question: is finding a model that explains the dynamics and sources of uncertainty always necessary to make decisions that work towards achieving the objective? Can a decision-maker instead find a way to decide what actions to perform, based on only how well those actions work to achieve the objective, regardless of why they do? One key example

of a decision-making framework following this alternative paradigm is an algorithm known as Intelligent Trial and Error (IT&E) [4]. In this work, a multi-legged walking robot is shown to be able to overcome varying types of damage to its limbs and find new ways to walk that still enable it to continue moving forward successfully, even in a reduced capacity. This is accomplished by sampling a space of diverse walking behaviors, testing how well those behaviors perform, and then scoring those behaviors, and closely related ones, based on their performance. Weighted sampling is performed based on the behaviors' expected performance, so with very few iterations, new behaviors that perform well can be found. The robot does not need to understand why it can no longer walk successfully as it did before and perform top-down reasoning to find a new way to walk that agrees with a model. Instead, it simply performs "whatever works best" to continue its task, without an explicit understanding of why the new behavior works. This is similar in concept to model-free RL but is a different approach, since there is no attempt to learn the full value function.

A similar phenomenon, based on this paradigm of finding what works best without needing to find a model to explain why, can be exhibited by robotic swarms. What is often described as "swarm intelligence" is the ability of a swarm of robots to accomplish tasks that are more complex than any agent in the swarm is designed to be capable of individually [103–105]. Through exchanging information between agents to influence the swarm's collective behavior, swarms are able to demonstrate "emergent intelligence" or "emergent behaviors" that overcome unmodeled dynamics and/or uncertainties without explicitly modeling them [106, 107]. These properties of swarms are highly pertinent to the objectives of this research proposal and motivate many of the proposed methods being explored.

## 2.6 Literature Review Summary

In summary, the key gaps in existing research, with regard to decision-making that is resilient to ambiguity, are methods that can overcome unexpected sources of uncertainty or altered system dynamics without relying on a model that describes the situation being experienced. Therefore, this research focuses on methods of decision-making that can perform better at the task, without a need to directly model the ambiguity being encountered. As a baseline, however, existing decision-making methods, applicable to the case studies upon which the proposed methods will be tested, must first be explored in more detail.

# 3

# Decision-Making Preliminaries

## 3.1 Chapter Overview

The objective of the work described in this chapter is to provide an overview of the baseline methods used in autonomous robotic decision-making. This includes both theoretical presentations of each major framework as well as specific examples, showing implementations of each of the frameworks. Discussion about the challenges and limitations for each of these frameworks, based on each of the examples, is provided, along with a comprehensive discussion at the end of this chapter about the limitations of these frameworks, motivating the need for the research objectives of this proposal, which are presented in the remaining chapters.

This chapter begins with a discussion of finite state machines. Two finite state machine examples are provided, one for a robot foraging task, from the author's work on the robot Cataglyphis, which was built by the West Virginia University (WVU) team that won the NASA Sample Return Robot Centennial Challenge, and another from the author's work on the robot BrambleBee, which autonomously pollinates bramble (i.e., blackberry and raspberry) flowers in a greenhouse. Next, the framework of Markov Decision Processes (MDPs) is discussed, presenting the formal definition of the MDP framework and the MDP solution method used in this research. Afterwards, the Partially Observable Markov Decision Process (POMDP) framework is discussed, presenting the formal definition of the POMDP framework, as an extension of that of MDPs and a brief discussion of solutions methods. And finally, a summary discussion about the limitations of these frameworks is presented to motivate the concepts proposed in the following chapters.

## 3.2 Finite State Machines

A finite state machine (FSM) is a computational model of a discrete decision-making problem. It encodes both a model of the problem, as the system's possible *states*, *inputs*, and *state transitions* (i.e., actions), as well the decision-making logic used to solve the problem as the *mapping* between each state, input and state transition. This mapping may be deterministic, meaning that given a particular current state $s$ and input $u$, the new state $s'$ is always the same, or the outcomes may be stochastic, meaning there exists a probability distribution over possible new states.

FSMs are widely used in robotics and many forms of industrial automation where a fixed automated procedure must be followed. Examples include manufacturing robots in factories, the

on-board software on robotic spacecraft, automatic transmissions in cars, turnstiles at public transit stations, and even drink vending machines. The case of a drink vending machine is used as an illustrative example. The machine starts in the idle state, waiting for a user to insert money. Once sufficient money has been inserted, the machine transitions to the selection state, where the user selects their desired drink. In this state, the user may also press the refund button and the machine will refund their money and return to the idle state. If the user proceeds to purchase a drink, however, the button pressed determines which drink is vended. Once a drink has been vended, the machine returns to the idle state. This FSM is illustrated in Fig. 3.2.1.

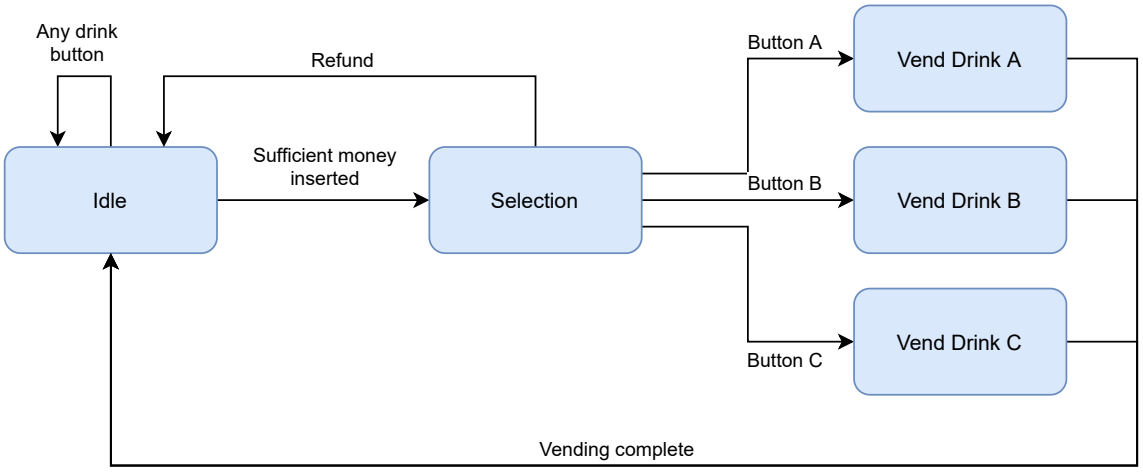



**Figure 3.2.1:** Vending machine FSM example

FSMs are simple to understand and easy to implement for problems such as this, where the number of states and state transitions are small and such a paradigm accurately describes the process being controlled. For more complex problems, specifically defining every possible state transition for every possible state and input combination becomes difficult and needing to increase the number of states to describe increasingly complex problems makes managing very large FSMs difficult. FSMs are still useful for controlling the decision-making of complex tasks, however, especially when they are used in a hierarchical manner, where reasoning is performed over varying levels of abstraction. Lower-level processes are encapsulated in their own FSMs, which themselves may make up the components of higher-level state machines which control more abstract levels of decision-making. Presented next are two examples of FSMs developed by the author for complex robotic

systems, performing high-level mission planning, enabling the robots to operate autonomously.

### 3.2.1 Example: NASA Sample Return Robot Challenge

One example of FSM decision-making is the mission planning algorithms developed for the robot Cataglyphis, built by the WVU team that won the NASA Sample Return Robot Centenntial Challenge. The objective of the challenge was analogous to the upcoming Mars Sample Return rover mission, which is to retrieve "samples" scattered over a large outdoor area and return them to a starting platform. This simulated the robot loading the sample into a rocket on the Mars rover's lander for transport back to Earth. Cataglyphis needed to operate fully autonomously for two hours, traverse a large portion of an approximately 20,000 $m^2$ outdoor area to search for samples, visually recognize and collect several objects (e.g., a small plastic cylinder with a hook, colored rocks, and small blocks of metal), and return these objects to the starting platform. A map of the challenge field and an image of Cataglyphis picking up a sample are shown in Fig. 3.2.2 and Fig. 3.2.3, respectively.

This search and retrieve task (i.e., robot foraging task) was especially challenging due to the many sources of uncertainties inherent to real-world environments that are difficult to model and because the challenge rules prohibited the use of GPS, magnetic compasses, or any other "Earth-based technologies" that would not be applicable on Mars [108, 109]. During the final challenge in the year 2016, Cataglyphis retrieved five samples and autonomously traversed a total distance 2,692 meters during the two hour challenge duration. Cataglyphis' path and the locations where samples were acquired are shown in Fig. 3.2.4.

In the end, Cataglyphis was the only robot to successfully complete the final challenge and the WVU team won $750,000 in prize money (final total of $856,000, including completing previous levels of the challenge in prior years). Despite Cataglyphis's success, several unexpected failure modes were encountered, which helped inspire the research objectives of this proposal. One of the most interesting of these was the failure of Cataglyphis to pick up one of the samples, due to its own shadow covering up the sample and the computer vision algorithms not recognizing the sample, due to lighting, shown in Fig. 3.2.5.

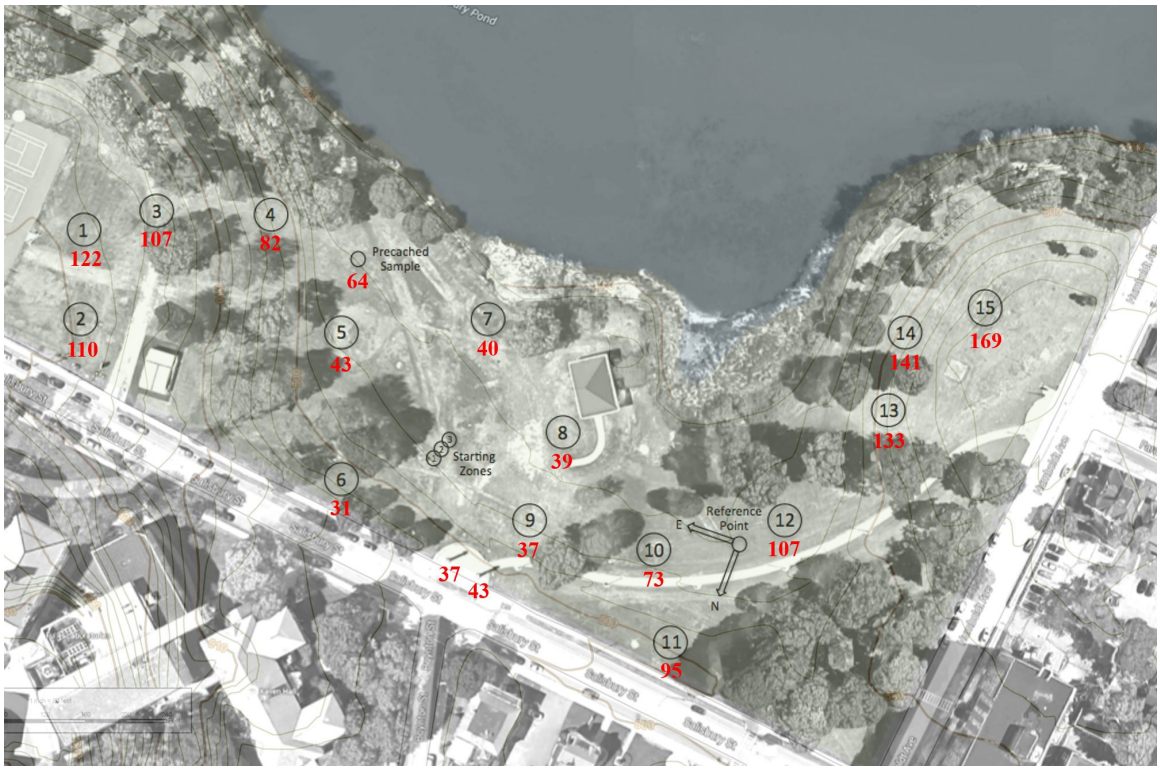



**Figure 3.2.2:** Map of the challenge field for the NASA Sample Return Robot Challenge, located at Institute Park in Worcester, MA. The large black circles indicate regions of interest in which samples may be located and the red numbers indicate this distance from starting zone 2 to the center of each region of interest, in meters.

When the robot was initially approaching the sample, its shadow was not covering it, so it detected the sample reliably. But as it got closer, while approaching the sample to attempt to grab it, its shadow then covered up the sample and the computer vision algorithms no longer detected the sample with high enough confidence to classify it as a possible true sample. The robot would then back up, thinking it lost sight of the sample (perhaps by having driven too far forward). It would then reacquire sight of the sample, once it backed up, and try to approach it again. While repeated failures to approach was a handled failure condition in Cataglyphis' decision-making software (the robot would give up and continue searching elsewhere), significant time was wasted during this process and the challenge ended before Cataglyphis would have a chance to return and pick up this sample, had it just been "lucky enough" to approach it from a different direction where its own


**Figure 3.2.3:** Cataglyphis picking up a sample during the NASA Sample Return Robot Centennial Challenge (photo credit: **NASA/Joel Kowsky**, © **CC BY-NC-ND 2.0**)

shadow would not have interfered. It also may not have failed if the time of day had been different, meaning the angle of the sun above the horizon, and therefore the shape of the robot's shadow, was different. The point is that this specific failure mode was never encountered during the many weeks of testing prior to the challenge, so there were no specific measures programmed into Cataglyphis' decision-making to attempt to mitigate a situation like this. And further, attempting to both identify and handle every possible edge-case, like this, is simply impossible, whether through explicit handling of specific failure modes, or through creating a sufficiently detailed mathematical model of the problem, such that all of the necessary dynamics are captured in the model. Such a list of specifically handled failure modes would be impractically exhaustive to capture and such a fully encompassing mathematical model would be impossibly difficult to derive. Therefore, this highlights the need for autonomous robotic decision-making that is resilient to these kinds of ambiguities

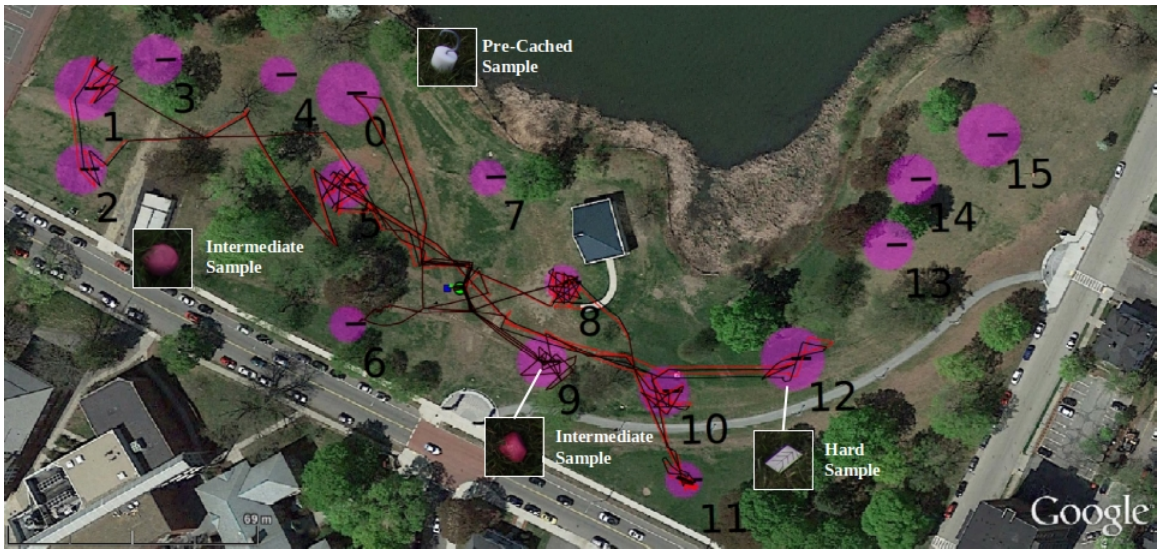



**Figure 3.2.4:** Overlay of the path driven by Cataglyphis during the first 80 minutes of the final challenge in 2016. The black line shows Cataglyphis' perceived path, based on its primary localization system, and the red line shows a solution maintained in parallel from simultaneous localization and mapping (SLAM). The purple circles represent the regions of interest and the locations from where samples were retrieved, up to this point in the challenge run, are shown.

that cannot be explicitly handled. The design of Cataglyphis' autonomy did include some small innovations in this regard, but it was still far from a sufficient case-study into this topic. However, this initial work does motivate some of the proposed methods and the design of the case studies presented in this research. The design of Cataglyphis' autonomous decision-making algorithms are therefore presented next.

Cataglyphis' autonomy was structured hierarchically to separate high-level decision making from low-level subsystems control, as shown in Fig. 3.2.6. The layer at the top level of the hierarchy, known as the Mission Planning State Machine (MPSM), was responsible for deciding the high-level **tasks** that the robot should perform. Cataglyphis had a total of 17 unique tasks, such as choosing new regions of interest (ROIs) to search, searching an ROI, approaching and collecting samples, and returning home to drop off samples. A complete list is provided in Tab. 3.2.1. Execution of these high-level tasks consisted of sequences of individual robot **actions**, such as actions to drive the robot, manipulate samples, and search for samples. These sequences of actions were then passed down to the execution layer to carry out the detailed robot control. Cataglyphis had a total


**Figure 3.2.5:** Cataglyphis attempting to approach a sample (can be slightly seen in the bottom left) during the challenge, but failing to do so, due to its own shadow causing the computer vision algorithms to lose track of the sample, as it got into grab position.

of nine unique actions, such as driving to waypoints in global or local coordinates, performing grab or drop sequences with the grabber, and searching for samples with the vision system. A complete list is provided in Tab. 3.2.2. Each action itself is a small, encapsulated FSM, sometimes requiring several steps to complete. For example, the Drive Global action consists of first pivoting in place to face the goal coordinate and then driving straight while maintaining the heading required to keep the robot aimed at the goal coordinate. Each task is also an encapsulated FSM, operating on actions as their states, though generally more complex than the action FSMs. Some tasks, such as Next Best Region, simply consist of a sequence of Drive Global actions, while others such as Approach have much more complex state transition logic, where the outcomes are dependent on the location and confidence of detected sample candidates from the vision system. And again, the MPSM itself is yet another FSM, but even more abstract. It is at the top of the hierarchy, with tasks as its states. This hierarchical separation made the autonomy software easier to develop and manage, as well as providing the modularity, flexibility, and resilience to ambiguity needed to make Cataglyphis functional when faced with challenging real-world uncertainties and unmodeled situations, all of

which are too complex to be completely and accurately described by the FSM hierarchy.

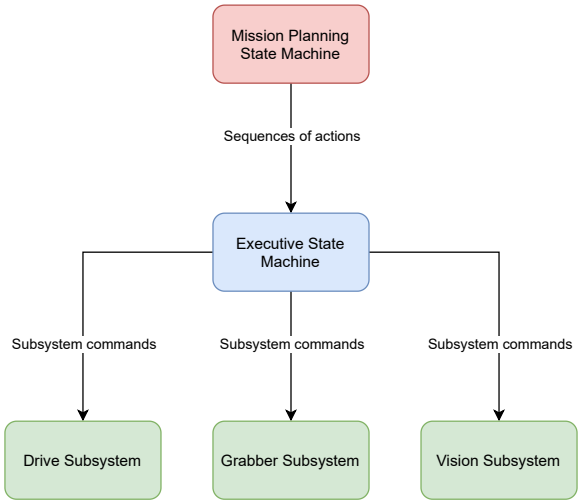



**Figure 3.2.6:** Cataglyphis autonomy architecture

The action and task level FSMs are fairly standard FSMs, but a number of innovations were made in the MPSM to attempt to handle the ambiguities that were not able to be explicitly accounted for in the design of the MPSM. The MPSM's state transition logic is very complex and depends on numerous inputs, which are condition flags representing various conditions about Cataglyphis and its mission, such as whether the robot sees a candidate sample, if the robot is in possession of a sample, if the robot is at the home waypoint, and more. When a task is complete (or when a task is interrupted, such as when obstacle avoidance is triggered), a truth table over the configuration of the condition flags is used to determine the next state transition. This implementation is not only less cumbersome than defining each state transition explicitly, but it also enabled a key feature of the MPSM that increased Cataglyphis' resilience to ambiguity: fallthrough conditions. Many combinations of condition flags were considered to be invalid and should never occur. For example, the robot should not be in drop-off position on the starting platform and also performing a reorient maneuver to approach a sample from a different angle. This condition does not make sense and should never occur during normal operation, but if a bug in the MPSM software was encountered that was never found during testing, or an unexpected condition occured, that was never antici-

| Task | Description |
|---|---|
| Initialize | Back the robot off the starting platform and initialize pose |
| Emergency Escape | Back up and drive to an offset position to avoid severe hazard |
| Avoid | Avoid obstacle in front of robot |
| Bias Removal | Remain stationary and remove IMU bias |
| Next Best Region | Choose next region of interest to search |
| Search Region | Plan waypoints to search region of interest |
| Examine | Drive to a favorable observation position of a candidate detected sample |
| Approach | Maneuver to grabbing position for detected sample |
| Collect | Attempt to pick up sample with grabber |
| Confirm Collect | Take another observation to confirm sample is no longer on the ground |
| Reorient | Move to approach sample from different direction after multiple failed approach attempts |
| Go Home | Return to the home waypoint in front of the starting platform |
| Square Update | Drive a sequence of waypoints around the homing beacon to attempt to remove localization error |
| Deposit Approach | Precisely drive onto starting platform to sample drop off location |
| Deposit Sample | Drop sample on ground |
| Safe Mode | If localization is lost, perform random drive maneuvers in hope of finding homing beacon |
| SOS Mode | Alternative to Safe Mode; stay in place and perform random rotations to signal SOS mode to human operators |

**Table 3.2.1:** List of Cataglyphis high-level tasks

pated when designing the condition flag truth table, an invalid condition such as this could occur. If such a condition did occur, however, the condition flag truth table would default to a fallthrough condition, which would perform a "soft reset" on the condition flags, setting them a "safe" configuration that would enable the robot to return to a "normal" line of decision-making. This may result in very suboptimal behavior in achieving the mission, such as repeating search of a region that has already been searched, but this is preferable to completely failing due to an undefined condition.

These undefined conditions are a type of ambiguity that may be encountered by autonomous robots in the real-world and if their decision-making is not designed to consider that unmodeled situations may occur, then the robots' decision-making will be very fragile and may fail as soon as

| Action | Description |
|---|---|
| Idle | Command drive system to stop |
| Halt | Command drive system to stop and hold position on a slope |
| Drive Global | Command drive system to drive to a global coordinate |
| Drive Relative | Command drive system to drive to a position relative to its current position |
| Grab | Command grabber system to go through grabbing sequence |
| Drop | Command grabber to go through drop-off sequence |
| Open | Command grabber to open without dropping |
| Search | Command vision system to capture an image and search for nearby samples |
| Wait | Keep all subsystems idle for a period of time |

**Table 3.2.2:** List of Cataglyphis actions

an unanticipated situation occurs. Handling this with the fallthrough condition implemented for Cataglyphis may be a very naïve approach, but understanding the root problem it was designed to address was an important inspiration for the research objectives of this proposal.

### 3.2.2 Example: Autonomous Robotic Pollination

Another example of robotic decision-making, based on FSMs, is the mission planning state machine developed by the author for the autonomous precision pollination robot, BrambleBee. BrambleBee is a robot developed by an interdisciplinary team at WVU to autonomously pollinate bramble plants (e.g., blackberries and raspberries) in a greenhouse [110]. As shown in Fig. 3.2.7, it is a mobile manipulation platform, equipped with perception systems for both localizing itself and identifying and mapping the locations of flowers to be pollinated, and a robotic arm with a custom-built end-effector for precisely pollinating individual flowers. BrambleBee's objective is to operate fully autonomously in a greenhouse, during the days which the bramble plants are flowering, search the greenhouse for flowers that require pollination, and then individually pollinate them.
This is a complex problem and there are many sources of uncertainty and aspects of the problem that are difficult to model accurately, such as the limited pollination viability time of flowers, reliably keeping track of flowers that have already been pollinated to avoid wasting time pollinating them


**Figure 3.2.7:** Precision pollination robot BrambleBee pollinating blackberry flowers in the WVU greenhouse

again, and the inherently difficult problem of reliable flower detection and mapping.

Similar to Cataglyphis, as described previously, BrambleBee's autonomy is hierarchical, with lower-level encapsulated FSMs controlling more specific functions, such as the operation of the manipulator, with an overall high-level mission planning FSM controlling the mission level tasks. The mission planner decides waypoints for the robot drive base as well as when to begin the pollination procedures, which are carried out by an encapsulated manipulation FSM. BrambleBee's mission consists of two phases: the survey phase and the pollination phase. During the survey phase, the robot drives around the rows of plants in the greenhouse, attempting to detect unpollinated flower clusters and record their locations in a map of the greenhouse. After the survey phase is complete, the robot moves on to the pollination phase, where it must decide where to park the drive base to enable the pollination of reachable flowers by the manipulation system. The detected flower map is binned into several "plant row sectors" on each side of each plant row, each of which

are the approximate width of the arm's reachable space. Each sector has an associated "parking pose" to which the robot can drive to position itself so that the flowers within that sector are easily reachable. The mission planner must decide the order in which the robot should visit the parking poses of sectors with unpollinated flowers. This is currently accomplished using a simple one-step greedy heuristic, which minimizes a cost function based on the number of reachable, unpollinated flowers in each plant row sector and the distance from the robot's current location to the parking pose for each sector, as shown in (3.1).

$$\underset{i}{\operatorname{argmin}} \frac{C_f}{F_i} + C_d \left\| \mathbf{x_i} - \hat{\mathbf{x}}_\mathbf{r} \right\|_2 \tag{3.1}$$

Where $i$ is the plant row sector index, $F_i$ is the number of flowers in sector $i$, $\mathbf{x_i}$ the position of the parking pose of sector $i$, $\hat{\mathbf{x}}_\mathbf{r}$ is the robot's estimate of its current pose, and $C_f$ and $C_d$ are weighting constants for the number of flowers and the distance, respectively.

Once the robot arrives at a parking pose in front of a plant row sector, the manipulation subsystem is then commanded to begin pollinating flowers. The drive base remains stationary while all of the manipulation actions are performed. Then, once the manipulation subsystem reports that it has completed the pollination procedures, another sector with unpollinated flowers is chosen as shown in (3.1) and this process repeats until all the sectors with unpollinated flowers have been pollinated. The FSM implemented by the mission planner that carries out this sequence of decision-making is illustrated in Fig. 3.2.8.

While the BrambleBee mission planner is just a standard FSM, lacking the innovations added to the Cataglyphis MPSM to attempt to make it more resilient to ambiguity, there are still some important conclusions from this work in regard to the research objectives. One is that a simple design that is purposefully based on lower-fidelity, incomplete models can in fact be more resilient to ambiguity than a more complex design that considers many more states and uses more sophisticated decision-making strategies. The more states describing highly specific details of the problem that are considered by the decision-maker, the more the decision-maker depends on the value of each state and the models of the problem being accurate. And for more states to be useful, generally more sophisticated models are required as well. If these models are inaccurate or insufficient to describe the problem, then this just introduced more possible points of failure into the decision-maker. In

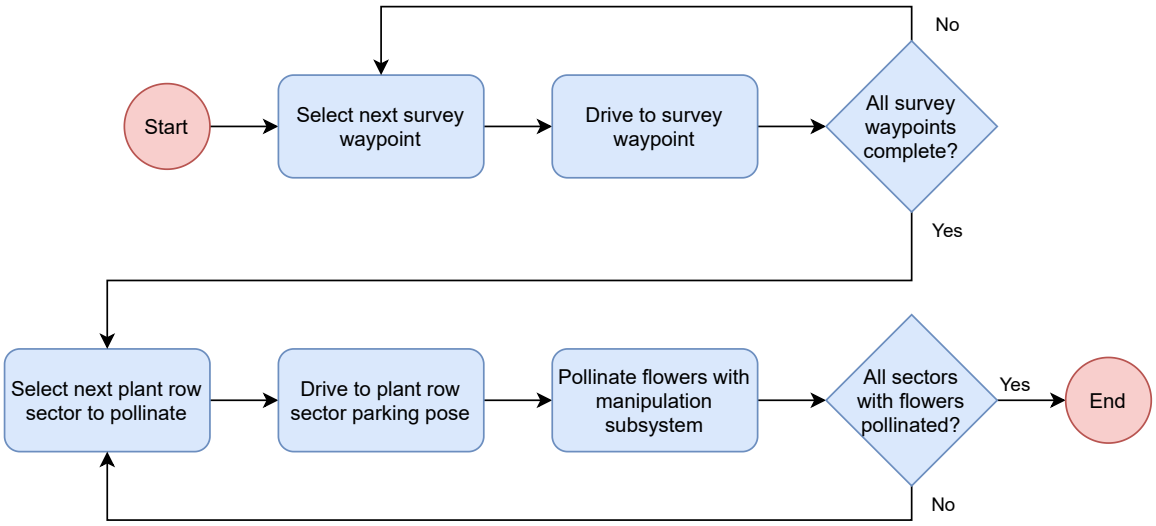



**Figure 3.2.8:** Flowchart describing the BrambleBee mission planner

other words, sometimes less specialized designs are more resilient to ambiguity than very complex and highly specialized ones. Of course, these simplistic methods are likely to suffer in terms of performance, compared to more sophisticated methods, but only when there are no ambiguities and the more sophisticated methods are accurate characterizations of the problem. When unmodeled situations occur, the simplistic methods may be less fragile, simply because there are fewer states and models that may be wrong.

## 3.3 Markov Decision Processes

Building on the concept of FSMs, if a problem can be formulated as a Markov Decision Process (MDP), then an action policy can be found algorithmically, based on the objective of maximizing the total expected reward, computed from the given reward function. This results in a decision-maker that more fully considers the possible long-term implications of chosen actions, given knowledge of the distributions of uncertainty. It does, however, add more complexity to both the definition of the problem and the implementation of the decision-maker. Solving for MDP policies may not always be trivial and significant computational resources may be required. This section will first describe the definition of an MDP and will then discuss MDP policy solution methods.

### 3.3.1 MDP Definition

An MDP is fully specified by its state space $S$, action space $A$, transition function $T$, and reward function $R$, as $\langle S, A, T, R \rangle$. As described previously in Chapter 2, the transition function $T$ represents a probability distribution over possible posterior states $s'$, given the prior state $s$ and action chosen $a$. This is shown formally below.

$$T(s, a, s') = P(s'|s, a) \tag{3.2}$$

In the most general case, the reward function $R$ is defined as a mapping from the posterior state $s'$, the action chosen $a$, and the prior state $s$ to an instantaneous reward value $r$. This is shown formally below.

$$r = R(s, a, s') \tag{3.3}$$

More specific cases may only depend on the action and the prior state $R(s, a)$, or simply only on the prior state $R(s)$, but the above form is the most general.

The objective of any algorithmic MDP solution is to find an action policy $\pi(s)$ that returns an action $a$, based on the current state $s$. Note that in this research, only **discrete** MDP problems are considered, meaning that the values of all states and actions are discrete, rather than continuous. This is an important condition for the methods to be discussed below. In order to find the *optimal* action policy $\pi^*(s)$, the concept of a *value function* must be introduced. The value function $V(s)$ is defined as the expected value of the discounted sum of all future rewards, given current state $s$, equal to the state at current time $s_t$. This is shown formally as follows:

$$V(s) = \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots\right] \ : \ s = s_t \tag{3.4}$$

Where $\gamma$ is a discount factor in the range $[0, 1]$. The discount factor used to weight near-term rewards higher than rewards further in the future. This can be simplified using a recurrence relation, resulting in the widely used form of the value function, known as the Bellman Equation, which is shown as follows:

$$V(s) = \mathbb{E}\left[R_{t+1} + \gamma V(s_{t+1})\right] \ : \ s = s_t \tag{3.5}$$

Since MDPs involve stochastic state transitions as a function of both the state $s$ and the action $a$, the Bellman Equation is extended to state-action pairs as follows:

$$Q(s,a) = \mathbb{E}\left[R_{t+1} + \gamma Q(s_{t+1}, a_{t+1})\right] \; : \; s = s_t, a = a_t \tag{3.6}$$

Where $Q(s,a)$ is the state-action value function. From here, a the expectation in (3.6) can be evaluated as follows

$$Q(s,a) = \sum_{s' \in S} T(s,a,s')\Big(R(s,a,s') + \gamma V(s')\Big) \tag{3.7}$$

Where $T(s,a,s')$ is the transition function, as defined in (3.2), $R(s,a,s')$ is the instantaneous reward, and $V(s')$ is the value of the posterior state $s'$. Maximizing $Q(s,a)$ over $a$ yields the optimal value function $V^*(s)$:

$$V^*(s) = \max_{a \in A} Q(s,a), \forall \, s \in S \tag{3.8}$$

And the optimal action policy $\pi^*(s)$ corresponds to the action $a$ which results in the optimal value:

$$\pi^*(s) = \operatorname*{argmax}_{a \in A} Q(s,a), \forall \, s \in S \tag{3.9}$$

Solving for $\pi^*(s)$ is the challenge of any MDP solution. Presented next are two *exhaustive* search methods, **value iteration** and **policy iteration**, which are iterative solutions, but are known to converge to the *globally optimal* policy, given sufficient iterations. These methods are computationally intractable for problems with large state and action spaces, however, so more *informed* search methods are also important for solving more complex problems. One of the key informed search methods, which will be presented, is Monte Carlo Tree Search. While not guaranteed to converge to the optimal solution, Monte Carlo Tree Search can perform significantly more efficiently than exhaustive search methods, because only state-action pairs that are likely to lead to desirable outcomes are considered. This algorithm estimates online (i.e., during execution) which branches of state-action pairs are likely to have high reward payoff and focuses searching those branches of the decision tree, as opposed to other branches that are not likely to have high payoff, or that are detrimental. Each of these approaches are summarized in the following sections.

### 3.3.2 Value Iteration

The optimal value function (3.8), thereby defining the optimal policy (3.9) can be found by initializing the optimal value over all states to zero and then iterating over every viable state-action-state combination and updating the value for each state using (3.8). The policy is also evaluated at each time step by finding the action that maximizes the value function. At each iteration of the algorithm, the maximum difference between the previous value function and the updated value function is computed. If the difference is smaller than a user selected threshold $\varepsilon$, the algorithm is considered to have converged. The value iteration algorithm is presented below.

**Algorithm 3.1:** MDP Value Iteration Algorithm

**Procedure** ValueIteration$(\gamma, \varepsilon, N, S, A, T, R)$
  $V \leftarrow [0, 0, \ldots, 0] : |V| = |S|$
  $\pi \leftarrow [0, 0, \ldots, 0] : |\pi| = |A|$
  **for** $i \leftarrow 1$ **to** $N$ **do**
    **for** $s \in S$ **do**
      **for** $a \in A$ **do**
        $v \leftarrow \sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma V_{i-1}(s')\right)$
        **if** $v > V_{i-1}(s)$ **then**
          $V_i(s) \leftarrow v$
          $\pi(s) \leftarrow a$
    **if** $\max V_i - V_{i-1} < \varepsilon$ **then**
      **return** $(V, \pi)$
  **return** $(V, \pi)$

### 3.3.3 Policy Iteration

Alternatively to iterating on the value function to find the optimal policy, an arbitrary initial policy can be iterated upon to find the optimal value function. This is known as policy iteration. Similarly to value iteration, the algorithm iterates until the difference of the value function between iterations drops below a threshold. Instead of considering the algorithm to have converged at this

point, however, the policy from the previous iteration is compared to the "optimal policy" evaluated from the current value function. If the policies are the same, the policy is considered stable and the algorithm terminates. Otherwise, the previous policy is updated with the current policy and the algorithm repeats. The policy iteration algorithm is presented below.

**Algorithm 3.2:** MDP Policy Iteration Algorithm

**Procedure** PolicyIteration$(\gamma, \varepsilon, N, S, A, T, R)$
  $V \leftarrow [0, 0, \ldots, 0] : |V| = |S|$
  $\pi \leftarrow$ random $: |\pi| = |A|$
  **for** $i \leftarrow 1$ **to** $N$ **do**
    **for** $s \in S$ **do**
      $V_i(s) \leftarrow \sum_{s' \in S} T(s, pi(s), s') \left(R(s, \pi(s), s') + \gamma V_{i-1}(s')\right)$
    **if** $\max V_i - V_{i-1} < \varepsilon$ **then**
      policy_stable $\leftarrow$ PolicyImprovement$(V, \pi)$
      **if** *policy_stable* **then**
        **return** $(V, \pi)$
  **return** $(V, \pi)$
**Procedure** PolicyImprovement$(V, \pi)$
  policy_stable $\leftarrow$ true
  **for** $s \in S$ **do**
    $b \leftarrow \pi(s)$
    $\pi(s) \leftarrow \text{argmax}_{a \in A} \sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma V_{i-1}(s')\right)$
    **if** $b \neq \pi(s)$ **then**
      policy_stable $\leftarrow$ false
      **return** *policy_stable*
  **return** *policy_stable*

### 3.3.4 Monte Carlo Tree Search

While many other types of solution methods exist and work into solving challenging MDPs is an active area of research, the particular solution method used in this work is a popular framework known as Monte Carlo Tree Search (MCTS). Given (3.7), (3.8), and (3.9), a variant of MCTS, based on

the Upper Confidence Bounds (UCB) applied to Trees (UCT) [33] algorithm is described. The core algorithms of this implementation of MCTS are shown in Alg. 3.3.

**Algorithm 3.3:** Functions used in the Monte Carlo Tree Search Algorithm

```
1  Procedure Expand(τ, v)
2      {a, s'} ~ uniform({A, S} \ v.{A, S})
         τ ← AddChild(τ, s', a)
3      return τ
4  Procedure Select(τ, v, C)
5      if FullyExpanded(v) then
6          a ← argmax_a Q(v.c(a),a)/v.c(a).n + C sqrt(log(v.n)/v.c(a).n)
7          s' ~ P(s'|v.s, a)
8          v ← GetChild(τ, v, a, s')
9      else
10         τ ← Expand(τ, v)
11     return v
12 Procedure Rollout(s, q, M)
13     if AtMaxDepth(M) then
14         return q + q̂
15     a ← RolloutPolicy(s)
16     s' ~ P(s'|s, a)
17     q ← q + R(s, a, s')
18     M ← M + 1
19     return Rollout(s', q, M)
20 Procedure Backpropagate(τ, v, q)
21     v.n ← v.n + 1
22     v.q ← v.q + q
23     τ ← UpdateTree(τ, v)
24     if IsRoot(v) then
25         return τ
26     else
27         v_p ← GetParent(τ, v)
28         return Backpropagate(τ, v_p, q)
```

As shown on line 6 of Alg. 3.3, actions are chosen using the UCB1 algorithm [74], which provides a trade-off between *exploration* and *exploitation*. The overall MCTS algorithm, which makes use of these functions, is shown in Alg. 3.4.

After performing MCTS for as many iterations as desired, as shown in Alg. 3.4, the MDP action policy $\pi$ is then implemented as follows in Alg. 3.5, to find the next best action, based on the tree that has been built up through the MCTS algorithm. After an action is taken and the decision-maker arrives at a new state, the root of the tree must be updated to the vertex that corresponds to the state-action pair that occurred after choosing an action. Branches of the tree other than the one resulting from the state action pair which occurred may now be *pruned*, since they are no longer relevant and maintaining branches that will never be visited is a waste of computational resources.

**Algorithm 3.4:** Monte Carlo Tree Search Overall Algorithm

```
1 Procedure MCTS(τ, N, M, C)
2     q ← 0
3     for i ← 0 to N do
4         v ← GetRoot(τ)
5         while !IsTerminal(v) & !IsLeaf(v) do
6             v ← Select(τ, v, C)
7         q ← Rollout(v.s, v.q, M)
8         τ ← Backpropagate(τ, v, q)
```

**Algorithm 3.5:** Monte Carlo Tree Search Action Policy

```
1 Procedure π(v)
2     a ← argmax_a v.c(a).q
3     return a
```

## 3.4 Partially Observable Markov Decision Processes

An important extension of the Markov Decision Process (MDP) is the Partially Observable Markov Decision Process (POMDP). In MDPs, the states are are fully observable, but POMDPs assume that there is uncertainty about the states and the decision-maker must maintain a probability distribution over all possible states, representing its *belief* in which states may be the true state. The belief is updated through *observations* that provide information about the true state, but still with uncertainty. As with MDPs, POMDPs can be solved to find algorithmically defined action policies that consider all possible outcomes, defined by the states and transitions, and that attempt to maximize the total expected reward, based on a given reward function. POMDP problems are generally much more complex to solve than MDP problems, however, and exact solutions are generally computationally intractable.

### 3.4.1 POMDP Definition

A POMDP is fully specified by its state space $S$, action space $A$, transition function $T$, and reward function $R$, observation space $\Omega$, and observation function $O$ as $\langle S, A, T, R, \Omega, O \rangle$. As with an

MDP, the transition function $T$ represents a probability distribution over possible posterior states $s'$, given the prior state $s$ and action chosen $a$, and is the same as shown in (3.2). The reward function $R$ is also the same as for an MDP and is the same as shown in (3.3). The observation function $O$ represents the likelihood estimate of the occurrence of an observation event $o$, given on the posterior state $s'$ and the action chosen $a$. This is shown formally below.

$$O(s', a, o) = P(o|s', a) \tag{3.10}$$

Since the states are not fully observable, a *belief* $b(s)$ must be maintained over all possible states $s \in S$. The belief represents a probability distribution over the state space. Every time an action is taken, the POMDP decision-maker must update its belief $b'(s')$, based on the previous belief $b(s)$, the action taken $a$, and the observation $o$. This belief update is computed using Bayesian probability, as follows [36]:

$$b'(s') = \eta O(s', a, o) \sum_{s \in S} T(s, a, s') b(s) \tag{3.11}$$

Where $\eta$ is a normalization constant, ensuring the belief sums to one, over all states.

As with MDPs, the objective of any algorithmic POMDP solution is to find an action policy that prescribes an action for any possible state in which the decsion-maker may be. Since the decision-maker does not have full knowledge of its state $s$, it must instead base the policy on the belief $\pi(b)$. Note that in this research, as with MDPs, only **discrete** POMDP problems are considered, meaning that the values of all states, actions, and observations are discrete, rather than continuous. This is an important condition for the methods to be discussed below. As with MDPs, the value function $V(s)$ is defined as the expected value of the discounted sum of all future rewards, given current state $s$, equal to the state at current time $s_t$, as was shown in (3.4) and (3.5) and is similarly extended to state-action pairs as shown in (3.6). Since the states are not fully observable and there is an additional stochastic process governing the observations, however, the expectation for state-action value $Q(s, a)$ for POMDPs is evaluated as follows:

$$Q(s, a) = \sum_{s' \in S} T(s, a, s') \Big( R(s, a, s') + \gamma \sum_{o \in \Omega} O(s, a, o) V(s') \Big) \tag{3.12}$$

Additionally, since the states are not fully observable, the exact value cannot be computed and must

instead be computed based on the belief, as follows:

$$Q(b,a) = \sum_{s \in S} b(s) \sum_{s' \in S} T(s,a,s') \Big( R(s,a,s') + \gamma \sum_{o \in \Omega} O(s,a,o) V(b') \Big) \tag{3.13}$$

Where $b'$ is computed as shown in (3.11). And similar to with MDPs, maximizing $Q(b,a)$ over $a$ yields the optimal value function $V^*(b)$:

$$V^*(b) = \max_{a \in A} Q(b,a), \forall\, b \in B \tag{3.14}$$

Where $B$ is the belief simplex of beliefs over all states $s \in S$. The optimal action policy, based on the belief, $\pi^*(b)$, then corresponds to the action $a$ which results in the optimal value:

$$\pi^*(b) = \operatorname*{argmax}_{a \in A} Q(b,a), \forall\, b \in B \tag{3.15}$$

Solving for $\pi^*(b)$ is the challenge of any POMDP solution. This is even more challenging than for MDPs, because the belief $b(s)$ is a *continuous* probability distribution over all states. Even when the state space is discrete, as is considered in this research, this is still computationally prohibitive. Exact solutions only exist for a special case of discrete problems where the state, action, and observation spaces are not only discrete, but finite. These methods still suffer greatly from the curse of dimensionality and only very small problems are feasible to be solved using exact methods. These exact methods, *value iteration* and *policy iteration* operate similarly to their MDP counterparts in that they are known to converge to the *globally optimal* solution, given sufficient iterations. Since only a very limited set of problems can be feasibly solved using these methods, however, there is significant interest in approximate methods which are more computationally efficient. A brief overview of value iteration, policy iteration, and a select group of widely used approximate methods will be given in the following sections.

### 3.4.2 Value Iteration

For discrete state POMDPs, the POMDP can be transformed into a belief state MDP, where the states of the MDP are the continuous belief states of the POMDP. The belief state MDP is defined

as the tuple $\langle B, A, T^b, R^b \rangle$ where the belief transition function $T^b$ is defined as follows.

$$T^b(b, a, b') = \sum_{o \in \Omega} P(b'|a, b, o) \sum_{s' \in S} O(s', a, o) \sum_{s \in S} T(s, a, s') b(s) \tag{3.16}$$

Where

$$P(b'|a, b, o) = \begin{cases} 1 & \text{if } b^a_o = b' \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

Where $b^a_o$ is the posterior belief from the belief update function in (3.11). The belief reward function $R^b$ is defined as follows.

$$R^b(b, a) = \sum_{s' \in S} \sum_{s \in S} b(s) R(s, a, s') \tag{3.18}$$

The optimal value $V^*(b)$ and optimal policy $\pi^*(b)$ can then be found using value iteration as normally used for MDPs, where the policy is represented as a set of piecewise linear $\alpha$-vectors [34, 36].

### 3.4.3 Policy Iteration

Policy iteration for POMDPs operates similarly to the MDP policy iteration algorithm, but with the value function represented as piecewise linear $\alpha$-vectors. The value function is iteratively updated and the policy resulting from that value function is compared to the current policy. If the policy does not improve, the algorithm terminates. Otherwise, the policy is updated and the algorithm continues [111].

### 3.4.4 Approximate Methods

Solving POMDPs using the exact methods described above is often computationally intractable due to the curse of dimensionality. Many approximate methods for solving POMDPs exist, which include point-based approximate solvers [37–40], and Monte-Carlo planning [42, 43]. The details of these methods are beyond the scope of this research, but the concept behind many of these methods is to use sampling to only solve for policies in relevant subsets of the belief space. Many

regions of the belief space will likely never be reached, due to the structure of the transition and observation models, and therefore computational resources are wasted searching and evaluating policies over them. These approximate methods do not provide as strong of guarantees at finding optimal polices as the exact methods, but they are often the only choice for problems with any but the most trivially small state and action spaces.

# 4

# Methodology

## 4.1 Chapter Overview

As discussed in Chapter 1, this research focuses on three factors in improving the resilience to ambiguity of autonomous robotic decision-making: **diversity**, **physical feedback**, and **local interactions**. This chapter begins with a discussion of these three factors and how they are hypothesized to be key to achieving the objectives of this research and how they will be leveraged in the design of the proposed methods. Following this discussion is a summary of the expected outcomes of the proposed methods. Then, a series of descriptions of problem formulations, case studies implementing these problems in simulation, and proposed methods, along with research hypotheses to be tested through the case studies for each proposed method will be presented. The conclusions from the results of these case studies are discussed incrementally and then summarized at the end before moving on to the overall concluding remarks and discussions on future work in the next chapter.

## 4.2 Key Factors and Expected Outcomes

**Physical feedback** is information the decision-maker gets from the environment in response to taking actions. The utility of such information is dependent, however, on the decision-maker's ability to interpret the information. In many cases, this is encoded into the decision-maker's *model* of the problem. It is common in real-world scenarios that the decision-maker's model may be incomplete and such information gained through physical feedback may not be very informative in and of itself. It may be possible for this information to be indirectly informative of how to solve the decision-making problem more effectively, however, without the decision-maker explicitly understanding the underlying reason. The decision-maker may not need to understand the *cause* of the better utility of one course of action versus another, but if it can simply understand the *correlation* between information gained through physical feedback and the potential for success of carrying out a particular course of action, then it is possible for the decision-maker to reason outside of the limitations of its models, at least in a very limited way. Building an effective understanding of these correlations requires large amounts of information gained through physical feedback, from a variety of conditions in the environment, however. This is where the concept of **local interactions** is leveraged. In scenarios where there are distributed robotic swarms, different agents will experience

different parts of the environment and gain different information through their own physical feedback. If different agents are able to exchange information with each other, then they may be able to more effectively find correlations between the information from physical feedback and the potential for successful actions, since each agent will gain different experience. This ability to successfully understand these correlations may be further improved through swarms with **diversity**. If different agents in the swarm make decisions differently, they are likely to experience different outcomes and gain different physical feedback, which may lead to a better understanding of the action-success correlations, in conjunction with information exchanged through local interactions, than would be achieved otherwise.

Again, the objective is not to develop a top-down understanding of the decision-making problem and attempt to create a more complete model that reduces the ambiguity and accurately describes the distributions of uncertainty and any missing information from the initial model. Rather, the idea is to enable the decision-makers to overcome ambiguity without needing to understand *why* an outcome is not occurring as expected, but simply identify *how* a better course of action may be taken. It is also not assumed that the ambiguity decision-makers encounter will be fixed and invariant with time. Therefore, decision-makers should not "converge" on behaviors to compensate for the ambiguity that cannot be quickly "unlearned" when the characteristics of the ambiguity change. This is what differentiates the underlying concept of the proposed methods in this research from other existing techniques, such as reinforcement learning.

To summarize the important points from the discussion above and tie these together with the objectives presented in Chapter 1, the expected outcomes of the methods proposed in this research are:

- Robotic decision-making frameworks that operate with incomplete/inaccurate models of the distributions of uncertainty involved in the decision-making problem (i.e., ambiguity) that suffer less degradation in performance at the objective of the problem than existing baseline methods, when faced with ambiguities.

- Robotic decision-makers that exhibit *emergent behaviors* which solve the problem in ways that were not explicitly programmed into the decision-making framework and without a full understanding of the problem, in terms of a state/model structure enabling the problem to be directly solved in such a way.

## 4.3 Research Hypothesis: Sensitivity to Ambiguity

Most decision-making under uncertainty frameworks assume that the decision-maker has accurate knowledge about the distributions of uncertainty involved in the decision-making problem. In other words, there is no ambiguity. If such methods are applied to problems where there is ambiguity and the distributions of uncertainty may not match those assumed to be correct by the decision-maker, then it is likely that the decision-maker's performance will suffer. While this is not a surprising expectation, it is important to establish this as a working hypothesis to both ground and motivate the proposed methods incorporating ambiguity to follow. Therefore, the author poses the following hypothesis about sensitivity to ambiguity:

> Decision-making strategies that do not account for ambiguity will generally suffer a reduction in performance at their objective when faced with ambiguities.

In order to investigate this hypothesis, a simulated case study problem is first described and then state-of-the-art decision-making under uncertainty strategies that do not account for ambiguity are tested with ambiguities injected into the problem.

## 4.4 Problem Description: Simulated Robot Foraging

Inspired by the author's initial work on autonomous robot foraging with the robot Cataglyphis, as described previously, robot foraging serves as a key case study problem for this research. Therefore, a simplified robot foraging scenario has been developed, in simulation, to rapidly test different decision-making frameworks. Cataglyphis faced many sources of difficulty to model real-world uncertainty, and this is one of the main motivations for this research. For example, Cataglyphis' ability to pick up samples on the ground often depended on the direction from which it approached them. This was due to a variety of factors, including the slope of the terrain and local terrain features, as well as lighting and shadows, which affected perception. While these factors demonstrate the kind of difficult-to-model real-world complexities upon which the outcomes of this research are focused, this level of complexity is not necessary to demonstrate the expected outcomes of the proposed decision-making methodologies. In addition to the benefit of more rapid testing of different

decision-making frameworks, as mentioned above, the simulation allows for specific types of ambiguities to be introduced into the simulation and trials performed both with and without them, to establish baselines for points of comparison. The desired outcome of this simulated robot foraging problem is test the resilience to ambiguity of the proposed decision-making frameworks, compared to baseline frameworks, based on existing methods. The frameworks will be tested against several variations of ambiguity in the form of unmodeled source of uncertainty that is not accounted for in the decision-makers's model. An MDP problem formulation for the simulated robot foraging problem is presented next.

The foraging robot exists in a two-dimensional discrete grid world, where it must find food, pick the food up, return to its home base, drop off the food, and continue this process until the time limit expires. At each time step, the robot can choose to perform an action to move to any of its eight neighboring grid cells, attempt to grab food located in the robot's grid cell, or drop food, if it is carrying any. It is assumed that the robot is holonomic and can move in any of these directions without needing to pivot. The robot also has a limited amount of energy stored in its battery and each action it performs expends energy. It may recharge its battery by returning to its home location. The robot knows the map of all food locations as well as its own location at all times. The map used in the case studies implementing this problem is 5 by 5 grid cells in size, with the robot's starting position and home location located one grid cell to the east of the south-west corner (i.e., position [x,y] = [1,0]). There are two clusters of food located in the north-west and north-east corners of the map. Each cluster consists of 3 food, for a total of 6 food in the map. This configuration is illustrated in Fig. 4.4.1.

Formally, the robot's state space $S$ and action space $A$ are defined as follows:
The robot's transition function $T(s, a, s')$ defines the probability of the robot transitioning from one state $s$ to another $s'$, given the chosen action $a$. The robot's motion in the grid world is considered to be deterministic, so for example, when the robot chooses to move North-East, the next state will always result in the robot moving to the grid cell location $[x+1, y+1]$ from its current location (as long as the new position is not beyond the grid world boundary). Move actions also incrementally deplete the robot's battery.

The stochastic element, which makes this problem an MDP, is the robot's ability to grab food.

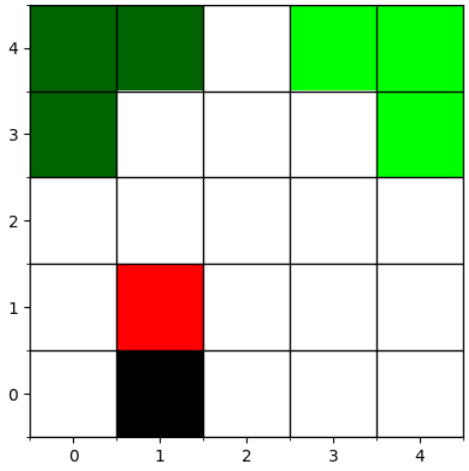



**Figure 4.4.1:** Grid world configuration used for the single robot foraging case study. The red grid represents the robot location, the black grid the home location, and the green grids are food. Different shades of green food represent different food headings.

When the grab action is performed, there exists a probability of the robot successfully picking up food $P_{\text{grab}}$, if there is food at the current location. As far as the robot knows, $P_{\text{grab}}$ is fixed and does not depend on any other states or parameters. In the back end of the simulation (i.e., the true environment) the robot's $P_{\text{grab}}$ is actually a function of an additional *heading* state of both the robot and each food in the map. If the robot's heading aligns with the food heading, then $P_{\text{grab}}$ is high and the robot is likely to successfully pick up the food. The more the robot's heading and the food's heading differs, however, the lower $P_{\text{grab}}$ becomes. This is the source of ambiguity introduced into this problem, which is inspired by witnessing a similar situation occur with the robot Cataglyphis during real-world testing. An additional state and aspect of the state transition dynamics that the robot does not model affects the outcomes of its actions. Along with the move actions, the drop action is also considered to be deterministic, however. If the robot drops the food, it will always be dropped, but it is only allowed to drop food at its home location. Finally, based on these transition

| States |
|---|
| X position |
| Y position |
| Has food? (true/false) |
| Battery charge |
| Food map |

| Actions |
|---|
| Stay |
| Move East |
| Move North-East |
| Move North |
| Move North-West |
| Move West |
| Move South-West |
| Move South |
| Move South-East |
| Grab food |
| Drop food |

**Table 4.4.1:** Definition of robot foraging states and actions.

dynamics, the reward function is defined as follows:

$$R(s,a,s') = \begin{cases} -1,000 & : s'.\text{bat} = 0 \\ -1 & : a \in A_{\text{move}} \\ 0 & : a = \text{stay} \\ -20 & : a = \text{grab} \;\&\; !s'.\text{has\_food} \\ 20 & : a = \text{grab} \;\&\; s'.\text{has\_food} \\ 100 & : a = \text{drop} \;\&\; s.\text{has\_food} \;\&\; !s'.\text{has\_food} \;\&\; \text{at home} \end{cases} \tag{4.1}$$

With this reward function, the robot gets a large reward if it drops food off at home, a large penalty if the battery runs out, a small incremental penalty for each move action, a small penalty for failing to grab food and a small reward for successfully grabbing food.

## 4.5 Case Study: Simulated Single Robot Foraging

The robot foraging problem described above in Section 4.4 is solved using two methods. One through an FSM solution and the second as an optimal MDP policy, using value iteration as pre-

| Model | NE Heading | NW Heading | NE true $P_{\text{grab}}$ | NW true $P_{\text{grab}}$ |
|---|---|---|---|---|
| $T_0^t$ | east | east | 0.9 | 0.9 |
| $T_1^t$ | east | north | 0.9 | 0.1 |
| $T_2^t$ | north | west | 0.9 | 0.0 |

**Table 4.5.1:** True Robot Foraging Transition Models, given a robot heading of East

sented in Section 3.3.2. Three different ambiguity scenarios (i.e., true models) of the simulation world, with different food headings for the two food clusters, which are unknown to the robot, are tested to investigate how resilient each decision-making solution is to ambiguity in the food grab probability. Both the FSM and the MDP solutions assume that $P_{\text{grab}}$ is 90% for all food in the map. The first true model matches the robot's model of the grab probability, so there is no ambiguity in this case. The second true model presents a slight difference from the robot's model, where one of the food clusters has a slightly different food heading, resulting in a reduced grab probability for that cluster. And finally, the third true model presents a much stronger difference, where the heading of the food in one of the clusters is opposite that of the robot and is impossible for the robot to pick up. These true model scenarios are summarized in the table below.

While both the FSM and MDP solutions have incorrect models in the second and third cases, the FSM solution contains a rudimentary heuristic for handling ambiguity, while the MDP solution does not and cannot, without formulating an entirely different MDP problem. The performance of the FSM and MDP solutions, for all three true model cases are compared, based on 1000 simulated Monte Carlo trials of each method, simulated for 100 time steps each. The performance of each decision-making strategy, for each true model, is compared by examining the distribution of the accumulated reward $U$, defined below, summed over all time steps, for each Monte Carlo trial and each scenario.

$$U = \sum_{t=0}^{t_{\max}-1} r(s_t, a_t, s_t + 1) \tag{4.2}$$

The accumulated reward $U$ is compared across all of these scenarios in two ways. First, it is compared through the empirical cumulative distribution function (eCDF) $\widehat{F}_n(U)$, which represents the fraction of Monte Carlo trials that perform less than or equal to a certain value of accumulated re-

ward. The eCDF representation essentially shows the distribution of the accumulated reward over all Monte Carlo trials, sorted in ascending order. And second, the results are compared through box and whisker plots of $U$ to examine the upper and lower extremes, compared to the median performance. These are two representations of the same distributions of outcomes and simply provide different visualizations. Conclusions about the resilience to ambiguity are drawn from these data. An explanation on how to interpret these representations of the results, in terms of resilience to ambiguity, is presented in Appendix B.

### 4.5.1 FSM Solution

The FSM solution consists of four decision-making states, with the state transition diagram as shown in Fig. 4.5.1.

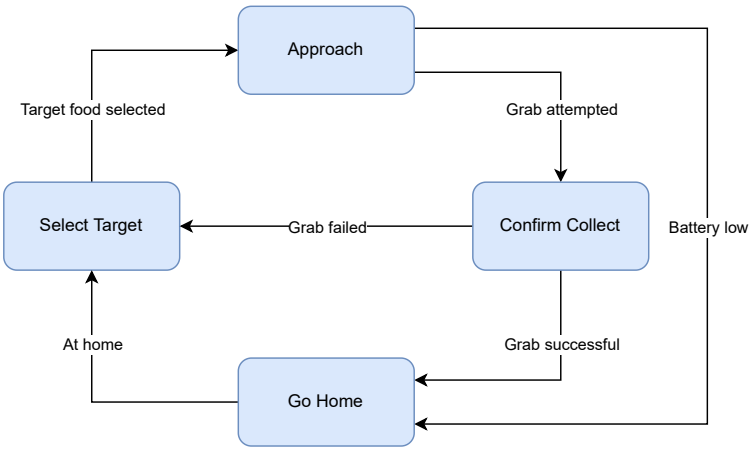



**Figure 4.5.1:** State transition diagram for the FSM action policy implemented as the baseline solution to the foraging MDP

The *Select Target* state chooses which particular food the robot will approach and attempt to grab next. The target food is selected randomly with the probability of a particular food being inversely proportional to the food's distance from the robot's current location. The target food selection process also has knowledge of the cluster to which each food belongs. Once the target food is selected, however, the FSM then transitions to the *Approach* state, which commands the robot to move to the location of the target food. Once the robot arrives at the food location, it then attempts

to grab the food, which transitions the FSM to the *Confim Collect* state. If the robot successfully picked up the food, the FSM then transitions to the *Go Home* state, to drop off the food at home and get a battery recharge in the process. If the robot fails to grab food, however, it transitions back to the *Select Target* state to select a new food to approach and attempt pick up. During *Approach*, if the robot's battery drops below a minimum threshold, before arriving at the food location, however, the FSM also transitions to the *Go Home* state, to recharge its battery by reaching the home location.

### 4.5.2 MDP Solution

The MDP solution is solved simply by taking the problem formulation as described in Section 4.4 and solving for a policy using value iteration, as described in Section 3.3.2. For the first true model scenario, the MDP solution is expected to perform better than the FSM solution, but only because the model happens to be correct. Unlike the FSM solution's rudimentary handling of food grab ambiguity by selecting target food with a weighted random heuristic, however, MDP solution is likely to be fragile when faced with ambiguity presented in the second and third true model cases and it is expected to keep performing actions that are not effective, simply because it does not understand how to reason outside of its fixed model. This method lacks any resilience to ambiguity based on the *physical feedback* of repeatedly failing to pick up the same food, should such a situation occur.

### 4.5.3 Results and Conclusions

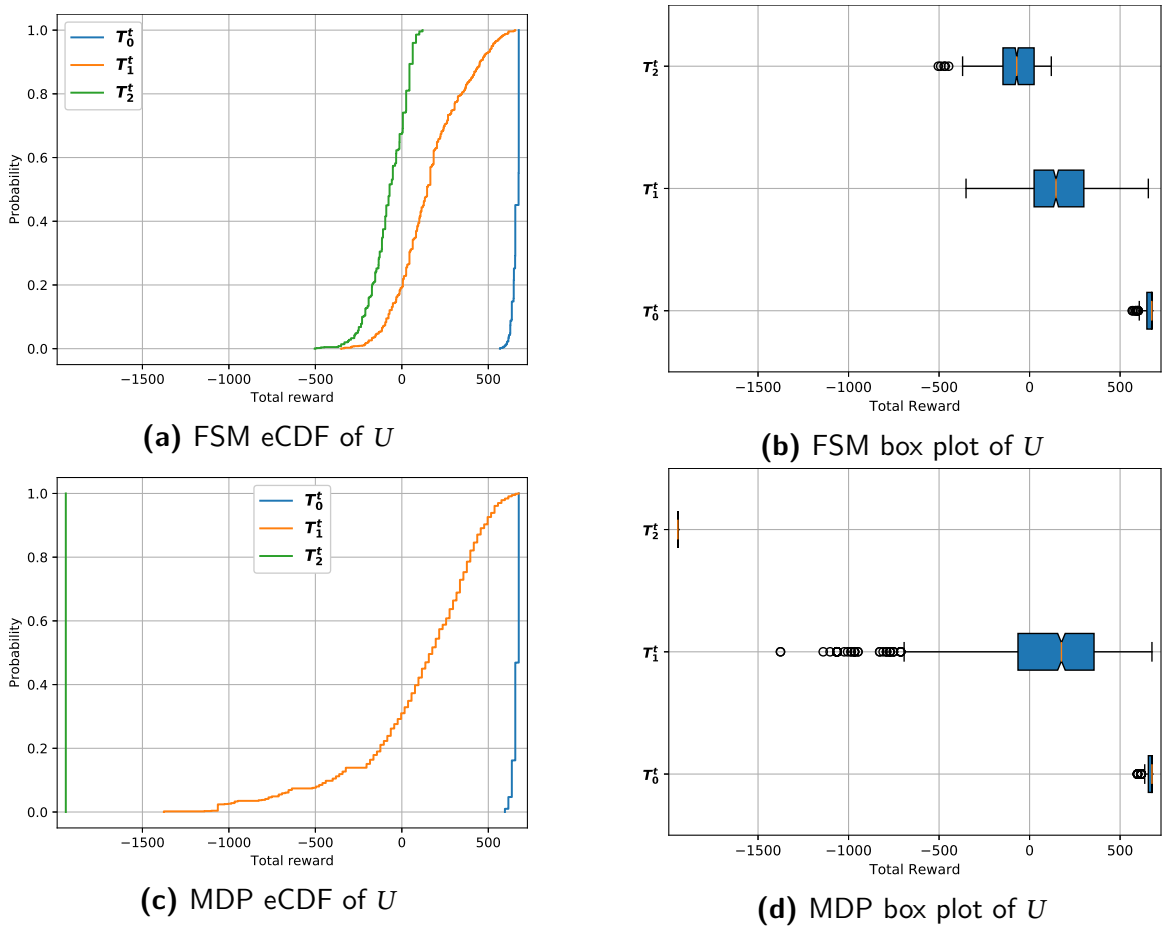
Results from the simulations performed for both the FSM and the MDP solution are presented in the Fig. 4.5.2. The percent change in the major statistics of the Monte Carlo trials, from the FSM solution as baseline to the MDP solution, for each true model scenario, are also presented in Tab. 4.5.2. It can be seen that performance suffers for both the FSM and the MDP the more the true model differs from the model the decision-makers are expecting. The MDP solution slightly outperforms the FSM solution when the model is correct ($T_0^t$), but the MDP solution is more sensitive to ambiguity than the FSM solution and its simple heuristic method of selecting different target food after failed grab attempts. In the case of true model $T_2^t$, where the north-west food cluster cannot be grabbed, the robot based on the MDP solution goes to one of the food in this cluster first, because it is the closest distance to its starting location and just repeatedly keeps trying

to pick up this one food that it cannot pick up, until the end of the simulation. In every Monte Carlo trial of the MDP solution with true model $T_2^t$, the robot goes to the closest food in the north-west cluster and keeps trying to pick it up for the entire duration of the simulation, never trying any other actions. This is why $U$ for every trial of this scenario results in the same large negative value. All that the MDP decision-maker is doing is repeatedly incurring the failed grab penalty, at every time step. The heuristic in the FSM solution may be less optimal when the model is correct, but it provides an element of *diversity*, due to its randomness, that prevents the robot from getting completely stuck in a "mission failure" situation, like the MDP solution does with true model $T_2^t$. The FSM's performance is degraded, but it can still achieve "partial completion" of the mission due to its better resilience to ambiguity. Specific handling of these types of failures could be encoded into the MDP formulation, but these would likely increase the size of the state space dramatically, resulting in the problem quickly becoming computationally intractable for such methods. And again, specific handling of failure modes in such a way is antithetical to the objectives of this research, because it is not feasible to predict all of them ahead of time.

While this demonstration of a simple heuristic FSM solution having more resilience to ambiguity than an "optimal" MDP solution is a very rudimentary example, it demonstrates that *diversity* in decision-making may increase its resilience to ambiguity. Based on this outcome, more sophisticated methods for including diversity in decision-making are desired. This leads to the research hypothesis and the family of proposed methods, presented in the following sections.

| | $T_0^t$ | $T_1^t$ | $T_2^t$ |
|---|---|---|---|
| Max | 0% | 3.05% | -1719% |
| Upper Quartile | 0% | 18.77% | -7872% |
| Mean | 0.14% | -42.33% | -2531% |
| Median | 0.15% | 20.55% | -2637% |
| Lower Quartile | 1.23% | -356.0% | -1222% |
| Min | 5.11% | -292.0% | -285.5% |
| Std Dev | -7.82% | 90.02% | -100.0% |

**Table 4.5.2:** Percent change between the MDP and FSM (baseline) of major statistics for three different true models: $[T_0^t, T_1^t, T_2^t]$.




**(a)** FSM eCDF of $U$

**(b)** FSM box plot of $U$

**(c)** MDP eCDF of $U$

**(d)** MDP box plot of $U$

**Figure 4.5.2:** FSM and MDP foraging solution results for three different true models: $[T_0^t, T_1^t, T_2^t]$.

## 4.6 Multi-Model Decision-Making

As demonstrated thus far, the traditional MDP framework is sensitive to ambiguity. While the examples presented thus far may seem trivial to address by simply reformulating the problem to account for these unmodeled states and state transition dynamics, this is not the point. These examples are simple on purpose so that controlled experiments in simulation can be performed and the point is that this type of situation is likely to occur in a more complex, real-world situation, where there are additional factors that create unmodeled sources of uncertainty about which the decision-

maker is unaware. The objective, as stated previously, is to investigate decision-making strategies which are resilient to these types of ambiguities. One way to extend the MDP and provide some rudimentary handling of ambiguity is to provide it with a set of multiple possible transition models, instead of a single model.

Though a brief aside from the primary focus of this research, this initial work done on multi-model decision-making was key to exploring some of the concepts of this research. A family of extensions to the classic MDP and POMDP frameworks are presented in this section, which include decision-making based on multiple models. The first in this family of methods is known as the Multi-Model Markov Decision Process (MM-MDP) and the second method is an extension of this same strategy to partially observable cases, known as the Multi-Model Partially Observable Markov Decision Process (MM-POMDP). These methods are related to a similar, but markedly different method, also named Multi-model Markov Decision Process (MMDP), proposed by Steimle, et. al [100]. While this method combines rewards from different models into a single weighted value function and then solves for a single policy, the methods proposed in this research solve for a policy for each model separately and then combine their proposed actions through an arbitration scheme online, during execution.

### 4.6.1 Multi-Model Markov Decision Process (MM-MDP)

A traditional Markov Decision Process (MDP) is defined by the tuple $\langle S, A, R, T \rangle$, where $S$ is the state space, $A$ is the action space, $R$ is the reward function, and $T$ is the state transition function. The definition of MDPs is discussed previously in Section 3.3. In contrast an MDP, the proposed Multi-Model Markov Decision Process (MM-MDP) framework is defined by the tuple $\langle S, A, R, \mathbb{T} \rangle$ where instead of a single state transition model $T(s', a, s)$ a set of multiple state transition models $\mathbb{T} = \{T_1(s', a, s), \ldots, T_N(s', a, s)\}$ is utilized. All of these models follow the same structure, but they differ in terms of parameters. Each model $T \in \mathbb{T}$, along with the other MDP elements, can each be considered as separate, individual MDPs and separate, individual action policies can be solved for every model (i.e., $\forall\, T \in \mathbb{T}, \exists\, \pi_T(s)$).

Each individual MDP policy $\pi_T(s)$, based on a particular uncertainty model $T$ is referred to herein as a *decision-maker* $D(T)$. The set of all decision-makers $D(T) \in \mathbb{D}(\mathbb{T})$ is the realization of an MM-MDP. The remaining MM-MDP elements $\langle S, A, R \rangle$ are assumed to be common to all

decision-makers, so only the uncertainty model $T$ needs to be specified to distinguish one decision-maker from another.

The challenge now is to develop a way to arbitrate, or combine the outputs of each action policy from each decision-maker. Each policy $\pi_T(s)$ will output an action to perform, based on its particular model $T$ and the current state $s$, which is common to all MM-MDP policies. In the end, however, the system can only perform one action at each time step, so the outputs must be combined in an informed way that leverages the potential benefits that multiple decision-makers may provide. A naïve method of combining the policies is to use "majority voting." Each policy's output $a_T$ is considered as a "vote" for that particular action and the action that receives the most votes $a^*$, over the space of possible actions $A$, is the one that will be taken at that time step. This can be formally stated by the following.

**Algorithm 4.1:** MM-MDP Action Majority Voting

```
1 Function MajorityVote(𝕋, Π, s, A):
2     z ← [0, 0, ..., 0] : |z| = |A|
3     for i ← 1 to |𝕋| do
4         a ← π_i(s)
5         z[a] ← z[a] + 1
6     return argmax_a z[a]
```

Where $z$ is the vote tally for each action $a \in A$. One of the main limitations of this approach, however, is that every decision-maker gets an equal vote. Some decision-makers' models may not match the true, underlying model very well and the actions they suggest may result in detrimental outcomes. Therefore, there is a need to estimate the "quality" of each model, in terms of how well it matches the true, underlying model, and use this information to select actions that are likely to achieve the best outcomes.

The proposed method for estimating a model quality metric is to use the concept of "uncertainty of the uncertainty model" (UoU). Inspired by the fundamental concept of maintaining a belief distribution over all possible states, $b(s)$, as is done in POMDPs, UoU is quantified by maintaining a belief distribution over all possible models $b(T)$, also referred to as "belief of the uncertainty model" (BUM). Similar to the standard POMDP state belief update shown in (3.11), the model

belief update is performed as a Bayesian belief updated. The concept is to update the belief of each model $b(T) \forall \mathbb{T}$ with the probability of the particular state-action-state tuple $T(s, a, s') = P(s'|a, s)$ as a way to describe how well the state-action-state tuple fits each model. This is done as follows.

$$b'(T) = \eta T(s', a, s) b(T), \forall\, T \in \mathbb{T} \tag{4.3}$$

Where $\eta$ is a normalization constant to ensure the updated belief $b'(T)$ sums to one.

Using BUM as a metric of model quality, the majority voting concept introduced above can be modified to include weights, based on the BUM distribution. Since $b(T) \in [0, 1]$, the BUM values can be used directly as weights. This "weighted majority voting" scheme can be formally stated by the following.

**Algorithm 4.2:** MM-MDP Action Weighted Majority Voting

```
1 Function WeightedMajorityVote(𝕋, Π, s, A, b(T)):
2     z ← [0, 0, ..., 0] : |z| = |A|
3     for i ← 1 to |𝕋| do
4         a ← π_i(s)
5         z[a] ← z[a] + b(T_i)
6     return argmax_a z[a]
```

Where $z$ is the weighted vote tally for each action $a \in A$. The advantage of this approach is that it gives decision-makers whose models are estimated to better match the unknown true underlying model stronger influence over the action selection, lessening the chance that detrimental actions, selected by decision-makers with models that do not match the current situation well, may be taken. While this kind of weighting can be beneficial when one or more of the models closely match the true underlying model, it comes with the disadvantage that, over time, due to being a Bayesian belief update, the BUM will tend to converge entirely towards a single model (unimodal BUM distribution), or subset of the models (multimodal BUM distribution), that best match the true, underlying model, if that model is constant. The effect being that all decision-makers, except the one or small subset converged upon, will be essentially "locked out" of having any influence over the decision-making, meaning many of the advantages of the MM-MDP are lost, and the decision-

maker will behave as a single MDP or only a small set of MDPs, losing the benefit of *diversity*. If the converged upon decision-maker (or decision-makers) have models that match the true, underlying model very closely, then the effect of locking out all other decision-makers is fairly benign, since a single MDP that closely matches the true model is likely the best solution in terms of performance. Given real-world sources of uncertainty, however, it is unlikely that any single MM-MDP model matches the true, underlying model closely enough to ensure beneficial actions are likely to be chosen. This is a caveat of the proposed MM-MDP methodology and partially motivates the other proposed methods, which do not suffer from this issue, to be described in later sections.

### 4.6.2 Multi-Model Partially Observable Markov Decision Process (MM-POMDP)

The MM-MDP framework presented in the previous section can also be extended to the Partially Observable Markov Decision Processes, resulting in the Multi-Model Partially Observably Markov Decision Process (MM-POMDP) framework. A POMDP is defined by the tuple $\langle S, A, R, T, \Omega, O \rangle$, where $S$ is the state space, $A$ is the action space, $R$ is the reward function, $T$ is the state transition function, $\Omega$ is the observation space, and $O$ is the observation function. In contrast, the proposed MM-POMDP is defined by the tuple $\langle S, A, R, \mathbb{T}, \Omega, \mathbb{O} \rangle$, where instead of a single state transition model $T(s', a, s)$ and a single observation model $O(s', a, o)$, a set of multiple state transition models $\mathbb{T} = \{T_1(s', a, s), \ldots, T_N(s', a, s)\}$ and a set of multiple observation models $\mathbb{O} = \{O_1(s', a, o), \ldots, O_M(s', a, o)\}$ are utilized. All of these models follow the same structure, but they differ in terms of parameters. Each unique combination of $T \in \mathbb{T}$ and $O \in \mathbb{O}$ forms an *uncertainty model state* $\xi$, or simply *model* for short. Each $\xi \in \Xi$, along with the remaining MM-POMDP elements, can each be considered as separate, individual POMDPs. Therefore, separate, individual action policies can be solved for every model (i.e., $\forall\, \xi, \exists\, \pi_\xi\, (b_\xi(s))$). Each individual POMDP decision-maker also tracks its own state belief $b_\xi(s)$ separately, using its specific transition model $T_\xi(s', a, s)$ and observation model $O_\xi(s', a, o)$, but based on the single action $a$ selected by the arbitration step, combining the outputs of multiple decision-makers, and single observation event $o$ that occurs, based on the unknown, true, underlying model. For every decision-maker, the state belief is updated via the standard POMDP belief update equation, as shown in (3.11), but for every

model $\xi \in \Xi$.

$$b'_{\xi}(s') = \eta O_{\xi}(s', a, o) \sum_{s \in \mathbb{S}} \left[ T_{\xi}(s', a, s) b_{\xi}(s) \right], \forall\, \xi \in \Xi \tag{4.4}$$

Where $\eta$ is a normalization constant.

Each individual POMDP policy $\pi_{\xi}\left(b_{\xi}(s)\right)$, based on a particular uncertainty model $\xi$, and it's associated state belief $b_{\xi}(s)$ is referred to herein as a *decision-maker* $D(\xi)$. The set of all decision-makers $D(\xi) \in \mathbb{D}(\Xi)$ is the realization of an MM-POMDP. The remaining MM-POMDP elements $\langle S, A, R, \Omega \rangle$ are assumed to be common to all decision-makers, so only the uncertainty model $\xi$ needs to be specified to distinguish one decision-maker from another.

The challenge now is to develop a way to arbitrate, or combine the outputs of each action policy from each decision-maker, as was the case for the MM-MDP framework. Each policy $\pi_{\xi}\left(b_{\xi}(s)\right)$ will output an action to perform, based on its particular model and its particular state belief. In the end, however, the system can only perform one action at each time step, so the outputs must be combined in an informed way that leverages the potential benefits that multiple decision-makers, that can approach a problem in different ways, can provide. The same set of proposed methods for the MM-MDP (majority voting, weighted majority voting, highest preference, and weighted highest preference) can be extended to the MM-POMDP. The modified versions of these methods, based on the same concepts as for the MM-MDP, are presented below, starting with majority voting.

**Algorithm 4.3:** MM-POMDP Action Majority Voting

```
1 Function MajorityVote(Ξ, Π, b_ξ(s), A):
2     z ← [0, 0, ..., 0] : |z| = |A|
3     for i ← 1 to |Ξ| do
4         a ← π_i(b_{ξ_i}(s))
5         z[a] ← z[a] + 1
6     return argmax_a z[a]
```

Where $z$ is the vote tally for each action $a \in A$. Again, as with the MM-MDP, this gives every decision-maker an equal vote and does not incorporate any information about which decision-

makers' models better match the true underlying model. Therefore, the same concept of "belief of the uncertainty model" (BUM) is extended to the MM-POMDP. The concept is to weight the probability of the observation event $o$, given the state $s$ and the action taken $a$ with the state belief of a particular decision-maker $b_\xi(s)$, to describe how well the observation fits that decision-maker's model $\xi$. This is done as follows.

$$b'(\xi) = \eta \sum_{s' \in \mathbb{S}} \left[ O_\xi(s', a, o) \sum_{s \in \mathbb{S}} \left[ T_\xi(s', a, s) b_\xi(s) \right] \right] b(\xi), \forall\, \xi \in \Xi \tag{4.5}$$

Where $\eta$ is a normalization constant.

And again, as the MM-MDP, this BUM metric can be used as weights to create a weighted majority voting arbitration scheme, shown below.

**Algorithm 4.4:** MM-POMDP Action Weighted Majority Voting

```
1 Function WeightedMajorityVote(Ξ, Π, b_ξ(s), A, b(ξ)):
2     z ← [0, 0, ..., 0] : |z| = |A|
3     for i ← 1 to |Ξ| do
4         a ← π_i(b_{ξ_i}(s))
5         z[a] ← z[a] + b(ξ_i)
6     return argmax_a z[a]
```

Where $z$ is the weighted vote tally for each action $a \in A$. Again, as with the MM-MDP, this results in a loss of *diversity* upon convergence to a single model, or subset of models, which motivates other proposed methods discussed in later sections.

## 4.7 Revisited Case Study: Simulated Multi-Model Single Robot Foraging

The possible improved resilience to ambiguity of multi-model decision-making is investigated first by implementing an MM-MDP solution to the robot foraging problem as described in Section 4.4 and comparing this solution to the heuristic FSM solution and the traditional single MDP solution

| Model | NE cluster $P_{\text{grab}}$ | NW cluster $P_{\text{grab}}$ |
|---|---|---|
| $T_0$ | 0.9 | 0.9 |
| $T_1$ | 0.5 | 0.5 |
| $T_2$ | 0.3 | 0.9 |
| $T_3$ | 0.7 | 0.1 |
| $T_4$ | 0.5 | 0.0 |

**Table 4.7.1:** MM-MDP Robot Foraging Transition Models

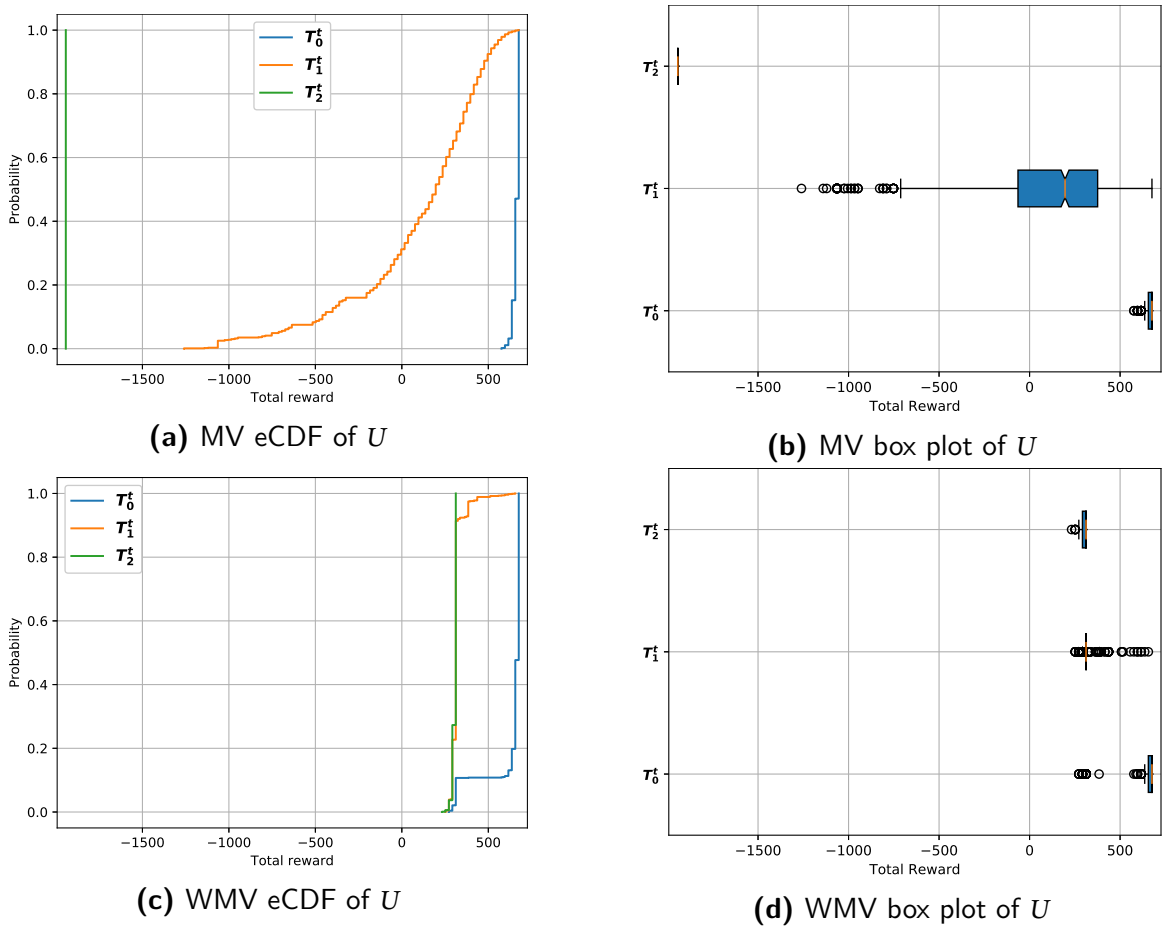
results shown in Section 4.5. Recall that in the foraging problem described in Section 4.4 as far as the robot knows, its probability of successfully acquiring food is a fixed probability and does not depend on any other factors. In the true system, implemented by the simulation, however, the robot's ability to grab food is dependent on an additional "heading" state of both the robot and the food. If the robot's heading and the food's heading are aligned, the robot's probability of picking up the food is high and the probability decreases as the difference in heading is greater. The robot's decision-maker does not know about this additional state which causes this source of uncertainty, however, in order to give the MM-MDP solution the chance to account for this ambiguity, the definition of the problem described in Section 4.4 is extended to enable the MM-MDP robot to make a wider range of choices. This is done by giving the robot knowledge that food may be located in one or more *clusters* and that food in different clusters may have a different $P_{\text{grab}}$. The reason this knowledge of clusters and their possibly different $P_{\text{grab}}$ is necessary is to enable the robot to make decisions about which clusters to visit and attempt to grab food. One cluster may be closer to home than another, enabling the robot to traverse less distance, but it may have a lower probability of successfully grabbing food at that cluster and it would be more successful at its mission traveling to the farther cluster. But since $P_{\text{grab}}$ is dependent on a hidden heading state, the robot's assumption about the $P_{\text{grab}}$ value for a cluster may be incorrect, leading to ambiguity.

To define an MM-MDP, a set of multiple transition models are used to consider different possibilities for the grab success rate. The underlying reasons for these different grab probabilities are not understood by the robot's decision-maker; it simply understands that a fixed assumption about the grab success probability may be incorrect. A set of five transition models $\{T_0, T_1, \ldots, T_4\}$ are considered by the MM-MDP decision-maker and their specific values of $P_{\text{grab}}$ assumed for each cluster are described below in Tab. 4.7.1.

Using the same food map and grid world configuration as was used in the case study described in Section 4.5, a set of separate MDP policies $\pi_T(s)$ are solved for each $T \in \mathbb{T}$. Each of these policies are tested using both the majority voting (MV) and weighted majority voting (WMV) arbitration schemes for three different true underlying model conditions. The true underlying models are three different cases of the unknown food heading for each of the two food clusters. These true models are the same as were used in the case study in Section 4.5 and have the same parameters as presented in Tab. 4.5.1

The performance of each decision-making strategy is compared by examining the distribution of the accumulated reward $U$, the same as described in the case study presented in Section 4.5 and (4.2). These results are presented in Fig. 4.7.1 and Tab. 4.7.2. For the majority voting method, the performance across all three true scenarios follows a similar trend to that of the single MDP scenario, presented in Section 4.5.3. It does, however, have more outliers, especially for true model $T_1^t$. And again, the same failure mode is encountered for true model $T_2^t$, where the robot repeatedly tried to pick up food it cannot pick up and incurs nothing but failed grab penalties, for the entire duration of the simulation, for all Monte Carlo trials. This is worthy of note, because one of the decision-maker's models, $T_5$, does describe the scenario where food in the north-west cluster cannot be picked up. However, this decision-maker's proposed actions get out-voted each time, during the majority voting arbitration process, due to the other four decision-makers believing that there is a chance to pick up food from the north-west cluster. The decision-maker based on model $T_5$ essentially gets treated as an outlier, by the other decision-makers, based on the other models and even if the *physical feedback* the robot is getting about repeated failures to grab is demonstrating that this model may better describe the situation than any of the other models.

The weighted majority voting strategy is a way to incorporate this physical feedback. It can be seen that the scenarios with weighted majority voting generally performed better than majority voting, especially for the case of true model $T_2^t$. In this case, unlike with the single MDP and majority voting, the robot is able to perform the mission in a limited capacity and not get stuck trying to pick up food that is impossible for it to pick up. The robot must first experience some failures to gain some information through physical feedback, but once this information is gained, the votes of models that agree with the physical feedback are given stronger influence over decision-making, since they better describe the situation the robot has encountered. This combination of *diversity*




**(a)** MV eCDF of $U$

**(b)** MV box plot of $U$

**(c)** WMV eCDF of $U$

**(d)** WMV box plot of $U$

**Figure 4.7.1:** MM-MDP foraging solution results, comparing majority voting (MV) and weighted majority voting (WMV), for three different true models: $[T_0^t, T_1^t, T_2^t]$.

and *physical feedback*, using physical feedback to arbitrate diverse decision-makers, based on which ones are more likely to propose successful solutions, is a key outcome for the objectives of this research. This method's resilience to ambiguity, however, is highly dependent on having a set of sufficiently diverse models to create sufficiently diverse decision-makers. An additional case study on a different decision-making problem, a simulated traffic intersection controller, is provided in Appendix A. This problem is modeled as partially observable and implements the MM-POMDP framework. Some final conclusions about the multi-model decision-making frameworks investigated thus far are presented next.

| | $T_0^t$ | $T_1^t$ | $T_2^t$ |
|---|---|---|---|
| Max | 0% | -2.96% | 116% |
| Upper Quartile | 0% | -17.02% | -116.1% |
| Mean | -5.46% | 233.0% | 116% |
| Median | 0% | 59.18% | 116.0% |
| Lower Quartile | 0% | 588% | 115% |
| Min | -52.78% | 120.0% | 111.9% |
| Std Dev | 551.9% | -90.21% | $\infty$% |

**Table 4.7.2:** Percent change between WMV and MV (baseline) of major statistics for three different true models: $[T_0^t, T_1^t, T_2^t]$.

## 4.8 Multi-Model Decision-Making Conclusions

The multi-model decision-making frameworks described and demonstrated through the case studies above partially address the research objective of increasing the resilience of decision-making to ambiguity through the addition of *physical feedback* and *diversity* to aid the decision-maker in choosing actions based on different models, using an estimate of its "uncertainty of the uncertainty model" (UoU). This UoU estimate encodes the physical feedback and the library of different models to arbitrate provides the diversity. With all of these scenarios, however, the different proposed actions by each diverse decision-maker must be *arbitrated* down to one action to be taken at each time step, because it is one system that can only take one action at a time. While diversity is used to propose several different possible actions, some of the benefit of diversity is lost in the arbitration step.

Also, these multi-model frameworks only operate within a limited space of models, provided to them by the human designer. They cannot handle ambiguity that is not at least partially described by one or more of the models they are provided with. While this is an interesting approach to consider for some types of decision-making problems, it does not fully satisfy the motivations of this research. What is of more interest is investigating ways in which decision-makers can overcome ambiguity without needing to explicitly estimate the ambiguous distributions of uncertainty. Discussions leading to a proposed framework that works towards achieving this are presented next.

## 4.9 The Curse of Ambiguity

The use of multiple uncertainty models in the decision-maker is a naïve way to provide resilience to ambiguity, but one that can be effective if enough of the space of possible ambiguity scenarios is considered. Other methods exist as well which also attempt to incorporate an estimate of the parameters defining the ambiguity [54–57] and some make use of an "ambiguity attitude" [57] to guide the selection of actions. In many cases, however, these existing decision-making under ambiguity methods suffer greatly from the curse of dimensionality and are often only applicable to the simplest of problems. Even decision-making under uncertainty methods that do not incorporate ambiguity suffer from the curse of dimensionality, because outcomes are represented over probability distributions. Considering ambiguity only compounds this problem, because now outcomes are represented over distributions of distributions. This is referred to as the "curse of ambiguity" [57]. Therefore, most existing methods do not sufficiently address the motivation of this research, which is to improve robotic decision-making in the real-world, where problems can be assumed to be simple enough to solve with these methods. Further research into other methods that work to mitigate these limitations is needed.

## 4.10 Research Hypothesis: Trial and Error Methods

The curse of ambiguity makes solving problems involving ambiguity computationally challenging. Methods that exhaustively search the state-action space to find optimal solutions are generally intractable and approximate methods are needed. Even for approximate methods that use informed search, such as MCTS, addressing high-dimensionality problems, this curse of ambiguity is still a challenge. For these methods, the challenge tends to arise from the need to sufficiently sample the state-action space of the problem in order to either attempt to model the ambiguous distributions of uncertainty or attempt to indirectly model them through directly approximating the "optimal" state-action value function $Q(s, a)$. This is the classical philosophy of the field of decision-making: attempt to develop a globally consistent, top-down understanding of the problem and then apply a decision-making strategy based on that understanding and an estimate of the "quality" of that understanding, if applicable (e.g., similar to estimating "belief" in a POMDP). In order to explore other solutions to the curse of ambiguity, a step must be taken outside of this paradigm.

One category of decision-making strategies that, when utilized correctly, can escape this paradigm are trial and error methods. Trial and error decision-making is defined in this context as a decision-making method that uses a source of randomness to alter the decision-making strategy when outcomes are judged to be not providing sufficient payoff or not making sufficient progress towards the objective [112]. It is hypothesized that trial and error methods can be applied to complex decision-making under ambiguity problems to both address the computational issues of the curse of ambiguity and to investigate decision-making that is resilient to ambiguity. Therefore, the following research hypothesis is posed:

> Informed trial and error methods can solve complex decision-making under ambiguity problems while providing resilience to ambiguity and without significant computational cost.

A common challenge with trial and error methods, however, is informing the sampling to select new strategies in an informed way that is not just uniformly random. In the context of decision-making under ambiguity and the objectives of this research, the three key factors outlined earlier (physical feedback, diversity, and swarm local interactions) will be used to inform the trial and error methods proposed next.

## 4.11 Proposed Framework: Ambiguity Trial and Error (AT&E)

Let a decision-making problem be defined in a similar manner to a Markov Decision Processes (MDP), with a set of states $S$, a set of actions $A$, a transition function $T$ and a reward function $R$. Only discrete problems are considered, for simplicity. The transition function $T$ describes the model of how actions, taken from a current state, map to new state outcomes. Some outcomes may be deterministic, but in general, the transition function $T$ represents probabilistic outcomes. However, the transition function is also parameterized by one or more finite valued *ambiguity parameters* $\theta \in \Theta$. This results in a transition function defined as follows.

$$T(s', a, s; \theta) = P(s'|s, a; \theta) \tag{4.6}$$

Different values of these ambiguity parameters $\theta$ result in different probability mass distributions that affect the outcome of stochastic state transitions. Parametric distributions in decision-

making processes are not a new concept; parametric MDPs or pMDPs are an existing field of study [113]. Most existing methods assume, however, that the decision-maker has full knowledge of the transition model $T(s', a, s; \theta)$ for every value of $\theta$. AT&E relaxes this assumptions by only assuming that the decision-maker has knowledge of the number of ambiguity parameters $\theta \in \Theta$ and their ranges of possible values. It does not know the true values of $\theta$, nor does it necessarily know how the state transition model is defined for all values of $\theta$. The decision-maker may have an *initial estimate* of the probabilities of the possible state transition outcomes for one or more values of $\theta$, but not for all values of $\theta$. Additionally, even for the known values of $\theta$, the initial estimate of the state transition probabilities could be incorrect. It is also possible that the values of $\theta$ may be parametrically dependent on the state $\theta(s)$. For example, they may vary with position or time. These are the sources of *ambiguity* in this problem framework. The robot must decide how to choose actions to attempt to achieve its objective, defined by the reward function $R$, given that the distributions of uncertainty involved in the problem are not fully known, that the initial knowledge of the known portions may be incorrect, and that the distributions may vary parametrically.

The proposed framework to solve this problem is known as Ambiguity Trial and Error (AT&E). The core concept of this framework is different than most "top-down" decision-making frameworks. The objective is not to attempt to jointly estimate explicit values of the state transition probabilities in $T(s', a, s; \theta)$ as well as attempt to develop a model of $\theta(s)$ over all states. Rather, the idea is to use informed trial and error methods, sampling over different possible values of $\theta$, to find actions that lead to desirable outcomes, in terms of $R$, when the initial knowledge of a portion of $T(s', a, s; \theta)$ (if any) is not leading to successful outcomes. Essentially, try different actions, based on sampling different values of $\theta$, and find ones which lead to action strategies that work better at achieving the objective. The key source of prior knowledge here is that the space of ambiguity parameters $\Theta$ is known, so the sampling occurs over a finite space of possible values.

There are two key innovations presented by AT&E. The first is a *mission progress monitor* (MPM) function that estimates if the robot's current decision-making strategy, based on the current values for the ambiguous parameters $\theta$, is making sufficient progress towards the robot's objective, or if ambiguity parameter sampling should be triggered for the parameters associated with the current subtask. The definition of the MPM is specific to the decision-making problem to which this framework is applied; however, the general purpose of it is always the same: recognize that progress is not being made and choose when to sample a new value for a particular ambiguity parameter. This

aspect of AT&E could be an entire field of study on its own, since deciding that sufficient progress towards the objective is or is not being made may be challenging in some problems. It is not the aim of this research to investigate this aspect thoroughly, so it is acknowledged that some of the methods used in the MPM in the case studies may be overly simplistic.

The second innovation, which is the larger focus of this work, is informing the ambiguity parameter sampling strategy, based on information observed about the problem, to be more likely to find successful action strategies. Retaining a full history of outcomes and repeatedly incorporating new experience into an attempt at modeling the distribution over $\theta$ in an attempt to develop the "most accurate" sampling strategy is not in line with the philosophy of this framework and would be very computationally expensive. Instead, heuristic strategies are employed to weight the sampling of $\theta$ to be more informed than uniform random sampling, when possible. This is where the three key factors of **physical feedback**, **diversity**, and **swarm local interactions** come into play. The decision-making agent uses its own most recent experience of succeeding or failing at the task, and the most recent experience of other agents it may encounter if it is part of a swarm of multiple agents, to bias the sampling of $\theta$ towards successful outcomes and away from unsuccessful outcomes. The operation of is shown in Alg. 4.5.

**Algorithm 4.5:** Ambiguity Trial and Error

**Procedure** AT&E$(s, \theta)$
  $a \sim \pi(s, \theta)$
  $s' \sim W(s, a)$
  $s_o \leftarrow C(s')$
  $r \leftarrow R(s', a, s)$
  **if** $MPM(r, s', a, s)$ **then**
    $P(\theta|s', s_o) \leftarrow$ weightSampling$(\theta, s', s_o)$
    $\theta \sim P(\theta|s', s_o)$
  **return** $s', r, \theta$

Where $\pi(s, \theta)$ is the agent's action policy, given the current state and currently assumed values of the ambiguity parameters $\theta$, $W(s, a)$ is the interface to the real world, where an action is taken and a new state outcome is returned based on an ambiguous stochastic process, $C(s')$ is the communication function, which gets the states of the other agents $s_o$ within communication range, if any,

when the agent is at state $s'$, $R(s', a, s)$ is the agent's reward function, $\text{MPM}(r, s', a, s)$ is the mission progress monitor function, and $\textsc{weightSampling}(\Theta, s', s_o)$ produces a probability distribution over $\theta$, based on the robot's own state $s'$ and the states of the other visible agents $s_o$, to be used for sampling a new value of $\theta$. Both MPM and $\textsc{weightSampling}$ must be specifically defined for the particular problem to which this framework is applied. The action policy $\pi(s, \theta)$ may in general be a stochastic policy, though this is not a strict requirement of the AT&E framework. This means that for a given prior state $s$ and ambiguity parameter value $\theta$, the chosen action may be stochastic. This is not a strict requirement and a determinisitc policy may be used if it is more appropriate to the problem. The reason for allowing stochastic policies is that problems involving ambiguity may be better addressed by a stochastic policy (i.e., trying different actions in order to explore different outcomes, since the distributions of uncertainty are not known). In the case of AT&E, the ambiguity parameters $\theta$ can be used to weight the stochastic action choices, biasing them towards actions that lead to more successful outcomes.

The weighting of the ambiguity parameter sampling, based on the agent's own experience, and that communicated to it by other swarm agents as it encounters them, is the key focus of this algorithm. The most recent experience is the only experience kept to avoid computational issues due to retaining long histories and to enable the agent to react faster to non-stationary ambiguity that may be changing rapidly. This is leveraging *physical feedback* from the environment to influence the agent's decision-making. Avoiding becoming locked into making decisions only one way is important to remain resilient to ambiguity, especially when the ambiguity may be non-stationary and may not remain constant with location or time. The agent must be able to maintain *diversity* in its choice of actions in order to explore other possibilities or overcome non-stationary ambiguity. This is why a sampling strategy for new parameters of $\theta$ is used. Sampling new values only when the decision-making strategy is not making sufficient progress towards the goal, however, enables the agent to *explore* new strategies when it is not performing well, but keep the strategy it has when it is performing well and *exploit* its ability, at least at that moment, to achieve the objective as best it can. And finally, the incorporation of the same type of experience from other agents, in the case of the agent being a part of a swarm, can help to augment the agents own experience and provide more information than it would have gathered through its own physical feedback alone, possibly improving its ability to overcome ambiguities even further. This exchange of information and influencing of behavior through *swarm local interactions* also enables more complex behaviors to arise that may

not be possible in the case of single agents. The properties of swarms and the local interactions are discussed more next, since this is a key area where AT&E can leverage its full potential.

## 4.12 Swarm Local Interactions

In a distributed robotic swarm, every agent in the swarm does not have full knowledge or the ability to communicate with the entire swarm at any one time. The agents act mostly independently, attempting to achieve their objectives, based on their own knowledge, but the group of robots would not be considered a "swarm" if there was not at least some level of *interaction* between agents. In this work, swarm agents are able to interact by communicating with other agents in their local neighborhood (i.e., with other agents that are physically nearby) and exchanging information. This information exchanged during these local interactions may influence the agent's decision-making, resulting in different actions being chosen than would have been chosen otherwise, had there not been an exchange of information between the agent and its neighbors. An individual agent makes decisions based on its own objectives and knowledge (e.g., models) and *physical feedback* from the environment, gained from observations after taking actions. Local interactions between swarm agents enable the influence of physical feedback to spread beyond just one agent and indirectly influence the decision-making of other agents as well. This concept of information from different agents being exchanged and influencing their decision-making, if leveraged correctly, may result in what is known as *emergent behaviors*, which are actions of the swarm as a whole (or local neighborhoods within the swarm) that are not explicitly designed into any of the individual swarm agents' decision-making capabilities. Emergent behaviors are not straightforward to design into a swarm decision-making framework in a predictable way and attempting to do so is often contradictory to the purpose of emergent behaviors in the first place. Their purpose is to help the swarm solve the problem in ways it was not explicitly programmed to do and it is hypothesized that this is the mechanism through which a robotic swarm may demonstrate resilience to ambiguity.

This concept is related to and partially inspired by the Particle Swarm Optimization (PSO) algorithm [114, 115]. PSO uses swarms of "particles" distributed throughout the space of a problem to work to solve the problem by each particle sampling the objective function at their different "locations" within the problem space. They then use their own experience, plus that of other particles nearby in the search space, to influence their iterative movement through the search space to find

the "best fit approximation" at solving the objective. The key difference in how the proposed swarm AT&E framework operates, compared to traditional PSO, is in how the "objective" and "problem to be solved" is defined. Rather than solving for a "best fit" solution to scalar function defined over some space, as is done is PSO, AT&E is used to guide the selection of actions of each agent (or particle) in the swarm to carry out an evolving, stochastic, ambiguous decision-making process. Classical decision-making theory may argue that finding a "policy" to solve a decision-making process is simply a matter of finding a best-fit solution to the "value function" over all the space of state-action outcomes (i.e., approximating $Q(s, a)$) and that the purpose of a framework such as AT&E is fundamentally no different than that of PSO. As explained previously, however, doing so requires a considerable amount of accurate prior knowledge about the structure and parameters that define the dynamics of the problem (i.e., accurate models). When outcomes are randomly distributed (i.e., stochastic problems), then models must describe probability distributions over outcomes, increasing the complexity of the problem. This increase in complexity is significantly compounded when it cannot be assumed that this knowledge of the distributions of uncertainty is accurate, or in other words, when there is ambiguity. Attempting to model this explicitly leads to "distributions of distributions" over outcomes, which is computationally challenging, as described earlier when discussing the *curse of ambiguity* in Section 4.9. This generally makes finding solutions to decision-making under ambiguity problems, using classical methods, intractable for all but the simplest problems. It is the objective of AT&E, therefore, to use the concept of exchanging diverse information through swarm local interactions, in a similar manner to that of PSO, to solve complex decision-making under ambiguity problems without computational challenges of methods more directly based on classical decision-making theory. Therefore, to explore these concepts and evaluate the proposed AT&E framework, the robotic foraging problem that has been used in the previous case studies is extended both to enable the use of swarms, but also to add additional challenges and sources of ambiguity to make the problem a better approximation of a complex real-world situation that requires the use of these techniques.

## 4.13 Problem Description: Simulated Robotic Swarm Foraging

The grid world foraging problem description originally presented in Section 4.4 can be extended to include multiple robots, all trying to perform the same objective: pick up food and bring it home.

Each robot chooses its own actions to solve the foraging problem collectively, forming a *distributed swarm*. In order to give the robots the ability to have *local interactions* and the potential to develop *emergent behaviors* that provide resilience to ambiguity, collectively as a swarm, different robots in the swarm need to be able to communicate with their neighbors and exchange information. It is assumed that the robots have a limited communication range and can only exchange information with their neighbors when they are within a certain distance of each other. This distance threshold will be specified in the descriptions of the case studies that implement this problem formulation. The information the robots exchange with each other includes at a minimum their: x,y position, state of possessing food or not, and an identification (ID) number unique to each robot in the swarm. Additional information may be needed for later case studies and will be specified in the case study descriptions.

Some additional constraints are needed for simulating this swarm foraging problem. One is that robots are not allowed to occupy the same grid cell as other robots and robots are prohibited from selecting move actions which would move them to the same grid cell as another robot. For simplicity, it is also assumed that each robot in the swarm performs its actions sequentially, rather than all agents performing their actions simultaneously. In other words, robot ID 0 perceives the state of the environment, chooses its action, carries out that action and updates its states and the state of the environment first, then robot ID 1 performs the same sequence, and so on. Once all robots have performed their actions, then one time step in the simulation is complete. It is understood by the author that this sequential execution framework is limiting and gives preference to the earlier robots in the sequence, but optimizing the execution of swarm simulations is beyond the scope of this research, so this simple formulation is considered sufficient.

Also, in order to make the problem a better approximation of a real-world foraging problem, some of the assumptions that were made in the original formulation are removed. These assumptions were necessary previously in order to make the problem computationally tractable as an MDP, but the solution methods that will be applied moving forward are low computational cost methods, implemented within the AT&E framework Therefore, increasing the dimensionality of the problem to make it more realistic is not a significant concern. Specifically, the aspects of the problem that are being changed are:

1. The robot no longer has full knowledge of the map. It can only detect food within a "percep-

tion range," which is the same as the swarm local interaction communication range.

2. The robot no longer has an explicit concept of "food clusters." Without full knowledge of the locations of food in the map, this is of little use and significantly reduces the amount of prior knowledge the robot must be provided with about the problem. As described later in this section, the robot does assume the properties of food to be spatially correlated, but less directly than in the previous formulation.

The consequences of relaxing these assumptions is the robot must now search for food, rather than directly approach it at already known locations. This essentially splits the foraging decision-making problem into two subtasks: search and approach, whereas previously it was just one approach task.

The foraging problem is also further extended to include additional sources of ambiguity. The original source of ambiguity, food grab probability due to the unconsidered "heading" state, is still present, but an additional source of ambiguity, food approach direction, is added. This ambiguity parameter describes a phenomenon where the probability of the robot successfully grabbing food may also depend on the direction from which the robot approaches the food. The inspiration for this approach direction ambiguity comes from an unexpected situation that was encountered when performing initial tests of real-world foraging experiments, which will be described in a later section. Nonetheless, due to observing this phenomenon, it was decided to include it as a key part of the problem description. In terms of formally defining ambiguity parameters, as is needed to define an AT&E problem, the ambiguity parameter space $\Theta$ consists of a set of two parameters, the food grab probability $\theta_G$ and the food approach direction $\theta_A$. Both of these ambiguities are considered to possibly vary with location and with time. The parameters are formally defined as follows:

1. Food grab probability $\theta_G$: A scalar probability value between 0 and 1, representing the probability of grabbing food.

2. Food approach direction $\theta_A$: A discrete integer ranging from 0 to 8, each corresponding to a direction (E, NE, N, NW, etc.) from which the robot is to approach food it is attempting to grab.

Presented next is a case study implementing a simulated swarm foraging scenario with this updated foraging problem. The focus of this case study is to investigate the resilience to ambiguity

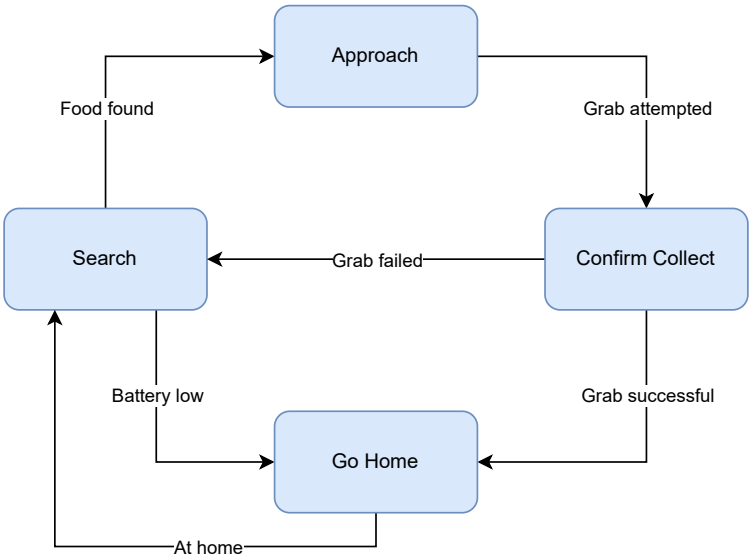



**Figure 4.14.1:** State transition diagram for the AT&E FSM action policy

of the proposed AT&E framework. The specific decision-making strategies used to implement a solution to the foraging problem, within the AT&E framework will be described, along with how the ambiguity parameters $\theta_G$ and $\theta_A$ are used to influence the decision-making.

## 4.14 Case Study: Simulated Robot Swarm Foraging

The swarm foraging problem described above in Section 4.13 is solved using a heuristic FSM solution, similar to the FSM solution presented in Section 4.5.1. This FSM solution is modified, however, to account for the increased problem complexity and additional sources of ambiguity, as well as to implement the AT&E framework.

### 4.14.1 AT&E FSM Solution

Instead of a *Select Target* FSM task, a *Search* FSM task is now implemented, since the map is now unknown and the robot must search for food. *Search* implements a stochastic search policy where search locations are selected in a weighted random fashion, when information for weight is available. The new FSM state transition diagram is shown in Fig. 4.14.1.

The information used to weight search location selection is the robot's current assumption about the food grab probability ambiguity parameter for every x and y location in the map (i.e., $\theta_G(x, y)$. In the interest of keeping computational requirements low and providing resilience to ambiguity, a purposefully naïve update rule is used for $\theta_G(x, y)$. The values of $\theta_G(x, y)$ are weighted based on the robot's own memory of the "last successful food location" and the "last failed food location," as well as those communicated to it by the other swarm agents with which it may have local interactions. In other words, the robot remembers both the most recent location it successfully picked up food and the location where it most recently failed to pick up food and communicates these locations to other agents during local interactions. To avoid computational issues associated with recording long histories of outcomes and also to avoid strong convergence, only the most recent memories are kept and the memories communicated by other swarm agents are cleared every time a robot visits home. As it encounters other swarm agents while searching, it will exchange memories with them again. Each successful food location, from both it and communicated from other agents, creates a peak in the distribution $\theta_G(x, y)$ and each failed food location creates a valley. The stochastic search policy then samples an (x, y) pair from this distribution and sets that as the next search location to which the robot will travel. The weighting based on the previous successful and failed locations increases the probability that locations near previously successful locations will be chosen and decreases the probability that locations near previously failed locations will be chosen. An example of a particular $\theta_G(x, y)$ grid map used for selecting search locations is shown in Fig. 4.14.2

Also similar with the previous FSM implementation, once the robot finds food, it may choose to approach it. There are two main differences with the implementation of the *Approach* FSM task than with the previous foraging simulations. The first is that the choice to approach visible food is based on a heuristic that compares the "value" of that food being grabbed with the distance-discounted "value" of food that could be grabbed elsewhere. Essentially, the idea of this heuristic is to encourage the robot to explore other areas of the map if it experiences a grab failure in a particular location. If a failure is experienced, this will reduce the grab probability ($\theta_G$) of any nearby food, since the robot assumes spatial correlation of food grab probability. If the distance-discounted value of food elsewhere in the map exceeds the non-discounted value of the nearby food being considered, the nearby food will not be approached. Otherwise, it will choose to approach the nearby food, as would normally happen. This heuristic was added during testing, when it was observed

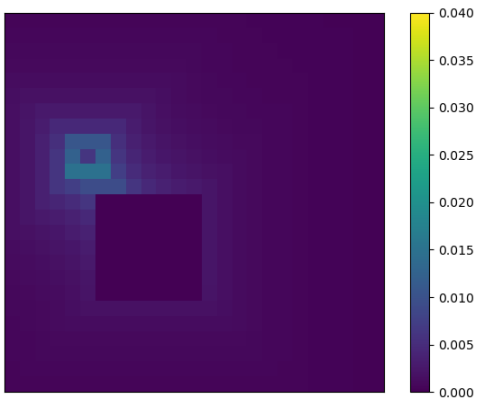



**Figure 4.14.2:** Example of the food grab probability map used to select search locations. Locations with higher probability are near previously successful locations and are more likely to be chosen by the stochastic search policy. Locations within a certain distance of the robot's current location are excluded in order to encourage the robot to explore farther locations. This is visible as the large dark square region. The robot is located at the center of this region.

that the robot would keep trying to grab all nearby, visible food even after a failure was experienced and some incentive was needed to make the robot explore other regions of the map after a failure.

The second difference with the *Approach* FSM task is the addition of maneuvering to an offset location, before approaching food and attempting to grab. This functionality is based on the approach direction ambiguity parameter $\theta_A$. The offset location to which the robot maneuvers before moving to the food location and attempting to grab is determined by the robot's current assumption of the value of $\theta_A$. For example, if the value of $\theta_A$ is north-east, the robot will move to the grid cell north-east of the food, before moving to the food location itself. As was mentioned previously, this ambiguity parameter and the directional approach logic was added to the AT&E FSM foraging problem after observing an unexpected source of uncertainty during testing of the real-world foraging experiment, analogous to this simulation. Further details on the configuration of the real-world experiment will be provided in Section 4.22.1, but the relevant aspect that motivates the inclusion of the $\theta_A$ ambiguity parameter is the wheel geometry of the holonomic robots used in the experiment. The robots used are a modified TurtleBot design that uses holonomic Kiwi drive. The robots have a roughly hexagonal top-down footprint, with wheels on three of the six sides. In order to grab the food pucks used in the experiment, the robot must drive over top of the food and

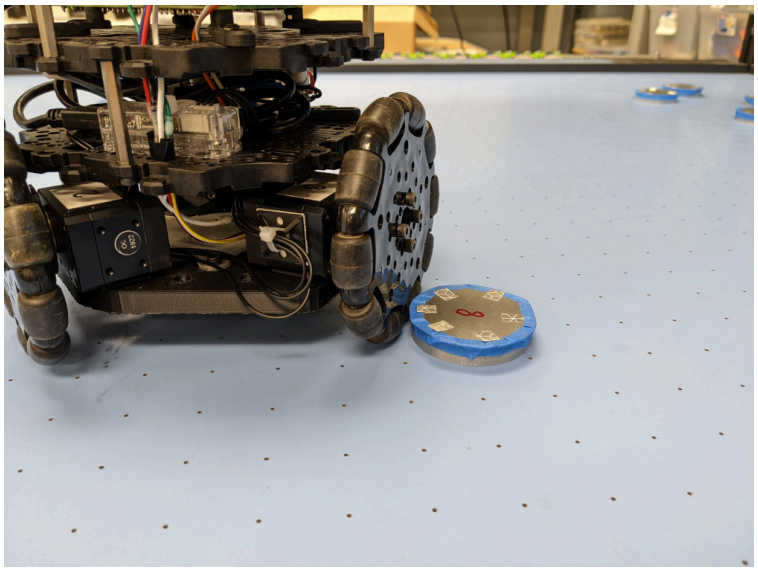


**Figure 4.14.3:** Example of the robots used in the real-world experiment pushing the food puck with its wheel and being unable to maneuver into position to grab the food. This was the motivation for the inclusion of the approach direction ambiguity parameter.

use an electromagnet mounted on its underside to attempt to pick up the food. It was discovered during testing that if the robot attempted to approach the food from any of the sides on which it has a wheel, it would push the food puck with its wheel and be unable to grab it. If it approached from a side without a wheel, however, it could successfully position the electromagnet near the food and have a chance of grabbing it. This "wheel pushing" situation is shown in Fig. 4.14.3.

Simple food pushing dynamics, based on the TurtleBot wheel geometry, were also added to the simulation to reflect this situation that was discovered through real-world testing. The robots are given no knowledge of these dynamics, or the uncertainty distributions associated with them, however. While knowledge of this approach direction uncertainty, due to the robot's wheel geometry, could have been hard-coded into the decision-maker's logic, it was decided that this represented an excellent opportunity to include another source of ambiguity in the problem. The robot is given the knowledge that the direction it approaches the food may affect its ability to successfully grab it (i.e., it has knowledge that there is a $\theta_A$ ambiguity parameter), but it does not know distribution of uncertainty over outcomes of different values for $\theta_A$. Therefore, this is a source of ambiguity.

The way the approach direction is handled by the AT&E FSM solution is the robot starts with an initial, uniformly random choice of the approach direction. If the robot is successful at grabbing food, using this approach direction, it continues with it. Once it experiences a grab failure, however, it then chooses a different approach direction, randomly. Each robot keeps a memory of the approach direction that was used with its last successful food grab and this is another piece of information that is shared between agents when local interactions occur. Therefore, when a robot needs to select a new approach direction, the random choice is weighted by memories of successful approach directions from other swarm agents it has encountered while searching. This means that more successful approach directions are more likely to be chosen, when sampling occurs. The robot does not simply mimic what the other agents do, however. It performs this trial and error process based on weighted random choice, so that it is more likely to choose what the rest of the swarm is finding to be successful, but sometimes it will make a different choice and explore other values of the $\theta_A$. While such a strategy may reduce the average performance of the robot, it provides a reasonable exploration-exploitation trade-off, in the face of ambiguity. Recall that resilience to ambiguity is more important to the objectives of this research than optimizing performance.

Both the MPM function and WEIGHTSAMPLING function, used to trigger and sample new values of $\theta_G$ and $\theta_A$ are separated into two pairs of functions that correspond to the two subtasks: search and approach. The $\theta_G$ parameter is sampled in the context of the search subtask and $\theta_A$ is sampled in the context of the approach subtask. The MPM function for $\theta_G$ is a "reset" function that purges the list of last successful and last failed food locations communicated from other swarm agents during local interactions. This reset is triggered if the number of search waypoints to which the robot travels and fails to find any food exceeds a threshold $a$ (chosen to be 5 in the following case studies). This is shown in Alg. 4.6.

**Algorithm 4.6:** Search Subtask MPM

```
1 Procedure MPM_{θ_G}(r, s', a, s)
2     if s'.numFailedSsearch > a then
3         return TRUE
4     return FALSE
```

The WEIGHTSAMPLING function for $\theta_G$ is what defines the discrete 2D PMF over the map, used

to sample search waypoints. An example of this is shown in Fig. 4.14.2 and is computed as shown in Alg. 4.7.

**Algorithm 4.7:** Search Subtask weightSampling

**Procedure** $\textsc{weightSampling}_{\theta_G}(\theta_G, s', s_o)$
  $F_s = [s'.F_s]$
  $F_f = [s'.F_f]$
  **for** $i \leftarrow 1$ **to** $|s_o|$ **do**
    $F_s.\textsc{append}(s_o[i].F_s)$
    $F_f.\textsc{append}(s_o[i].F_f)$
  $\theta_G \leftarrow \textsc{ones}(\textsc{size}(s'.\text{map}))$
  **forall** $(x, y) \in s'.map$ **do**
    **forall** $(x_s, y_s) \in F_s$ **do**
      $\theta_G(x, y) \leftarrow \theta_G(x, y)/(\textsc{dist}(x - x_s, y - y_s) + 1)$
    **forall** $(x_f, y_f) \in F_f$ **do**
      $\theta_G(x, y) \leftarrow \theta_G(x, y) \cdot (\textsc{dist}(x - x_f, y - y_f) + 1)$
    **if** $\textit{DIST}(x - s'.x, y - s'.y) < \beta$ **then**
      $\theta_G(x, y) \leftarrow 0$
  $\theta_G \leftarrow \theta_G/\textsc{sum}(\theta_G)$
  **return** $\theta_G$

Where $F_s$ is the list of successful food $x, y$ locations, $F_f$ is the list of failed food $x, y$ locations, and $\beta$ is the minimum distance threshold for how far a new search location should be from the robot's current location.

The MPM function for $\theta_A$ triggers sampling of a new value every time the robot experiences a failed grab attempt. This is shown in Alg. 4.8. The WEIGHTSAMPLING function for $\theta_A$ uses the last successful approach direction from each swarm agent to weight the selection of a new approach direction, from any of the eight possible approach directions, in the discrete grid world. This is shown in Alg. 4.9.

**Algorithm 4.8:** Approach Subtask MPM

```
1 Procedure MPM_{θ_A}(r, s', a, s)
2     if a == GRAB then
3         if s.hasFood == FALSE then
4             if s'.hasFood == TRUE then
5                 return TRUE
6     return FALSE
```

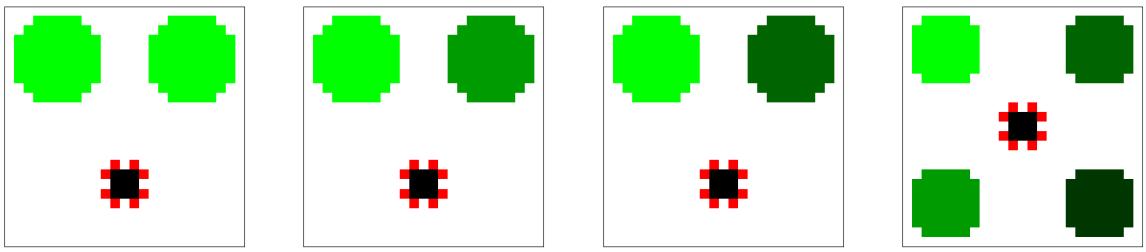
**Algorithm 4.9:** Approach Subtask weightSampling

```
1 Procedure weightSampling_{θ_A}(θ_A, s', s_o)
2     θ_A = ones(numApprDir)
3     for i ← 1 to |s_o| do
4         d ← s_o[i].d
5         θ_A[d] ← θ_A[d] + 1
6     θ_A ← θ_A / sum(θ_A)
7     return θ_A
```

Where $d$ is an approach direction (i.e., E, NE, N, NW, ...). The full code for running this implementation of foraging AT&E, in addition to the foraging simulation itself, can be found at `https://github.com/wvu-irl/foraging-sim-py`. The computational complexity of this implementation of foraging AT&E is driven by that of the weightSampling functions for both $\theta_G$ and $\theta_A$ for the search and approach subtasks, as shown in Alg. 4.7 and Alg. 4.9, respectively. The complexity of the search subtask is $O(kn)$ and the complexity of the approach subtask is $O(k)$, where $k$ is the number of swarm agents and $n$ is the size of the map (i.e., $n = |x||y|$). So overall, the worst-case complexity of this implementation of foraging AT&E is $O(kn)$.

### 4.14.2 Simulation Configurations

Several scenarios are tested with different parameters of the simulation varied to characterize the resilience to ambiguity of AT&E. One variable is the configuration of the map. Four different maps



**(a)** Model 0 **(b)** Model 1 **(c)** Model 2 **(d)** Model 3

**Figure 4.14.4:** Foraging maps used for the AT&E FSM simulations. The red grids represent robot locations, the black grids the home region, and the green grids food locations. Different shades of green represent different food headings.

are tested, with the first three maps (model 0, model 1, and model 2) having the same geometric configuration, but different food grab probabilities for certain food in the map. These scenarios are analogous to those tested in the previous foraging simulations, where one of the clusters of food has a different true "heading," making the grab probability different for each scenario. These grab probabilities are listed in Tab. 4.14.1.

| Model | NW Heading | NE Heading | NW true $P_{\text{grab}}$ | NE true $P_{\text{grab}}$ |
|---|---|---|---|---|
| Model 0 | east | east | 0.9 | 0.9 |
| Model 1 | east | north | 0.9 | 0.1 |
| Model 2 | north | west | 0.9 | 0.0 |

**Table 4.14.1:** True Robot Foraging Transition Models for the AT&E simulations, given a robot heading of East

A fourth map is also tested (model 3), which has a different geometric layout than models 0, 1, and 2, in order to verify that the AT&E FSM solution presented here generalizes to different environments. The configurations of all four maps are shown in Fig. 4.14.4.

In addition to the maps, the other parameters varied in these simulations are the number of robots and the perception/communication range of the robots. Scenarios with time varying ambiguity are also examined. All of the simulation variations are summarized in Tab. 4.14.2.
All of these variations, in addition to the performance of the AT&E FSM solution with two ambi-

| Parameter | Variations |
|---|---|
| Map | Model: 0, 1, 2, 3 |
| Number of Agents | 4, 8, 12 |
| Perception/Communication Range | 1, 2, 3 |

**Table 4.14.2:** Parameter variations in AT&E FSM foraging simulations

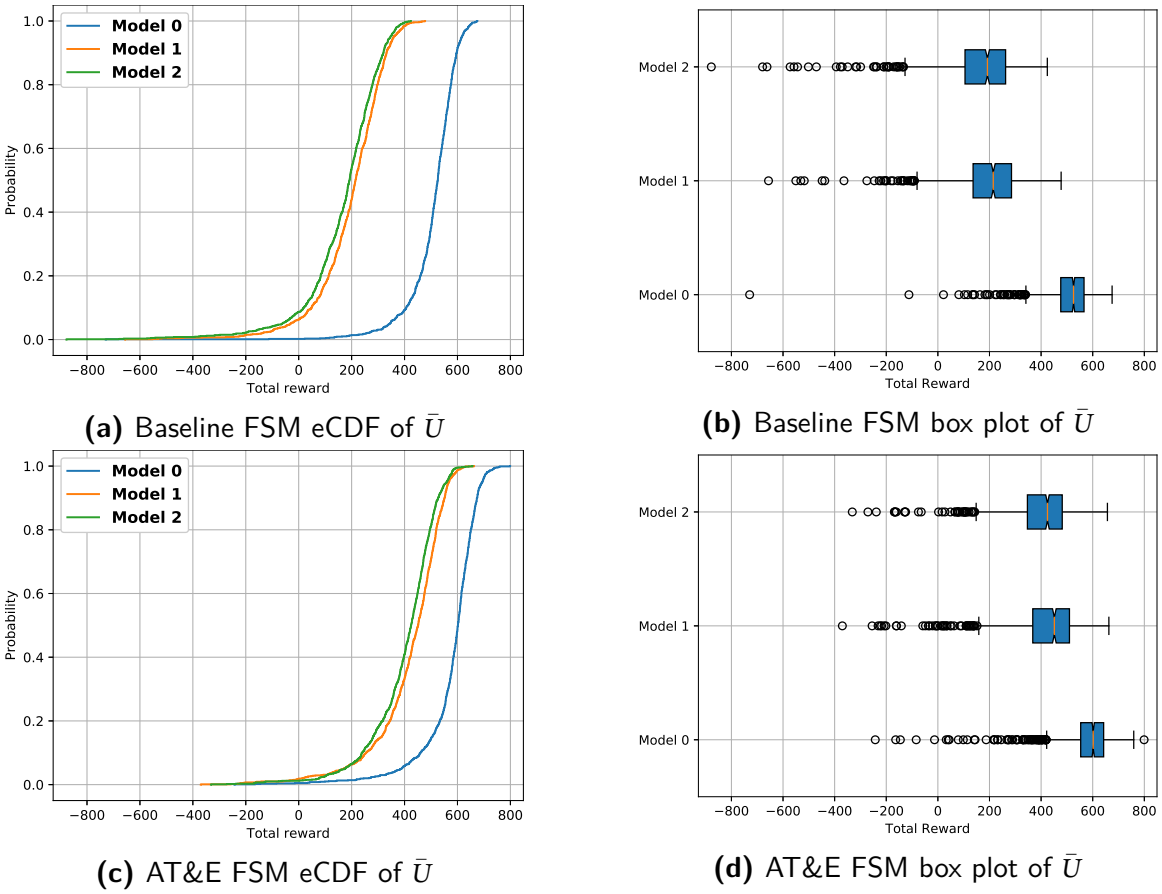
guity parameters, both separately and combined, are explored next.

### 4.14.3 Results and Discussion

When evaluating resilience to ambiguity, the performance of a particular solution must be compared to a baseline solution and both solutions must also be evaluated over a large number of trials, since the problems are stochastic. As was done previously in the initial case studies presented in Section 4.5 and Section 4.7, each method is simulated for 1000 Monte Carlo trials and results are characterized in terms of the eCDF over the total accumulated reward $U$. Simulations are also now run for 300 time steps. Since there are now multiple agents, however, the accumulated reward is summed for all agents and then normalized by the number of agents ($N$). The normalized accumulated reward $\bar{U}$ is computed as follows.

$$\bar{U} = \frac{\sum_{i=0}^{N-1} \sum_{t=0}^{t_{\max}-1} r(s_{t,i}, a_{t,i}, s_{t+1,i})}{N} \tag{4.7}$$

The baseline solution in all of the scenarios in this case study is that of a swarm that does not communicate information to other agents via local interactions, which is the core functionality enabling the ambiguity resilience of AT&E. A comparison to other existing methods, such as an MDP or reinforcement learning (RL) solution is not practical. In the case of the MDP, this solution is not practical due to the computational complexity that would be required to solve the problem with the additional factors added to make the problem more representative of real-world ambiguities. The addition of multiple agents also makes the dimensionality of the problem considerably higher, even though the agents are distributed and not centrally controlled. In the case of RL, this solution is not practical for the purpose of this case study, because of the prohibitive amount of training data that would be required. Some discussions on how this problem would be approached as an RL problem, however, are presented in Section 5.3, but solving such a complex problem in




**(a)** Baseline FSM eCDF of $\bar{U}$

**(b)** Baseline FSM box plot of $\bar{U}$

**(c)** AT&E FSM eCDF of $\bar{U}$

**(d)** AT&E FSM box plot of $\bar{U}$

**Figure 4.14.5:** Baseline and AT&E swarm foraging results for grab probability $\theta_G$ ambiguity only.

this manner is beyond the scope of this research, so a comparison against a simpler FSM solution that does not implement the full capabilities of AT&E as a baseline must suffice for the purpose of this research.

With this methodology of analyzing the results now established, evaluating the performance of the swarm AT&E FSM solution scientifically requires that different factors be isolated and examined separately. First, the ability of the AT&E solution to overcome the grab probability ambiguity $\theta_G$ is examined in isolation. Results from Monte Carlo trials of the baseline FSM and the full AT&E FSM solutions are presented in Fig. 4.14.5 and percent change in the major statistics of the distri-

| | Model 0 | Model 1 | Model 2 |
|---|---|---|---|
| Max | 18.37% | 38.73% | 54.91% |
| Upper Quartile | 13.42% | 78.72% | 83.61% |
| Mean | 13.81% | 113.05% | 135.22% |
| Median | 14.50% | 110.28% | 121.15% |
| Lower Quartile | 16.22% | 169.45% | 229.83% |
| Min | 66.74% | 43.60% | 62.22% |
| Std Dev | 14.02% | 3.25% | -11.94% |

**Table 4.14.3:** Percent change of major statistics between the baseline FSM and AT&E FSM solution for grab probability ambiguity.

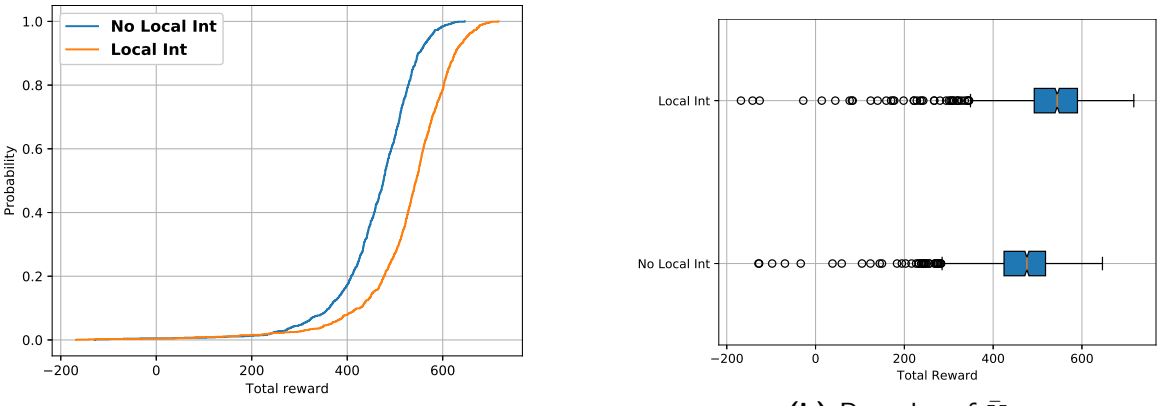
butions is presented in Tab. 4.14.3.

It can be seen that the AT&E method suffers less of a reduction in performance when faced with ambiguity in food grab probability than the baseline FSM. This is due to the swarm agents informing each other about the locations where they are successful and unsuccessful at picking up food and influencing the other agents' choices of search locations to more often choose to search near more successful locations.

Similarly, the approach direction ambiguity is examined in isolation. Results from Monte Carlo trials of the baseline FSM and the full AT&E FSM solutions are presented in Fig. 4.14.6 and the percent change in the major statistics of the distribution is presented in Tab. 4.14.4.

It can be seen again that the AT&E method suffers less reduction in performance when faced with ambiguity in the food approach direction than the baseline FSM. This is due to the swarm agents informing each other about what approach direction they used to successfully pick up food and influencing the other agents' choices of approach direction to more often choose more successful approach directions.

Next, both the grab probability and approach direction ambiguities are enabled together in the simulation, representing the full problem with all aspects of ambiguity in play. Results from Monte Carlo trials of the baseline FSM and the full AT&E FSM solutions are presented in Fig. 4.14.7 and the percent change in the major statistics of the distributions are presented in Tab. 4.14.5.



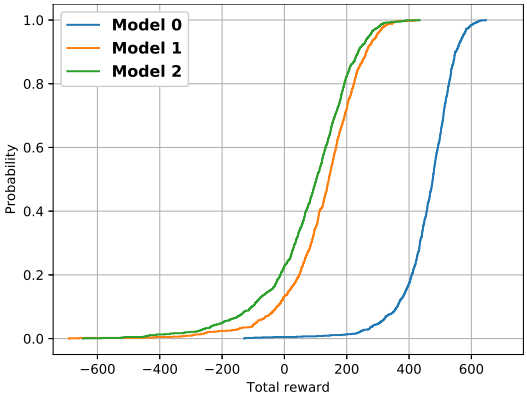

**(a)** eCDF of $\bar{U}$

**(b)** Box plot of $\bar{U}$

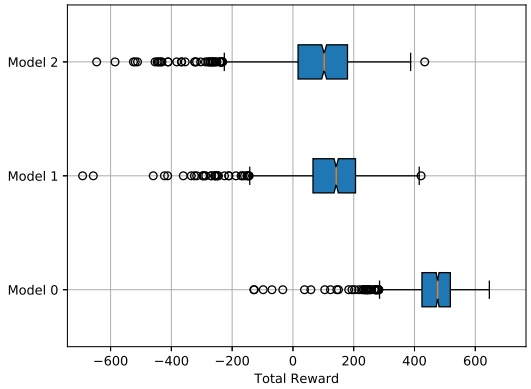
**Figure 4.14.6:** Baseline (no local interactions) and AT&E (local interactions) swarm foraging results for approach direction $\theta_A$ ambiguity only.

| | |
|---|---|
| Max | 10.93% |
| Upper Quartile | 13.89% |
| Mean | 14.68% |
| Median | 14.34% |
| Lower Quartile | 16.03% |
| Min | -30.61% |
| Std Dev | 11.69% |

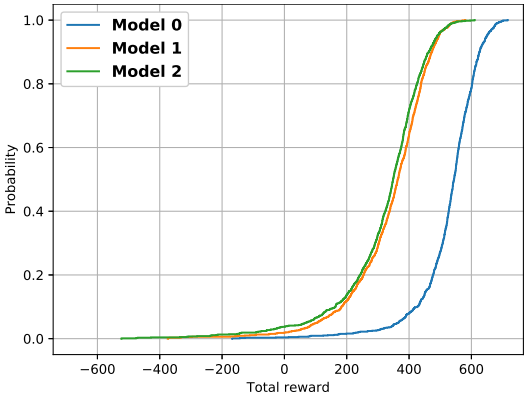
**Table 4.14.4:** Percent change of major statistics between the baseline FSM and AT&E FSM solution for approach direction ambiguity.



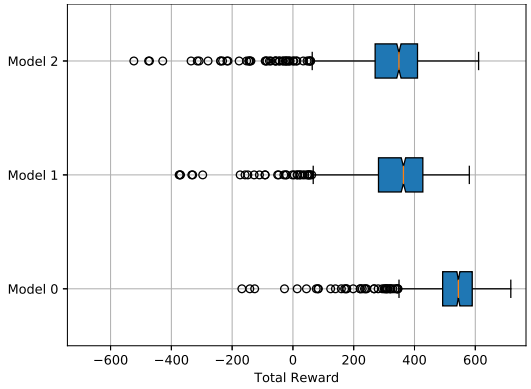

**(a)** Baseline FSM eCDF of $\bar{U}$




**(b)** Baseline FSM box plot of $\bar{U}$




**(c)** AT&E FSM eCDF of $\bar{U}$




**(d)** AT&E FSM box plot of $\bar{U}$

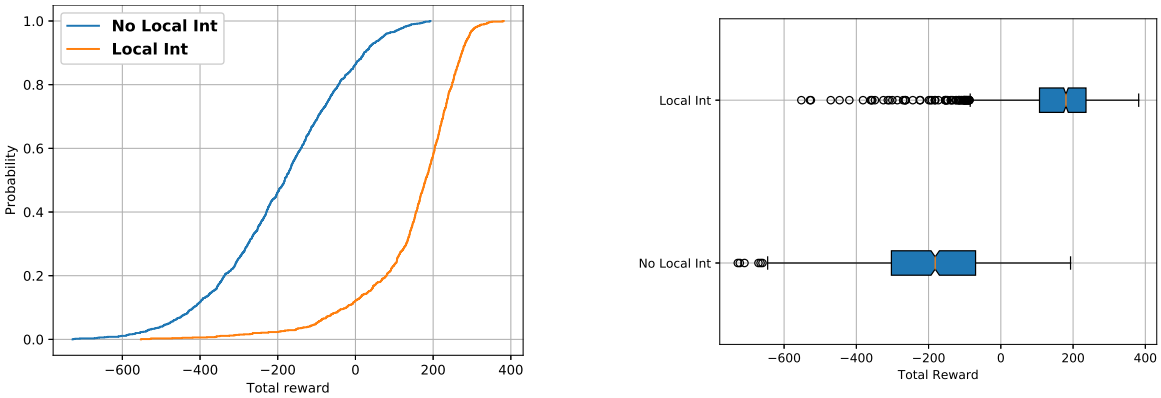
**Figure 4.14.7:** Baseline and AT&E swarm foraging results for both the grab probability $\theta_G$ ambiguity and approach direction ambiguity $\theta_A$ combined.

As expected given the results from testing the performance with the two ambiguity parameters independently, the AT&E method suffers less of a reduction in performance when faced with both forms of ambiguity than the baseline FSM. This is again due to the swarm agents informing each other about their assumptions of the parameters of ambiguity that led to successful behaviors.

| | Model 0 | Model 1 | Model 2 |
|---|---|---|---|
| Max | 10.93% | 37.65% | 40.88% |
| Upper Quartile | 13.89% | 107.59% | 128.32% |
| Mean | 14.68% | 170.27% | 301.10% |
| Median | 14.34% | 155.40% | 239.42% |
| Lower Quartile | 16.03% | 326.36% | 1499.82% |
| Min | -30.61% | 45.90% | 18.95% |
| Std Dev | 11.69% | 2.76% | -1.66% |

**Table 4.14.5:** Percent change of major statistics between the baseline FSM and AT&E FSM solution for both the grab probability and approach direction ambiguities.

In order to verify that the AT&E framework generalizes to different foraging scenarios and does not only perform well in the one map configuration (with different ambiguities) presented thus far, simulations are also performed on a different map configuration (Model 3 as shown in Fig. 4.14.4). Results from Monte Carlo trials of the baseline FSM and the full AT&E FSM solutions are presented in Fig. 4.14.7 and the percent change in the major statistics of the distribution is presented in Tab. 4.14.5. As expected, these results show the same trend as before.




**(a)** eCDF of $\bar{U}$

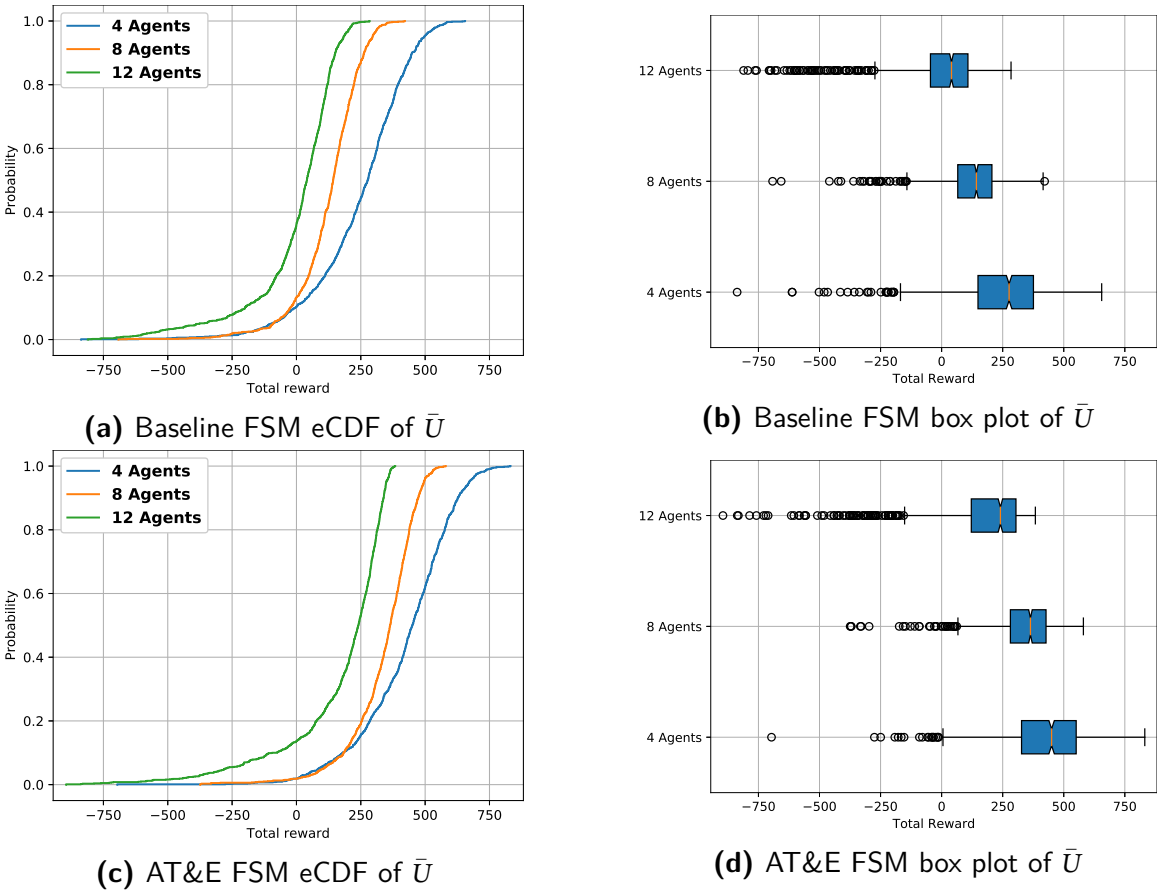
**(b)** Box plot of $\bar{U}$

**Figure 4.14.8:** Baseline (no local interactions) and AT&E (local interactions) swarm foraging results for both the grab probability $\theta_G$ and approach direction $\theta_A$ ambiguitiy combined for a different map configuration.

| | |
|---|---|
| Max | 10.93% |
| Upper Quartile | 13.89% |
| Mean | 14.68% |
| Median | 14.34% |
| Lower Quartile | 16.03% |
| Min | -30.61% |
| Std Dev | 11.69% |

**Table 4.14.6:** Percent change of major statistics between the baseline FSM and AT&E FSM solution for both the grab probability and approach direction ambiguities for a different map configuration.

In addition to examining the effects of the ambiguity parameters in isolation and combined, the sensitivity of the AT&E solution to the number of agents in the swarm is also examined. Scenarios are tested for 4, 8, and 12 agents in the swarm. The results above were all presented for the median scenario of 8 agents. These scenarios are tested on the Model 1 map, representing a "median" of the grab probability ambiguity. Results comparing the baseline FSM and the AT&E solution are presented in Fig. 4.14.9.




**(a)** Baseline FSM eCDF of $\bar{U}$

**(b)** Baseline FSM box plot of $\bar{U}$

**(c)** AT&E FSM eCDF of $\bar{U}$

**(d)** AT&E FSM box plot of $\bar{U}$

**Figure 4.14.9:** Baseline and AT&E swarm foraging results for different numbers of swarm agents in terms of the normalized accumulated reward across all agents.

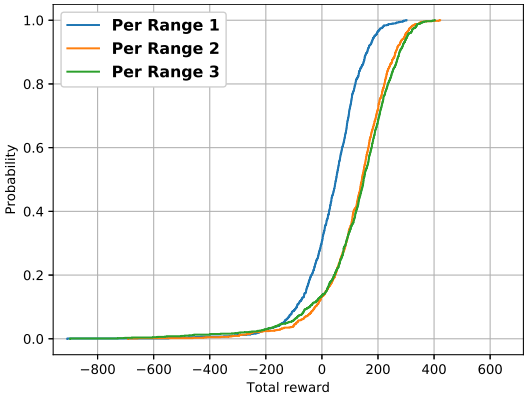
Interestingly, these results do not show the initially expected result. It was hypothesized that a larger number of agents in the swarm would result in better resilience to ambiguity, because the larger number of agents would both better sample the environment and spread information faster, due to more frequent local interactions. These results show, however, that the larger number of agents in the swarm has poorer performance. Initially this may look like a failure of the AT&E method, but in fact it is actually a failure of results interpretation. All of the results thus far have been presented in terms of the normalized accumulated reward $\hat{U}$ as defined in (4.7). While this quantity is "normalized" by the number of agents, this metric is not an accurate way to compare the performance of swarms with different numbers of agents. The number of food they may collect

and the number of failed grab attempts (i.e., penalties) they may accrue may not scale linearly with the number of agents in the swarm. Therefore, it makes sense that normalizing the accumulated reward by the number of agents, a linear operation, may yield a metric that is not comparable, since the effect of the number of swarm agents may be non-linear. Also, given the same environment configuration with finite food and a traffic bottleneck when many agents are attempting to return home at the same time, increasing the number of swarm agents may reduce the performance of the swarm overall, since more agents may be competing for limited resources. This is difficult to decouple from the way the results metrics are computed, however. This shows that the choice of results metrics is an important part of designing scientific experiments. It also shows that increasing the number of swarm agents may not result in monotonically increasing resilience to ambiguity.

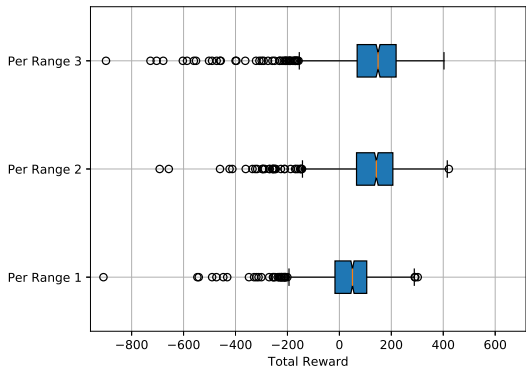
The concept of increasing the number of swarm agents was to increase the frequency of local interactions, hypothesizing that would lead to better resilience to ambiguity. Another way hypothesized to achieve that is to increase the perception range of the swarm agents, allowing them to communicate at longer distances, which should result in more frequent local interactions. Returning to a nominal value of 8 swarm agents, perception range scenarios of 1, 2, and 3 grid cells are tested. Results are presented in Fig. 4.14.10 and Tab. 4.14.7.

In the baseline scenario, it can be seen that the increased perception range provides increased performance by simply allowing the agents to detect food at a longer range, increasing their search efficiency. With AT&E and local interactions, however, the improvement provided by increased perception range is much more pronounced, showing that, as expected, the swarm performs better when there are more frequent local interactions.

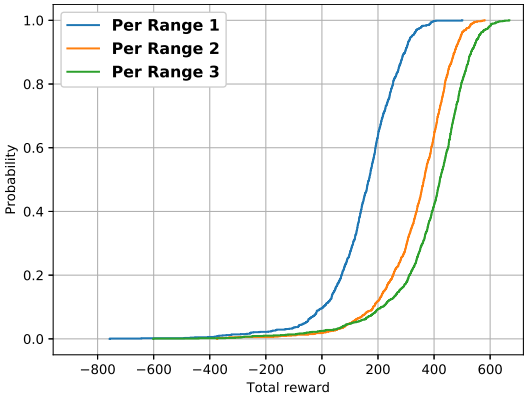
The same trend can be seen when varying perception range is tested on the alternate map configuration (Model 3), as shown in Fig. 4.14.11 and Tab. 4.14.8.



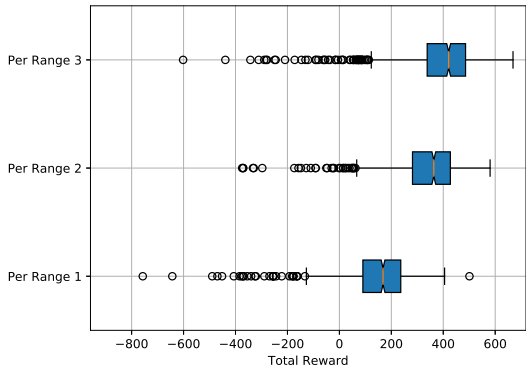

**(a)** Baseline FSM eCDF of $\bar{U}$




**(b)** Baseline FSM box plot of $\bar{U}$




**(c)** AT&E FSM eCDF of $\bar{U}$

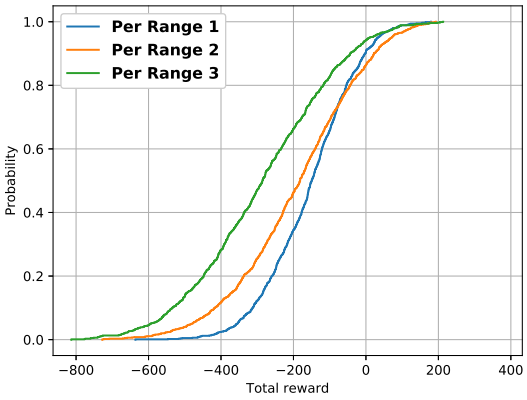



**(d)** AT&E FSM box plot of $\bar{U}$

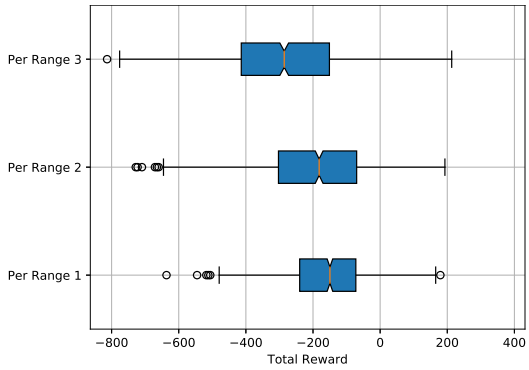
**Figure 4.14.10:** Baseline and AT&E swarm foraging results for varying swarm agent perception range scenarios.

| | Model 0 | Model 1 | Model 2 |
|---|---|---|---|
| Max | 10.93% | 37.65% | 40.88% |
| Upper Quartile | 13.89% | 107.59% | 128.32% |
| Mean | 14.68% | 170.27% | 301.10% |
| Median | 14.34% | 155.40% | 239.42% |
| Lower Quartile | 16.03% | 326.36% | 1499.82% |
| Min | -30.61% | 45.90% | 18.95% |
| Std Dev | 11.69% | 2.76% | -1.66% |

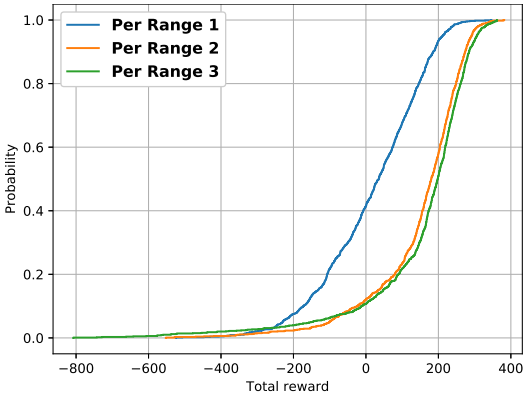
**Table 4.14.7:** Percent change of major statistics between the baseline FSM and AT&E FSM solution for varying swarm agent perception range scenarios.



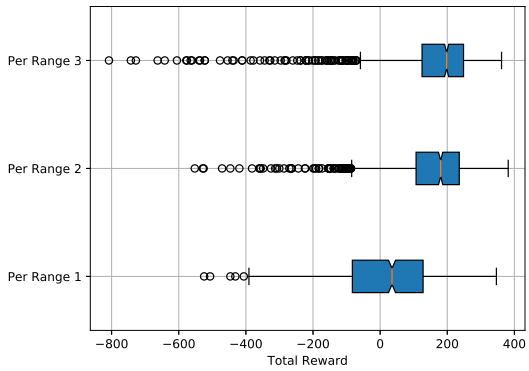

**(a)** Baseline FSM eCDF of $\bar{U}$




**(b)** Baseline FSM box plot of $\bar{U}$




**(c)** AT&E FSM eCDF of $\bar{U}$




**(d)** AT&E FSM box plot of $\bar{U}$

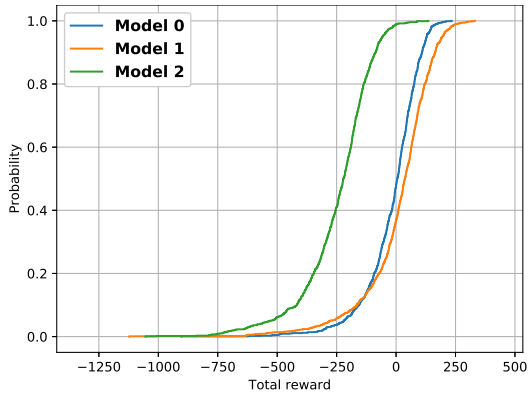
**Figure 4.14.11:** Baseline (no local interactions) and AT&E (local interactions) swarm foraging results for varying swarm agent perception range scenarios for a different map configuration.

| | Per Range 1 | Per Range 2 | Per Range 3 |
|---|---|---|---|
| Max | 92.97% | 97.73% | 69.64% |
| Upper Quartile | 276.81% | 437.13% | 264.47% |
| Mean | 111.38% | 178.40% | 154.72% |
| Median | 123.69% | 199.24% | 169.36% |
| Lower Quartile | 65.48% | 135.36% | 130.19% |
| Min | 17.62% | 24.15% | 0.69% |
| Std Dev | 15.67% | -22.20% | -15.30% |

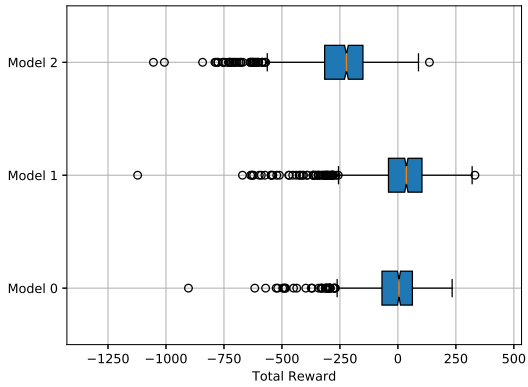
**Table 4.14.8:** Percent change of major statistics between the baseline FSM and AT&E FSM solution for varying swarm agent perception range scenarios for a different map configuration.

Finally, the last scenario tested for the swarm case study is that of time varying ambiguity. The ability to be resilient to this is one of the key motivations behind AT&E and one that sets it apart from other approaches, such as other decision-making under ambiguity frameworks, as well as reinforcement learning. In this scenario, the heading state that the swarm agents are not aware of that governs the food grab probability ambiguity changes half way through each simulation trial and the agents have no knowledge of this. A real-world analogy for the robot's changing their heading is solar powered robots that must change their orientation over time in order to keep their solar panels aligned with the sun. It is plausible in such a scenario that unmodeled dynamics, due to their heading, may exist and their effects on the foraging task are unknown, however, so this is still a source of ambiguity in this problem. Regardless, in this case study, their change in heading changes the grab probability of different food in the map, meaning that food clusters that once were easy for the robots to pick up may now be harder and ones that were harder or impossible may now be easier. The desired behavior in this case is that the swarm agents quickly "forget" the locations of food that have now become difficult to pick up and instead find and focus on food locations that have become easier for them to pick up. The results of this time varying scenario are presented in Fig. 4.14.12 and Tab. 4.14.9.

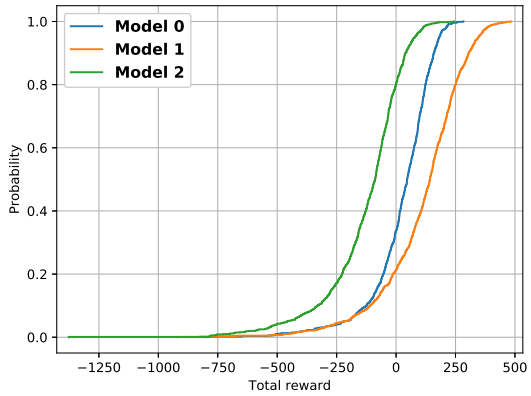
As expected, after the robot's heading changes, resulting in an ambiguity that is no longer stationary with respect to time, the AT&E method is able to react to this change and not remain "converged" to the strategy that was working well initially. A before and after example from one



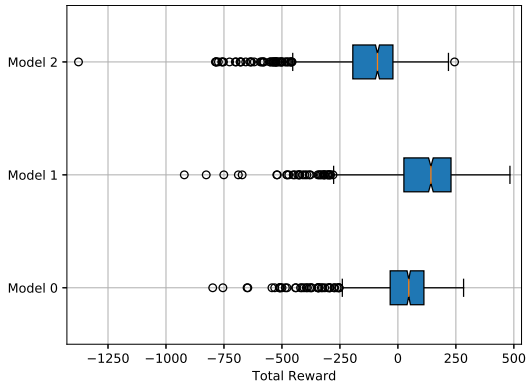

**(a)** Baseline FSM eCDF of $\bar{U}$




**(b)** Baseline FSM box plot of $\bar{U}$




**(c)** AT&E FSM eCDF of $\bar{U}$




**(d)** AT&E FSM box plot of $\bar{U}$

**Figure 4.14.12:** Baseline (no local interactions) and AT&E (local interactions) swarm foraging results for time varying ambiguity.

simulation trial of the swarm agents shifting their focus to the other food cluster at which they now have the higher grab probability is shown in Fig. 4.14.13.

Before drawing conclusions about the swarm foraging implementation of AT&E, there is another case study to investigate. The swarm agents thus far have all been identical or *homogeneous*. Given that they implement stochastic policies, the agents do not choose identical actions, but they are affected by the ambiguities of the problem in the same way. Some additional swarm emergent behaviors, resulting in resilience to ambiguity can be observed in swarms with *diversity* in the way they are affected by and respond to ambiguity.

| | Model 0 | Model 1 | Model 2 |
|---|---|---|---|
| Max | 21.11% | 45.46% | 79.69% |
| Upper Quartile | 79.18% | 120.62% | 85.76% |
| Mean | 256.34% | 820.45% | 48.90% |
| Median | 898.67% | 289.90% | 60.34% |
| Lower Quartile | 51.52% | 163.79% | 38.46% |
| Min | 11.58% | 17.93% | -30.64% |
| Std Dev | 15.36% | 19.63% | 12.44% |

**Table 4.14.9:** Percent change of major statistics between the baseline FSM and AT&E FSM solution for time varying ambiguity.

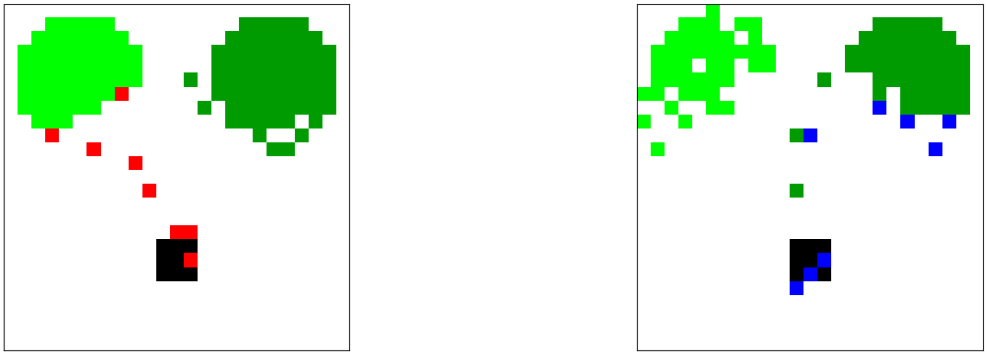


**Figure 4.14.13:** Example of AT&E overcoming time varying ambiguity. The figure on the left is before the robots' heading changes and they are focusing on the north west cluster, which at this time they have the highest grab probability and is where they are informing each other of the most success. After the heading change, however (indicated by the robots' color changing to blue) they begin experiencing failures at the north west cluster and after some brief exploration, shift their focus to the north east cluster, at which they discover they now have the higher grab probability.

## 4.15 Swarms with Diversity

As seen in the case study above in Section 4.14, swarms with the ability to exchange information through local interactions tend to be more resilient to ambiguity than the baseline solution. The swarms discussed so far, however, are *homogeneous swarms*, meaning all of the agents in the swarms share the same models, the same decision-making strategy, and are considered equally capable of performing all of the same tasks. It is possible, however, that different swarm agents may be designed such that they use different models and different decision-making strategies and that some agents may be better at performing certain aspects of their task than others. This is the concept of a *heterogeneous swarm* which is explored briefly here.

It should be noted that many discussions of swarms in existing research use homogeneity as a defining characteristic of a swarm. In other words, the fact that all agents are identical is what makes them a swarm. The author relaxes this definition for the purposes of this research, however, in order to better explore the objectives of this research, which are focused on resilience to ambiguity. The precise definition of what is or is not a robotic swarm (or additionally, a multi-agent system vs. a swarm) is not of importance here.

Swarms may be diverse in many ways. Different agents may use different models, they may have different objective functions, or they may have different properties which result in different decision-making strategies, even when using the same models and objective functions. One possible outcome of this diversity is that certain swarm agents may be more effective at performing their mission in a certain way and other agents may be more effective in a different way. This *specialization* of different swarm agents may be purposefully intended, so that different agents focus on different parts of the mission. While this is a very useful property of diverse swarms, what is of particular interest in this research is how unintended specialization may emerge when diverse swarms are faced with ambiguity. In addition, it is possible that a swarm may be intended to be homogeneous, but a source of ambiguity may be that sources of uncertainty do not affect all agents in the same way. This results in an implicitly heterogeneous swarm, that was intended to be homogeneous. If the *implicitly heterogeneous* swarm is resilient to ambiguity, however, specialization can still emerge. This case of implicitly heterogeneous swarms is of particular interest, because the case of swarm agents not knowing in which ways ambiguity may affect them differently is a very likely scenario when faced with real-world ambiguities. Developing emergent specialization to overcome

these challenges would be an interesting desirable outcome of this research. Regardless of the cause of the diversity, however, *emergent specialization* is the desired outcome of the following case study.

## 4.16 Case Study: Simulated Robotic Swarm Foraging with Diversity

The same simulated swarm foraging problem as described in Section 4.13 and implemented in Section 4.14 is implemented in this case study, but with a few specific additions. A total of 12 agents exist in the swarm for these simulations, but now two subgroups exist within the swarm. All these subgroups know is that they have a different *personality type* from other subgroups, but they do not directly know how this difference in personality changes their ability to perform the foraging task. This configuration of the swarm foraging grid world problem with swarm agents of two different personality types is illustrated in Fig. 4.16.1.

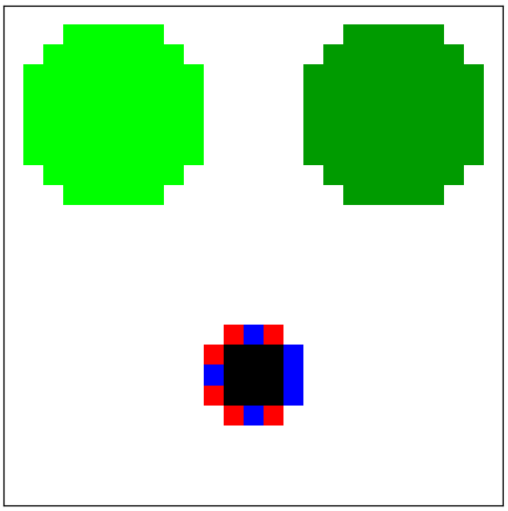
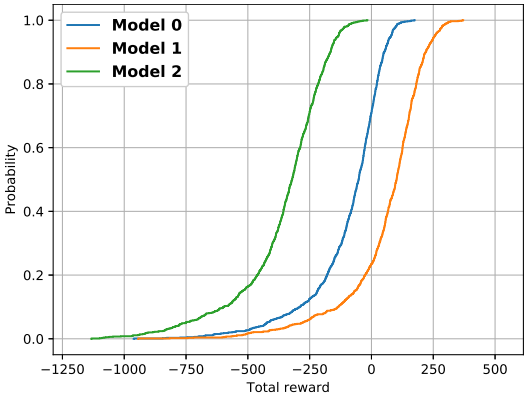


**Figure 4.16.1:** Grid world configuration used for the diverse swarm foraging case study. The red grids represent the robots of personality type 1, the blue grids robots of personality type 2, the black grids the home region, and the green grids are food. Different shades of green food represent different food headings.

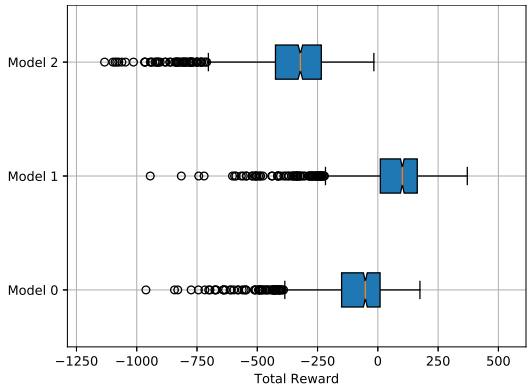
This personality type is an additional piece of information that swarm agents communicate with each other during local interactions and is used to determine how the information exchanged through

local interactions is utilized. The true situation is that robots with different personality have different hidden heading states. The red robots have a heading of east, as all robots have in all previous foraging case studies. The blue robots, however, have a heading of north. The result of this is that the robots whose heading aligns with the north-east cluster will be more successful if they attempt to pick up food from the north-east cluster and vice versa for the robots whose heading aligns with the north-west cluster. The robots are not aware of these differences in their properties, however. All they know is that they are "similar" to robots that have the same personality type as them, and "dissimilar" to any others.

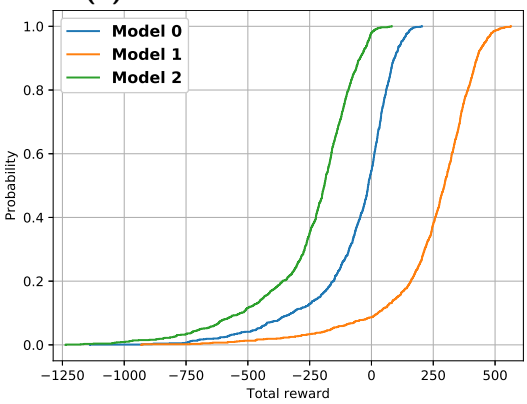
The difference in the decision-making strategy in this case study, compared to the previous one, is that robots only influence their decision-making with information from local interactions with other agents that are of the *same personality*. When they receive information from local interactions from dissimilar agents, they do not let this information influence their decision-making. The motivation for leveraging this knowledge of personality similarity to influence actions is inspired by models of social interactions between animals. Animals that behave collectively are more likely to imitate the behavior of other similar animals and less likely to imitate that of different animals. For example, if herds of both sheep and cattle are located in the same pen, the sheep are more likely to follow each other to find food than they are to follow any of the cattle, and vice versa [116]. The expected outcome for the robots in this simulated foraging problem, however, is that their local interactions will result in emergent behaviors where the robots of one personality type will focus more on attempting to grab food from the clusters of food where they have a high probability of success and the other personality type will focus on other clusters, where they have a higher probability of success. The agents do not understand why they are more successful at one cluster than another, but the information gained through physical feedback with the environment and communicated to other "similar" agents results in an *emergent specialization* that overcomes the ambiguity with which the swarm agents are presented. The results are presented in Fig. 4.16.2 and Tab. 4.16.1.



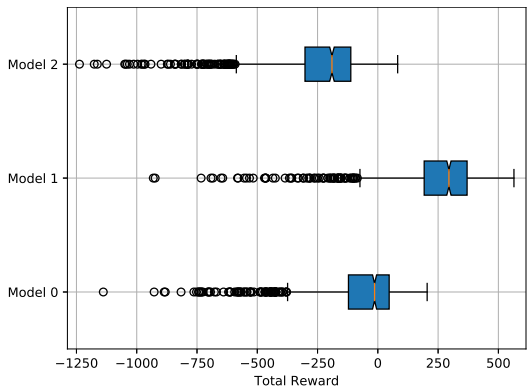

**(a)** Baseline FSM eCDF of $\bar{U}$




**(b)** Baseline FSM box plot of $\bar{U}$




**(c)** AT&E FSM eCDF of $\bar{U}$




**(d)** AT&E FSM box plot of $\bar{U}$

**Figure 4.16.2:** Baseline (no local interactions) and AT&E (local interactions) swarm foraging results for diverse swarms.

| | Model 0 | Model 1 | Model 2 |
|---|---|---|---|
| Max | 17.13% | 52.14% | 598.48% |
| Upper Quartile | 406.05% | 126.43% | 52.20% |
| Mean | 26.85% | 302.21% | 31.59% |
| Median | 73.91% | 190.12% | 40.79% |
| Lower Quartile | 18.92% | 1771.81% | 28.85% |
| Min | -18.39% | 1.33% | -9.20% |
| Std Dev | 18.24% | 18.53% | 12.75% |

**Table 4.16.1:** Percent change of major statistics between the baseline FSM and AT&E FSM solution for swarms with diversity.

As expected, the results show that diverse swarms utilizing AT&E demonstrate resilience to ambiguity and can develop emergent specialization. This can be seen in an example from one of the simulation trials, shown in Fig. 4.16.3, where the red agents focus on the one cluster at which they have the highest probability and the blue agents focus on the other cluster.

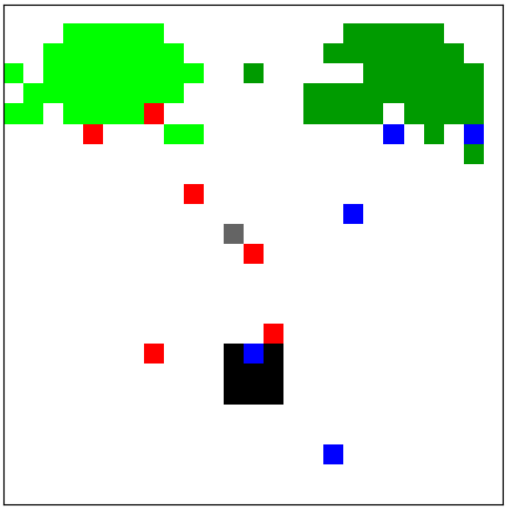
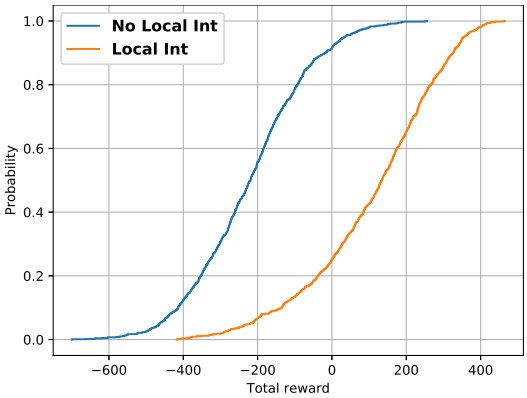
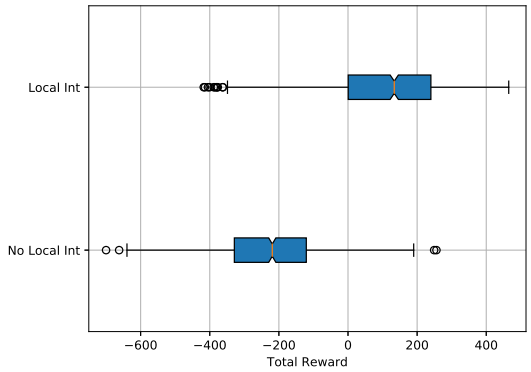


**Figure 4.16.3:** Example of emergent specialization within a diverse swarm, with different types of agents focusing on the food clusters for which they find they have a higher probability of success. The one gray grid is a robot whose battery ran out of power before it arrived back home.

In order to further verify the performance of diverse swarms, this scenario was also tested on the alternate map configuration of Model 3. The results from these trials are shown in Fig. 4.16.4 and Tab. 4.16.2. These results show the same trend as before.




**(a)** eCDF of $\bar{U}$




**(b)** Box plot of $\bar{U}$

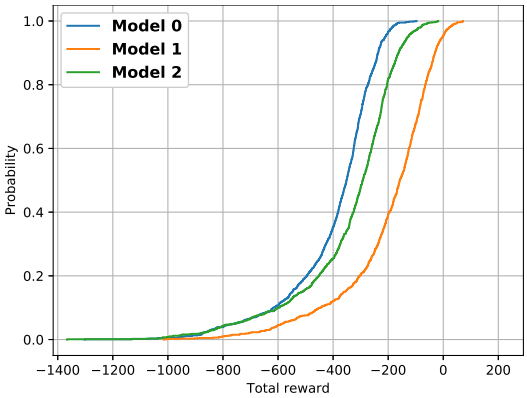
**Figure 4.16.4:** Baseline (no local interactions) and AT&E (local interactions) swarm foraging results for diverse swarms on a different map configuration.

| | |
|---|---|
| Max | 81.73% |
| Upper Quartile | 298.77% |
| Mean | 149.65% |
| Median | 161.08% |
| Lower Quartile | 100.19% |
| Min | 40.36% |
| Std Dev | 15.90% |

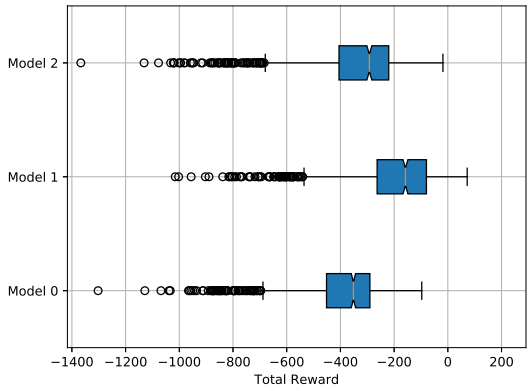
**Table 4.16.2:** Percent change of major statistics between the baseline FSM and AT&E FSM solution for diverse swarms on a different map configuration.

Additionally, the diverse swarm is tested in the same time varying ambiguity scenario as presented in Section 4.5. These results are presented in Fig. 4.16.5 and Tab. 4.16.3.

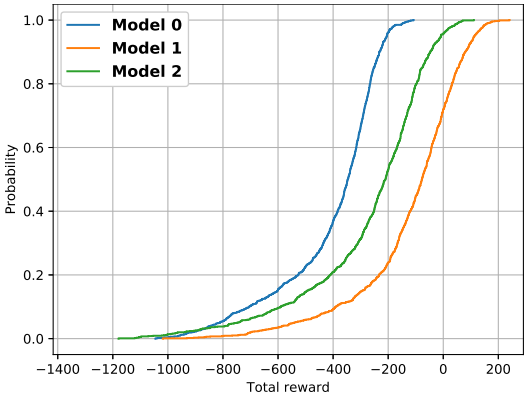
As expected, the results show that diverse swarms utilizing AT&E demonstrate resilience to ambiguity and can develop emergent specialization even with time varying ambiguity. And again, in order to further verify the performance of diverse swarms with time varying ambiguity, this scenario was also tested on the alternate map configuration of Model 3. The results from these trials are shown in Fig. 4.16.6 and Tab. 4.16.4. These results show the same trend as before.



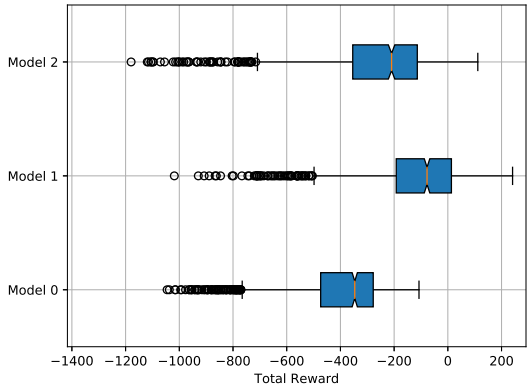

**(a)** Baseline FSM eCDF of $\bar{U}$




**(b)** Baseline FSM box plot of $\bar{U}$




**(c)** AT&E FSM eCDF of $\bar{U}$



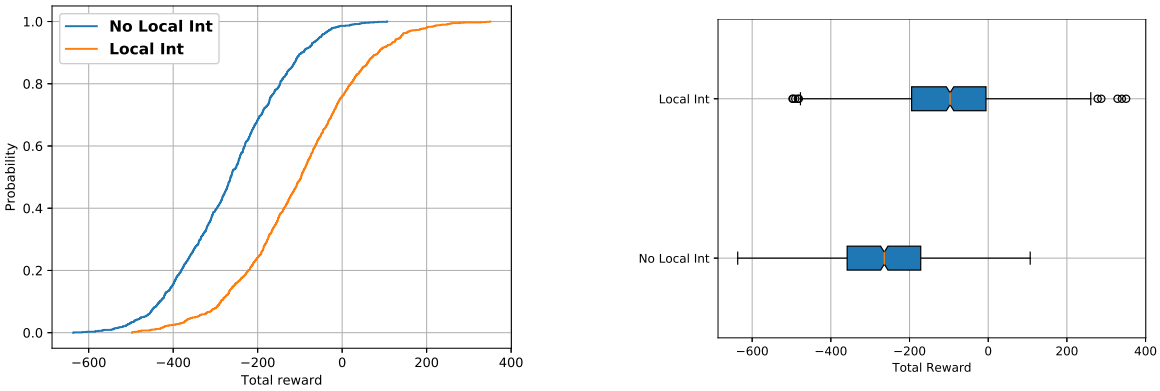

**(d)** AT&E FSM box plot of $\bar{U}$

**Figure 4.16.5:** Baseline (no local interactions) and AT&E (local interactions) swarm foraging results for diverse swarms with time varying ambiguity.

Again, as expected, the diverse swarms utilizing AT&E demonstrate resilience to ambiguity, even when the ambiguity is time varying. Some overall conclusions about all of the swarm foraging case studies examined thus far are presented next.

| | Model 0 | Model 1 | Model 2 |
|---|---|---|---|
| Max | -10.64% | 234.45% | 711.42% |
| Upper Quartile | 4.27% | 116.73% | 48.34% |
| Mean | -2.46% | 40.19% | 21.25% |
| Median | 1.41% | 50.92% | 28.34% |
| Lower Quartile | -4.82% | 27.04% | 12.61% |
| Min | 19.73% | -0.34% | -13.70% |
| Std Dev | 12.47% | 11.37% | 19.88% |

**Table 4.16.3:** Percent change of major statistics between the baseline FSM and AT&E FSM solution for swarms with diversity and time varying ambiguity.




**(a)** eCDF of $\bar{U}$

**(b)** Box plot of $\bar{U}$

**Figure 4.16.6:** Baseline (no local interactions) and AT&E (local interactions) swarm foraging results for diverse swarms on a different map configuration with time varying ambiguity.

| | |
|---|---|
| Max | 228.42% |
| Upper Quartile | 96.70% |
| Mean | 61.64% |
| Median | 63.29% |
| Lower Quartile | 45.69% |
| Min | 21.87% |
| Std Dev | 12.81% |

**Table 4.16.4:** Percent change of major statistics between the baseline FSM and AT&E FSM solution for diverse swarms on a different map configuration with time varying ambiguity.

## 4.17 Simulated Swarm Foraging AT&E Conclusions

Overall, it can be seen that the swarm foraging implementation of the AT&E framework tends to provide the intended benefit of resilience to ambiguity, compared to a baseline FSM solution of distributed swarm agents that do not communicate information to each other through local interactions. While there are many limitations to this method, the most significant of which is that a fair amount of prior knowledge is still required (e.g., knowledge of the space of parameters governing the ambiguity), this method does, however, show promise as an initial starting point for further investigations into this line of research. This framework is significantly less computationally intensive and generally requires less modeling and accurate prior knowledge than most other methods that would be applied to these types of problems, however. These benefits make this framework (and future derivatives of it) much more deployable onto systems working in complex real-world situations. Also, as demonstrated through the time varying scenarios, the AT&E framework is able to overcome time varying ambiguity without being "stuck" converged to a solution that was "learned" with repeated evidence, prior to experiencing a sudden change. Many other competing frameworks, such as reinforcement learning, would likely not be able to "unlearn" the previous configuration of the ambiguity as quickly, at least not without special prior knowledge provided to the framework that such a task might be necessary, if certain hard-coded conditions arise. No additional provisions are necessary for AT&E to handle this situation, however. The same, unmodified framework is used in both the time invariant case studies as well as the time varying ones. No explicit prior knowledge about the time varying nature of the problem is provided.

One possible drawback of the AT&E framework, as presented thus far, is that its primary benefit comes from being deployed onto a distributed robotic swarm. Some problems are better suited to single agents. This raises the question of whether a variation of AT&E can be applied to single agent decision-making that retains the resilience to ambiguity properties demonstrated on swarms. Considerations for this are discussed next.

## 4.18 Transferring Swarm Resilience to Single Agents

While robotic swarms can indeed be more resilient to ambiguity, as shown previously, swarms are not always applicable to particular problems. First, swarms are more complex, because there

are now many robots, instead of just one. Because of this, they can also be cost prohibitive and deploying a swarm is generally more expensive than a single robot. And finally, swarms simply take up more physical space than a single robot and some environments may be too small to support swarms. For many reasons, single robots may be preferred over swarms for many applications. Therefore, it is desirable to investigate whether the underlying properties of swarms that makes them resilient to ambiguity can be integrated into single robot decision-making frameworks.

Fundamentally, what is theorized to make the swarm decision-making based on the AT&E framework presented thus far more resilient to ambiguity is the exchange of multiple sources of *physical feedback*, based on *diverse* experience and decision-making strategies, through *local interactions* of different swarm agents, resulting in *emergent behaviors* of the swarm as a whole, that overcome the ambiguities they may encounter. A single agent on the other hand can only attain physical feedback through its own interactions with the environment, the diversity is limited to what the agent itself has experienced and how it chooses to make decisions, and there are no other agents through which to gain additional information through local interactions. The question then becomes: how can analogous factors be developed for single agent decision-making, that does not rely on other agents in a swarm? This is a challenging question and developing a general case that handles all of these factors is a significant research task. Therefore, incremental steps must be made on an achievable subset of this objective. One proposed strategy to begin investigating this concept, however, is for a robot to leverage its own prior experience through a "virtual" swarm-like structure of exchanging information from past experiences. A proposed methodology for doing this is presented next.

## 4.19 Proposed Method: Groundhog Day: Utilizing Prior Experience

In the swarm case studies explored thus far, *physical feedback* from different agents gaining *diverse* experience is communicated to different agents via *local interactions*. Utilizing the AT&E framework, this information is used by each agent to influence its individual decision-making to attempt to better solve the task and overcome ambiguity. For single agents, however, there are no other agents with which to exchange information and the possible benefits of these factors, in terms of resilience to ambiguity, cannot be directly applied to a single agent. While a single agent may not be part of a swarm, it can however, assume that it is part of a *virtual swarm*. The virtual swarm exists

entirely in the single agent's decision-making software and the one true single agent simulates its interaction with virtual swarm agents. The challenge is to determine what information drives the behavior of the virtual agents and how they could be useful if the one real agent is the only one able to interact with the true environment and receive physical feedback.

The method proposed here is to use the single agent's own *prior experience* at performing previous iterations of the same decision-making task as the information that drives the virtual agents. Specifically, each virtual agent is a recording of a *past version* of the single agent performing the same decision-making task. The name of this method, Groundhog Day, is inspired by the 1993 movie of the same name [117], in which the main character is stuck in a loop of experiencing the same day and the same events repeatedly, but has full memory of each prior version of the day, helping him to improve his future decisions. This aspect of the movie is very similar to the proposed Groundhog Day single agent decision-making method, because the agent will repeat the same scenario many times, each time experiencing different outcomes due to the stochastic nature of the decision-making problem and choosing to take different, *diverse* actions. In order for this to be effective, diversity in the agent's decision-making is hypothesized to be key. If the agent simply repeats the same set of actions each time, little new information is gathered about the decision-making problem during each iteration and the benefit of this method is lost. The Groundhog Day method is tested in the same simulated robotic foraging problem presented in Section 4.13.

In the context of this foraging problem, once the prior iterations have been recorded, however, Groundhog Day operates very similarly to the swarm foraging decision-making strategy with local interactions, presented previously in Section 4.14. Functionally, Groundhog Day uses the exact same framework of AT&E used in the swarm case studies, as described in Alg. 4.5. The only difference is the information communicated from "other swarm agents" comes from replaying the single agents own past experience. As the one true single agent takes actions to carry out the task, it "simulates" the virtual agents, operating in the same environment, by playing back a recording of each of their states and actions as they carried out the same task. Should the true single agent come within communication range of one of the virtual agents, it exchanges information with that agent via a local interaction, as discussed previously in Section 4.14. The virtual agent is simply a playback of a recording, so its behavior will not be influenced by the true agent, but the true agent's behavior will be influenced. The idea is that the true agent will gain diverse knowledge from the physical feedback experienced by past versions of itself, through virtual local interactions with these past versions of

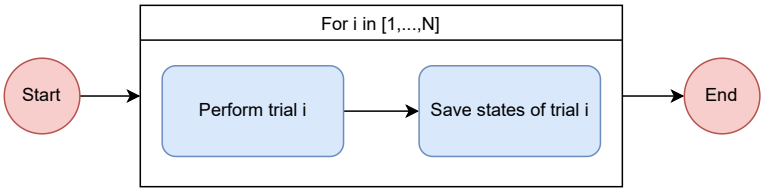
itself. This is somewhat analogous to the use of experience replay in reinforcement learning, where past experiences are remembered and are presented again during the learning process [118].

Given that the agent has full knowledge of the entire history of each previous trial of solving the same decision-making problem, one question is: why is this prior knowledge not used instead to "learn" a model of the decision-making problem in an attempt to characterize the ambiguity? The reason is that this type of "convergence" to one model that describes everything is contrary to the objectives of this research. It is not the intent for a set of prior iterations to collect enough diverse information to form a "complete" model (this is assumed to be a nearly impossible task for complex, real-world problems), nor are the decision-making strategies presented in this research intended to "reason" over models assumed to be complete and accurate. Even if such a model could be "learned" through these prior iterations, it is challenging to formulate the structure of the model in a way that retains the beneficial properties of the history and diversity of different trials. Even with such a formulation, it is likely to be computationally challenging to update and maintain such a model, given the spatio-temporally varying nature of the information that is being captured. Thus, while the virtual swarm approach of Groundhog Day may be "simplistic" and fails to consider some types of prior knowledge that could be leveraged to improve the performance, it represents a "good enough" solution that can be applied to complex problems to further explore resilience to ambiguity, without significant computational challenges. The proposed Groundhog Day method is expected to have many limitations and is not intended to be a highly-developed, ready to deploy, decision-making strategy for complex real-world decision-making problems for a single robot. It is simply an incremental step in investigating a different paradigm of decision-making strategies that attempt to overcome some of the limitations of existing methods. The next section presents results from Groundhog Day simulations and compares its performance to a single agent implementation of the foraging AT&E FSM.

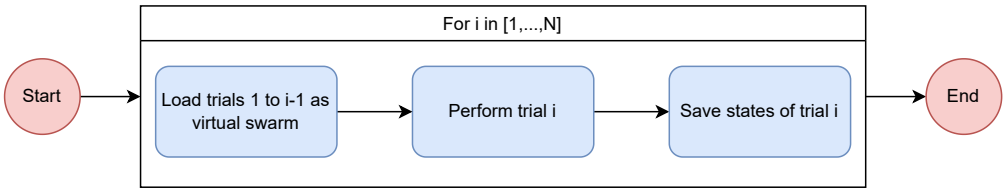
## 4.20 Case Study: Simulated Groundhog Day Robotic Foraging

The Groundhog Day foraging AT&E solution uses the same decision-making implementation as was described for the swarm case study in Section 4.14.1. Similarly, the same foraging grid world configurations described in Section 4.14.2 are used to test the single agent decision-making methods. The difference is there is now only one (real) agent in the environment at a given time. Sim-

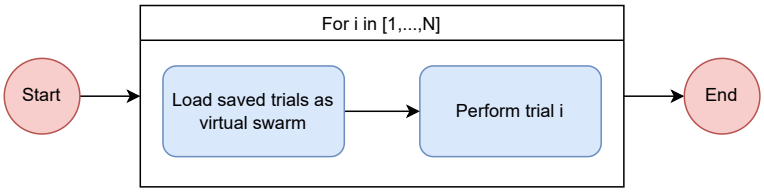
ulations are performed in each grid world model for a true single agent scenario as a baseline and for two variations of the Groundhog Day method. The first variation is called *Groundhog Day Individual*, meaning that each of the prior trials that are played back as the virtual agents are completely independent single agent trials of the problem. The second variation is called *Groundhog Day Recursive*, meaning that each successive training trial plays back each of the previous trials before it and incorporates their experience recursively into outcomes of that trial. In other words, the first training trial is a lone single agent, but the second training trial consists of the one real agent and one virtual agent, which is the playback of the first trial. Subsequently, the third trial consists of the one real agent and the replay of the first and second trials, and so on. This recursive training method better approximates the behavior of a real swarm, since the other virtual agents are also influenced by the information from other agents and are likely to find better ways to overcome the ambiguity they are facing. The way information from previous trials is recorded and used in both the individual and recursive methods, compared to how the evaluation trials are performed after the training trials have been recorded is illustrated in the diagrams in Fig. 4.20.1.




**(a)** Groundhog Day Individual Training



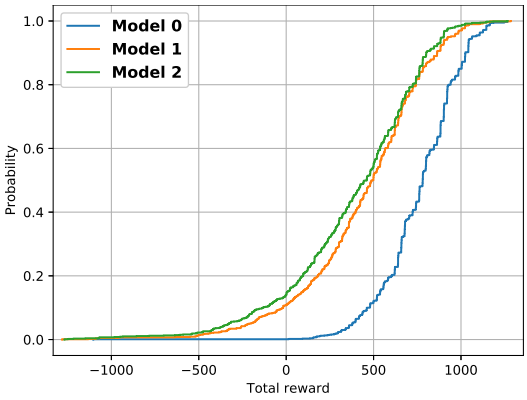

**(b)** Groundhog Day Recursive Training



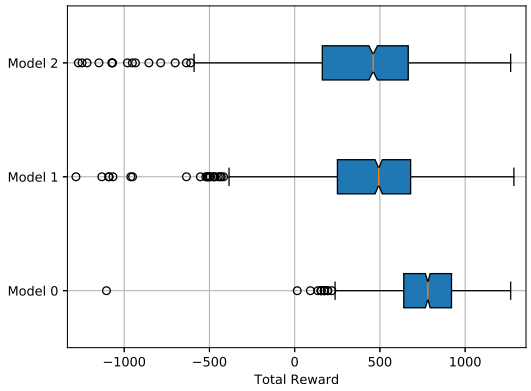

**(c)** Groundhog Day Evaluation

**Figure 4.20.1:** Illustrations of how training trials are recorded and used for both the Groundhog Day Individual and Groundhog Day Recursive methods, as well as how the evaluation trials are performed for both cases.

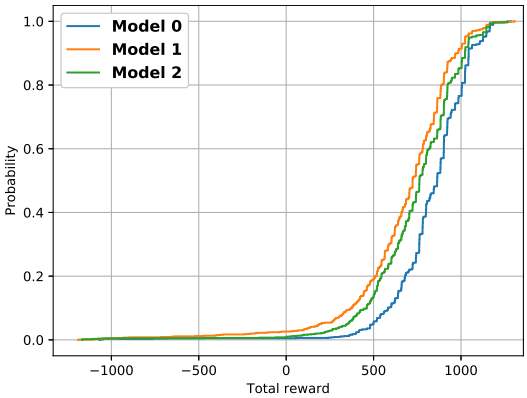
Both the individual and recursive methods are tested for a varying number of training iterations, which is analogous to a different number of swarm agents from the swarm case study, presented previously. As was done for the swarm case study, scenarios of 4, 8, and 12 training trials (i.e., the number of virtual swarm agents) are tested. The results are then evaluated over 1000 Monte Carlo trials, for each scenario, as was done in the previous case study. The baseline result is a single agent implementing the baseline FSM solution presented in the previous case study. The Groundhog Day Individual and Recursive solutions are then compared to this single agent baseline. The results from simulation, with the median number of 8 prior trials, of both the Groundhog Day Individual and Groundhog Day Recursive solutions, compared to the single agent baseline, are presented in Fig. 4.20.2, Tab. 4.20.1, and Tab. 4.20.2.



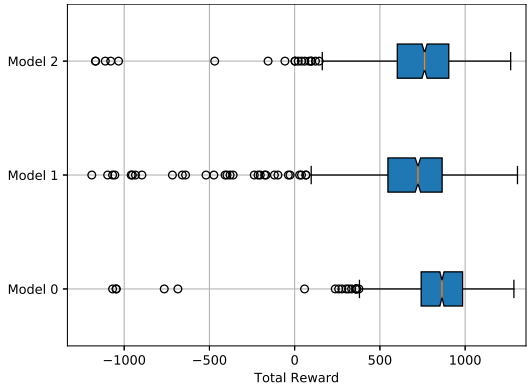

**(a)** Baseline Single Agent eCDF of $U$



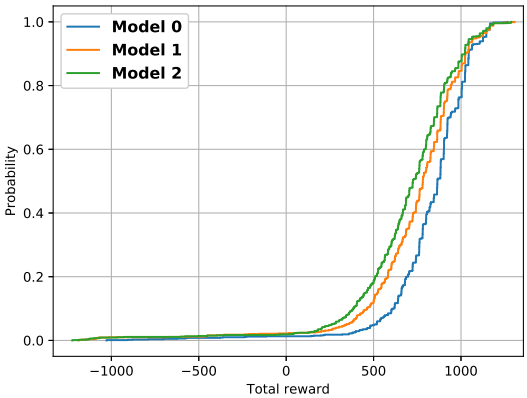

**(b)** Baseline Single Agent box plot of $U$



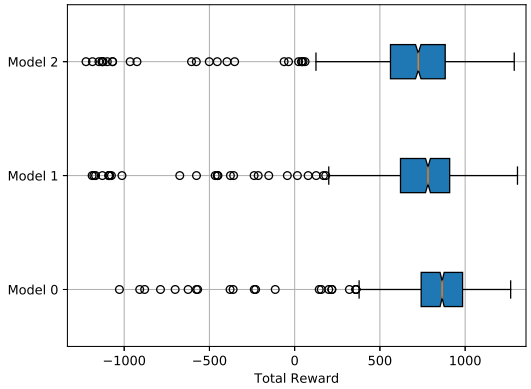

**(c)** Groundhog Day Individual eCDF of $U$




**(d)** Groundhog Day Individual box plot of $U$




**(e)** Groundhog Day Recursive eCDF of $U$




**(f)** Groundhog Day Recursive box plot of $U$

**Figure 4.20.2:** Baseline single agent, Groundhog Day Individual, and Groundhog Day Recursive foraging results for varying grab probability ambiguity.

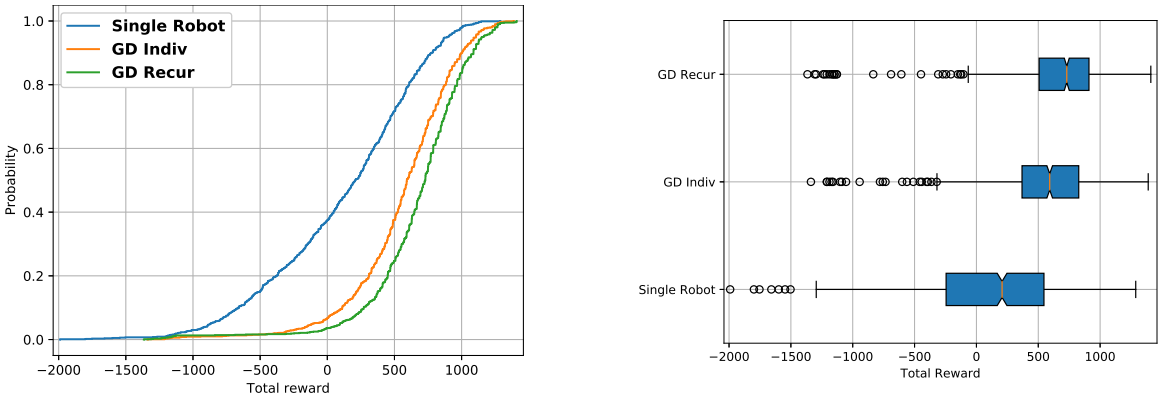
| | Model 0 | Model 1 | Model 2 |
|---|---|---|---|
| Max | 1.50% | 1.63% | 0.00% |
| Upper Quartile | 7.07% | 27.16% | 35.68% |
| Mean | 8.98% | 53.17% | 91.16% |
| Median | 10.63% | 46.65% | 65.47% |
| Lower Quartile | 15.94% | 118.15% | 271.60% |
| Min | 3.26% | 7.25% | 7.96% |
| Std Dev | 0.51% | -15.65% | -31.73% |

**Table 4.20.1:** Percent change of major statistics between the baseline single agent and Groundhog Day Individual solution for varying grab probability ambiguity.

| | Model 0 | Model 1 | Model 2 |
|---|---|---|---|
| Max | 0.00% | 1.63% | 1.66% |
| Upper Quartile | 7.07% | 33.66% | 32.53% |
| Mean | 8.85% | 67.15% | 79.09% |
| Median | 10.76% | 58.62% | 57.44% |
| Lower Quartile | 15.94% | 147.56% | 246.91% |
| Min | 6.88% | 7.40% | 3.55% |
| Std Dev | 8.73% | -14.38% | -18.98% |

**Table 4.20.2:** Percent change of major statistics between the baseline single agent and Groundhog Day Recursive solution for varying grab probability ambiguity.

It can be seen that overall, both Groundhog Day methods provide an increase in performance compared to the single agent baseline. The recursive method generally provides a slight increase in performance, compared to the individual method, but the results are not significant. It should be noted, however, that these results are sensitive to the outcomes of the small number of training trials upon which they are based. Only having access to the same 8 prior trials, for both methods respectively, could result in poor performance in some cases, because it is possible that the majority of the 8 trials (very small number in statistical terms) could have suffered poor outcomes and not provided much benefit to the evaluation runs. A more statistically complete study with more diverse prior trials is considered for future work, and will be discussed further in Chapter 5.




**(a)** eCDF of $U$

**(b)** Box plot of $U$

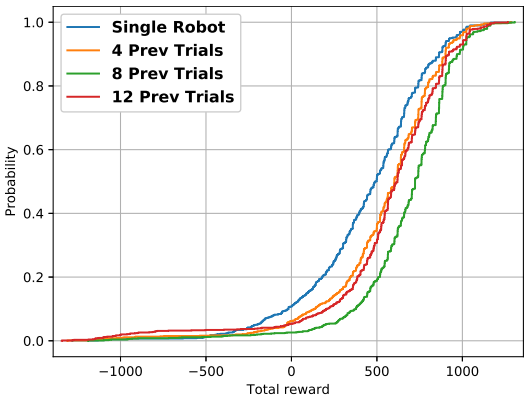
**Figure 4.20.3:** Baseline single agent, Groundhog Day Individual, and Groundhog Day Recursive foraging results for varying grab probability ambiguity on a different map configuration.

As before, both Groundhog Day methods were also tested on a different map configuration (Model 3). The results from these trials are shown in Fig. 4.20.3 and Tab. 4.20.3.

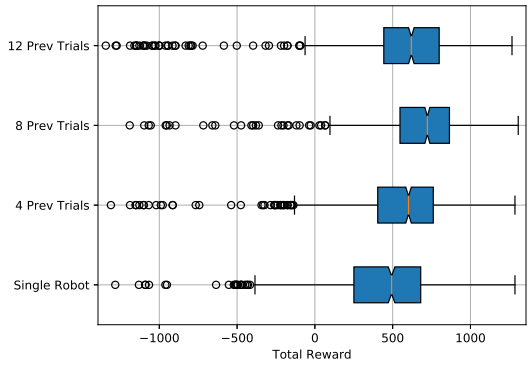
| | Individual | Recursive |
|---|---|---|
| Max | 7.84% | 9.47% |
| Upper Quartile | 51.51% | 66.67% |
| Mean | 367.65% | 460.74% |
| Median | 185.54% | 252.05% |
| Lower Quartile | 250.51% | 306.94% |
| Min | 32.80% | 31.39% |
| Std Dev | -30.30% | -31.86% |

**Table 4.20.3:** Percent change of major statistics between the baseline single agent and Groundhog Day Individual and Recursive solutions for varying grab probability ambiguity on a different map configuration.

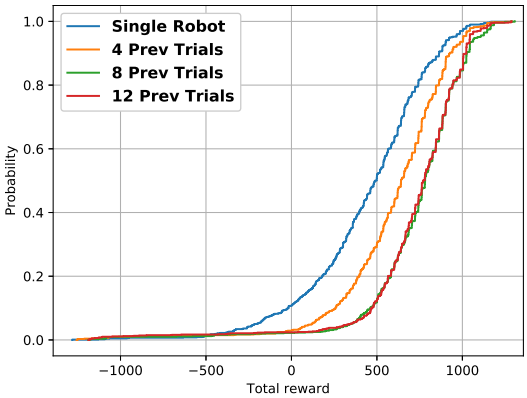
The sensitivity of both the individual and recursive methods to the number of prior trials is also examined. The "median" case of ambiguity Model 1 is used to examine this sensitivity. Results from these trials are shown in Fig. 4.20.4, Tab. 4.20.4, and Tab. 4.20.5.



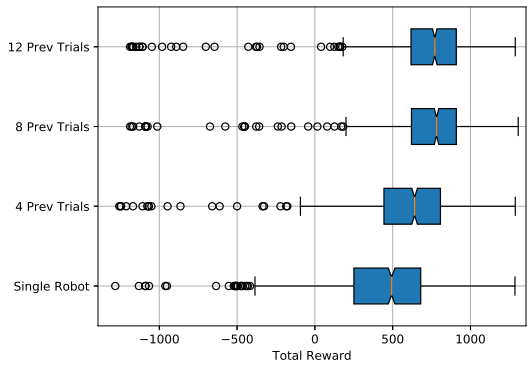

**(a)** Groundhog Day Individual eCDF of $U$




**(b)** Groundhog Day Individual box plot of $U$




**(c)** Groundhog Day Recursive eCDF of $U$




**(d)** Groundhog Day Recursive box plot of $U$

**Figure 4.20.4:** Baseline single agent, Groundhog Day Individual, and Groundhog Day Recursive foraging results for varying number of previous trials.

| | 4 Prev Trials | 8 Prev Trials | 12 Prev Trials |
|---|---|---|---|
| Max | 0.00% | 1.63% | -1.48% |
| Upper Quartile | 11.87% | 27.16% | 17.46% |
| Mean | 22.95% | 53.17% | 26.99% |
| Median | 22.11% | 46.65% | 25.76% |
| Lower Quartile | 61.12% | 118.15% | 76.67% |
| Min | -2.18% | 7.25% | -4.75% |
| Std Dev | -4.13% | -15.65% | 8.77% |

**Table 4.20.4:** Percent change of major statistics between the baseline single agent and Groundhog Day Individual solution for varying number of previous trials.

| | 4 Prev Trials | 8 Prev Trials | 12 Prev Trials |
|---|---|---|---|
| Max | 0.16% | 1.63% | 0.16% |
| Upper Quartile | 18.63% | 33.66% | 33.63% |
| Mean | 36.09% | 67.15% | 64.64% |
| Median | 30.02% | 58.62% | 56.29% |
| Lower Quartile | 77.07% | 147.56% | 146.36% |
| Min | -2.03% | 7.40% | 7.40% |
| Std Dev | -8.74% | -14.38% | -10.30% |

**Table 4.20.5:** Percent change of major statistics between the baseline single agent and Groundhog Day Recursive solution for varying number of previous trials.

As expected, increasing the number of prior trials, and therefore the number of local interactions the real agent has with the virtual agents, tends to increase the performance of the single agent. The recursive method also shows a slight increase in performance, compared to the individual method as well. Some final discussions about the simulation case studies as a whole are presented next.

## 4.21 Simulation Discussions and Key Take-Aways

Overall, the proposed methods, implemented through the AT&E framework do tend to provide added resilience to ambiguity, compared to the established baseline methods. The primary mechanism that enables AT&E's resilience is based on the three key factors that are central to the concepts of this research. These are: the gathering of *diverse* information through *physical feedback*, and then propagating this information through *local interactions*. In the case of swarms, these local interactions occur between other swarm agents and in the case of single agent decision-making with the Groundhog Day method, they occur between the real single agent and virtual agents that are replayed prior memories of itself performing the same task. While providing promising results as exploratory concepts, as shown in the previous case studies, both the swarm and single agent Groundhog Day implementations of AT&E still have many limitations and required assumptions. First, they are limited to problems where a swarm of agents attempting to perform the same task is applicable, or to single agent scenarios where multiple previous trials of the same problem with the same environment configuration can be performed. These case studies also assumed that the

robots do not have any localization uncertainty and that they operate in a discrete grid world environment. Additionally, the AT&E framework must be provided with prior knowledge of the space of ambiguity parameters involved in the problem and how different values of these parameters affect distributions of uncertainty involved in the problem. The innovation of AT&E is that there is no need to estimate the precise values of these parameters with a traditional "top-down" estimation framework. What is important is finding decision-making outcomes that lead to better performance at the objective, when faced with ambiguity. This is also not an optimal method, as it often experiences failures and even when experiencing success, it will sometimes choose to explore less successful courses of actions in order to remain resilient to ambiguities that may be non-stationary. It is also important to note that the severity of a lot of these failures can be seen with the long tails on many of the eCDF plots and the large number of low outliers in the box and whisker plots. These low performing outliers are primarily due to the scale of the different values returned by the reward function, however. Primarily, these are caused by the large penalty for a robot exhausting its battery before being able to return home and recharge. This happened to about 1% to 5% of the robots in the trials with 8 or less swarm agents and in 10% to 13% of the robots in the trials with 12 agents. It makes sense that the rate of battery depletion in the trials with larger numbers of agents would be higher because there are more robots in the same environment and they are competing, not just for food to pick up, but also for the availability of the home region in order to recharge their battery. In any case, the reason this causes the low outliers to have such extreme low values is because the penalty for a robot depleting its battery (-1,000) is ten times larger in magnitude than the reward for successfully delivering food home (+100). Therefore, if a robot's battery is depleted, it receives a penalty that is much larger than the rewards it is likely to have accrued thus far. Thus, the "perceived severity" of these low outliers in the Monte Carlo trials is simply due to the definition of the reward metric. If the penalty for battery depletion had a smaller magnitude, the distributions may look much different, at the low performing end.

In the end, the key take-away from the simulation results of both the swarm and single agent Groundhog Day implementations of AT&E is that this appears to be a promising initial framework that can be further developed by subsequent research. The author predicts at least two possible future research directions. In the case of the better availability of prior information about the structure of the problem and ambiguities involved, the decision-making policies and ambiguity parameter estimation methods (performed via trial and error in this research) can be replaced with more

sophisticated methods that may result in more optimal outcomes. Caution must be taken, however, because these problems quickly become computationally intractable due to the *curse of ambiguity*, as discussed before. Mitigating computational challenges that arise from this will likely be a key focus of this research direction. Another future research direction would be to go in somewhat the opposite direction and attempt to reduce the amount of prior knowledge available to the robot about the problem. This would include attempting to relax the assumption that the robot has accurate prior knowledge of the space of ambiguity parameters involved in the problem. This intersects with another area of research known as *problem solving*, which contrasted with *decision-making*, generally means the agent does not know the full definition of the problem [119] and its exploratory actions are aimed at not only attempting to estimate the values of parameters that define the problem, but also to "estimate" the structure of the problem and the existence of new parameters and dynamics that it may not have previously considered. This is an extremely challenging problem in general, but constraining such a problem to finding the "parameters of ambiguity" while assuming other factors are known could be a plausible direction for future research.

Before moving on to the final conclusions of this research, however, one additional case study is performed, considering the initial motivations of this research. That is, to perform a real-world experiment of a similar robot foraging problem, with real-world sources of ambiguity. The simulation case studies examined thus far have purposefully designed sources of ambiguity that are meant to mimic a small subset of real-world situations, but an investigation of how the methods proposed in this research perform in a real-world setting is necessary to better connect this research with its initial motivations: providing resilience to ambiguity in complex real-world situations. The experimental setup and results from a set of real-world experiment examining the Groundhog Day single agent implementation of AT&E is presented next.

## 4.22 Case Study: Real-World Robot Foraging Experiment

### 4.22.1 Experiment Setup

A real-world robot foraging experiment is performed to evaluate the resilience to ambiguity of the Groundhog Day single agent solution presented in Section 4.20. This experiment is configured similarly to the simulation environment, with a holonomic robot that operates in a 2D environ-

ment that must find and retrieve "food" located throughout the environment. The map used in this experiment is 40 by 20 grid cells (fitting the 2:1 aspect ratio of the air hockey table), with two clusters of food located near the north east and north west corners, similar to the simulation maps. A top-down view of the experiment setup is shown in Fig. 4.22.1.

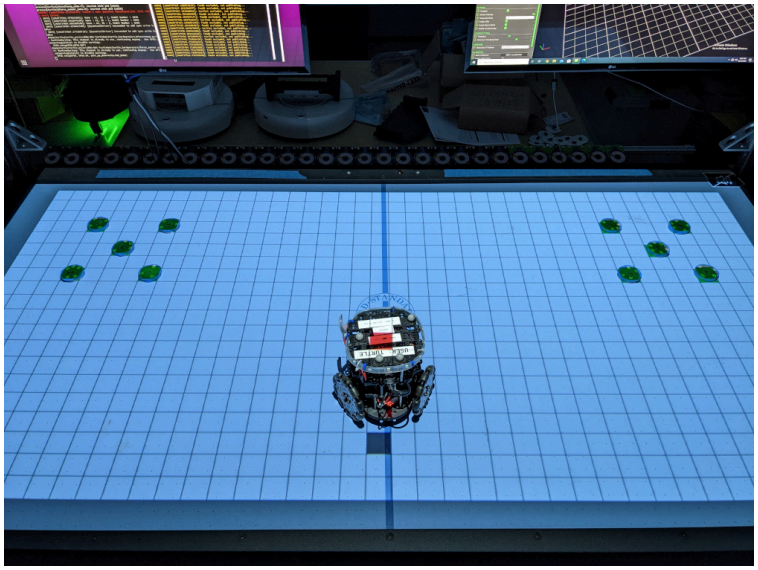


**Figure 4.22.1:** Top-down view of the robotic foraging experiment on the air hockey table testbed.

The robot used in this experiment is a modified 3-wheeled TurtleBot, which implements Kiwi drive, enabling it to move holonomically. The robot has an electromagnet on its underside which it uses to pick up "food pucks", which are made from thin disks of steel sheet metal, with 3D printed bases to raise them up near the height of the electromagnet. The robot, next to one of these food pucks, is pictured in Fig. 4.22.2.

To detect whether it has successfully picked up a food puck, a thin force sensitive resistor is placed over the tip of the electromagnet. This is used to detect the normal force between the electromagnet and the food puck if the robot has successfully picked up a food puck. The underside of the robot, showing the electromagnet and force sensitive resistor, along with a food puck successfully picked up by the robot is shown Fig. 4.22.3

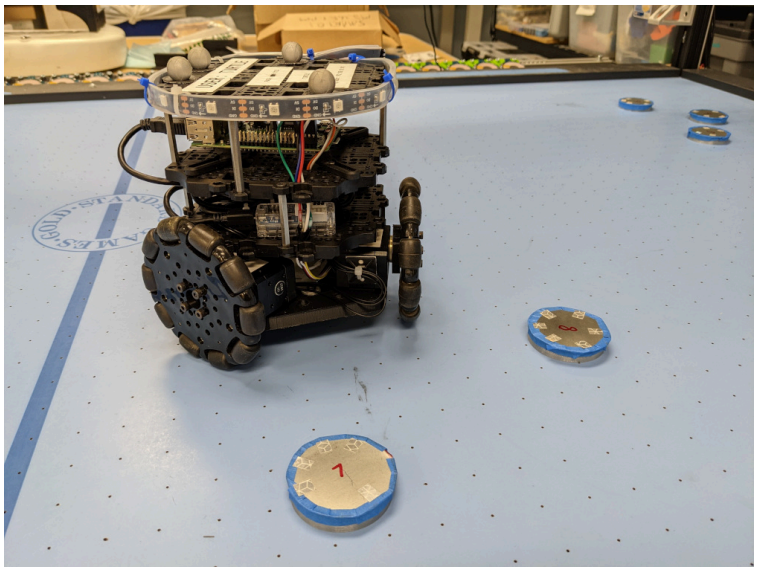


**Figure 4.22.2:** TurtleBot and two of the food pucks used in the foraging experiment.

The robot operates on a flat 2D surface located in an instrumented test environment known as the WVU Interactive Robotics Laboratory (IRL) Air Hockey Table. This testing environment, built on top of an air hockey game table, is equipped with a Vicon multi-camera motion tracking system, used to provide real-time pose feedback for the robot and each of the food pucks. The robot and the food pucks are equipped with retro-reflective markers, enabling the motion tracking system to track their pose in real time. It is assumed for the purposes of this experiment, as was done for the simulation, that the robot has full knowledge of its x,y position in the environment at all times and that it can accurately localize any food that is within its perception range. The motion tracking system is used to implement this functionality. The air hockey table and motion tracking system are shown in Fig. 4.22.4.

The table is equipped with two computers, one for running the Vicon motion tracking system and another for running the robot, over a local WiFi network. The robot carries a Raspberry Pi which runs the drive motors and actuates the electromagnet. Commands are sent to the robot over WiFi from the main computer, which runs the robot decision-making software and other software needed to perform the experiments. Much of the same software used to run the robot foraging simulations is directly used to run the robots, but with different interfaces for driving and receiving

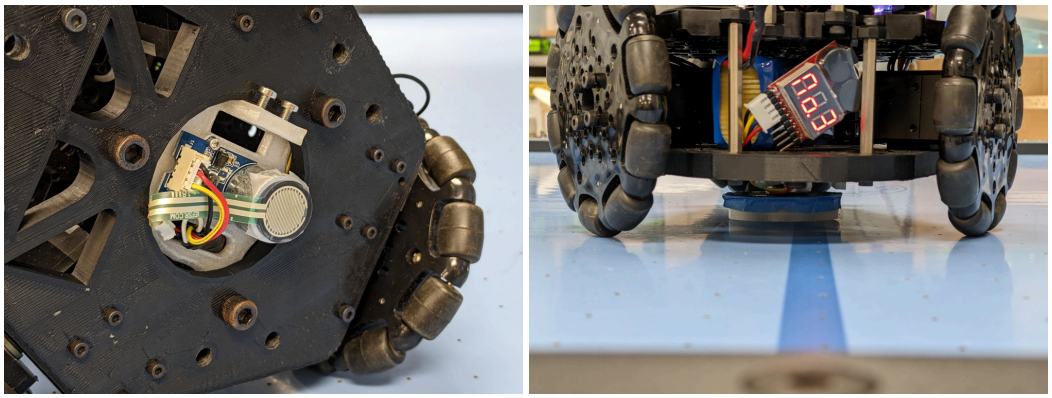


**Figure 4.22.3:** Electromagnet with force sensitive resistor (left) and an example of the robot picking up a food puck with the electromagnet (right).

feedback from the real robot and the Vicon system, instead of from the back-end of the simulation. The air hockey table also has an overhead projector, so that the grid world representation used inside the robot's decision-making software can be visualized in real-time, as experiments are being performed. This is what is displaying the grid world visualization shown previously in Fig. 4.22.1. The only interaction the human has with the experiment while it is running is to remove food pucks that the robot successfully drops off at the home location. This is analogous to the simulation, because any food successfully dropped off at home essentially "dissapears" from the environment and is recorded as food that has been successfully delivered home.

As was mentioned previously in Section 4.14, the physical geometry of the TurtleBot used in this experiment drove the inclusion of the approach direction ambiguity in the simulation. The roughly hexagonal shape of the TurtleBot, with wheels on three of the six sides, will push the food pucks and prevent the robot from picking them up, if the robot approaches the food from one of its sides with wheels. If it approaches from one of the sides with gaps, however, it may be able to successfully pick up the food. This is not the only source of uncertainty (nor ambiguity), however. Even with the high precision of the Vicon motion tracking system, the robot still experiences motion uncertainty 1) due to control errors and 2) due to the fact that its decision-making software operates over a discrete grid world, but the real-world scenario is inherently continuous. The food pucks are purposefully designed to be approximately the size of the robot's motion control error tolerance (2.5cm radius). Due to the motion uncertainty introduced by discretizing the continu-

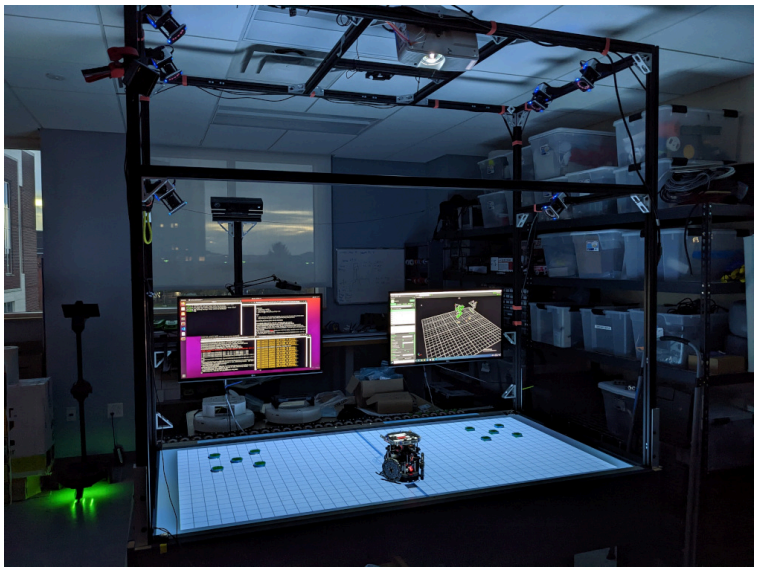


**Figure 4.22.4:** The WVU IRL Air Hockey Table testbed.

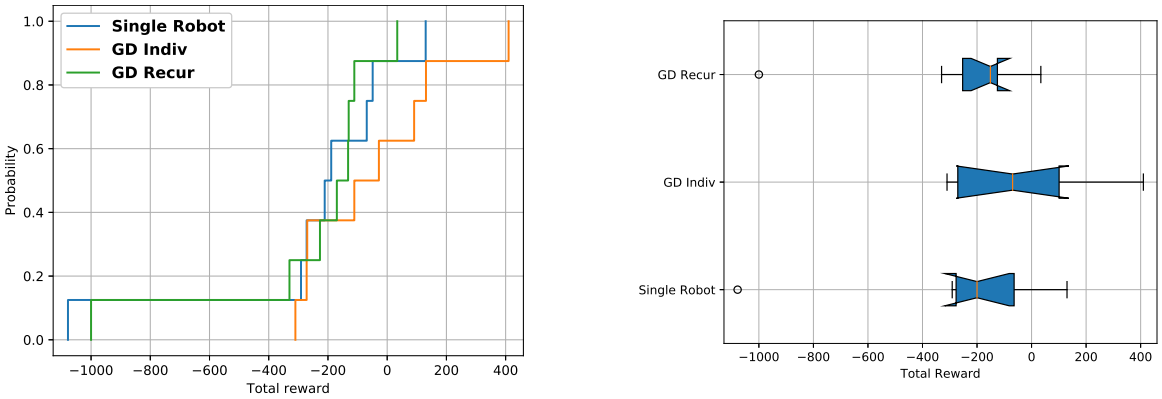
ous real-world into a grid world representation in the robot's decision-making software, however, the robot is not always able to position itself accurately to pick up the food pucks. If the food puck is located near the center of one of the robot's grid cells, it will likely maneuver to the correct location and pick up the food puck. If the food puck has been pushed by the robot's wheels, however, it may be partially between grid cells, and due to only being able to move by discrete grid cell increments, the robot may sometimes fail to align the electromagnet with the food. While not the same type of unmodeled uncertainty as the grab ambiguity due to the food and the robot having an unmodeled "heading" state, as in the simulation, this additional source of ambiguity due to unmodeled food grab uncertainty, resulting from the discretization, is in fact a desired property of the experiment and is purposefully not "designed out" of the experiment by providing the robot with a higher-fidelity model of the environment.

As described earlier, the same decision-making software as used for the Groundhog Day simulations is used for this real-world experiment. The objective and expected outcome of this experiment is the same as the simulation. Due to the practical limitations of running trials of a real-world experiment, compared to the ease of running many computerized simulations, however, the real-world experiment results cannot be evaluated in terms of a distribution over a large number of trials, as was done with the simulation results. The same quantitative metrics as were used to evaluate the sim-

ulations are available, but due to the limited number of trials of the real-world experiments, these results cannot be considered statistically significant. Since there are many uncertainties involved in this experiment and the robot's decision-making policy is also stochastic, the quantitative outcomes of any single trial or small set of trials is not indicative of the performance of the algorithms being tested. Therefore, while the numeric results will be presented the interpretation of the results of these experiments must be done qualitatively. They are analyzed as qualitative extensions of the simulation results, showing that the proposed methods discussed in this research have the potential to be useful frameworks for robot autonomy that is faced with complex real-world ambiguities. The results of the real-world experiments are discussed next.

### 4.22.2 Results and Discussion

Three scenarios of real-world experiments are performed. The first is an isolated single agent scenario, which serves as the baseline, the second implements the Groundhog Day Individual variant, and the third implements the Groundhog Day Recursive variant. For each of these scenarios, eight trials, 200 timesteps long, were performed. Results from these trials, in terms of the total reward $U$ are shown in Fig. 4.22.5 and Tab. 4.22.1. Given the small number of trials, the total reward for each individual trial is also shown in Fig. 4.22.6.



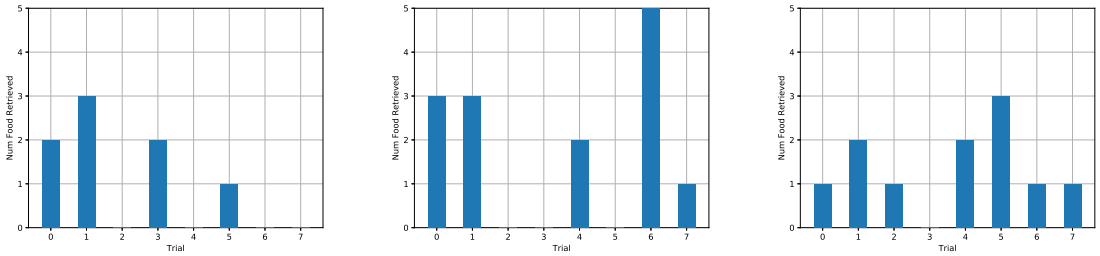

**(a)** eCDF of $U$

**(b)** Box plot of $U$

**Figure 4.22.5:** Baseline single agent, Groundhog Day Individual, and Groundhog Day Recursive results from real-world foraging experiment on the WVU IRL Air Hockey Table.

| | GD Indiv | GD Recur |
|---|---|---|
| Max | 215.38% | -73.85% |
| Upper Quartile | 257.81% | -95.70% |
| Mean | 82.31% | -1.82% |
| Median | 65.25% | 24.50% |
| Lower Quartile | 2.26% | 8.67% |
| Min | 71.24% | 7.24% |
| Std Dev | -31.00% | -12.09% |

**Table 4.22.1:** Percent change of major statistics between the baseline single agent, Groundhog Day Individual, and Groundhog Day Recursive solutions from real-world foraging experiment on the WVU IRL Air Hockey Table.




**(a)** Single Agent Baseline

**(b)** Groundhog Day Individual

**(c)** Groundhog Day Recursive

**Figure 4.22.6:** Baseline single agent, Groundhog Day Individual, and Groundhog Day Recursive number of food collected for each trial from real-world foraging experiment on the WVU IRL Air Hockey Table.

As expected, due to the small number of trials, these results are not statistically significant and no clear trend can be observed from the distribution over the total reward. Looking at just the number of food retrieved alone (without the penalties for failed grabs, distance driven, and depleted battery) shows that both Groundhog Day methods tended to retrieve more food than the single agent baseline. The single agent baseline retrieved 8 total, Groundhog Day Individual retrieved 14 total, and Groundhog Day Recursive retrieved 11 total, summed across all trials. This is still not a statistically significant result, however, and as stated earlier, these experiments must be evaluated qualitatively, as extensions of the results from the simulation trials. Even though not statistically significant, these results do show that the Groundhog Day methods tend to be more resilient to

the real-world ambiguities present in this experiment than a single agent operating without the benefit of AT&E. The most distinct difference observed between the Groundhog Day trials and the baseline method trials is that the Groundhog Day methods tended to choose an approach direction that was on one of the robot's open sides more often than the baseline method, resulting in more frequent successful food pickups. There were many occasions, especially during the trials for both Groundhog Day methods, when the robot successfully picked up a food puck, but due to grabbing it near its edge and not its center, the force sensor used to detect food pickup would report a false negative reading. This led the robot to set the food puck back down, thinking it had failed to pick one up. If these false negatives were not experienced, the number of food retrieved in both Groundhog Day scenarios would have been higher, possibly showing a more clear trend in the distribution of the total reward. The reason for the more frequent false negatives food pickups was due to the robot more frequently choosing a successful approach direction with the Groundhog Day methods, giving the robot more opportunities for successful food pickups. This is not directly observable from the collected data, but this qualitative result does show that the Groundhog Day methods were providing their expected benefit.

Less distinct, but still noticeable was also that the Groundhog Day methods tended to choose search locations that were closer to the real locations of the food on the table. Both of these observations are consistent with the expected behavior of the Groundhog Day AT&E foraging solution. Prior experience communicated to the robot from the virtual agents is used to inform the sampling of the ambiguity parameters $\theta_G$ and $\theta_A$, which help guide the robot's actions towards ones that better solve the task, involving these known ambiguities. These same behaviors were observed in the simulation results as well, so the experimental results are in agreement with those from the simulation.

As explained previously, the mapping from the discrete grid world representation in the robot's software to the continuous real-world caused considerable variance in picking up food successfully, due to motion uncertainty. Also, as expected, there are many other sources of uncertainty that were observed in the real-world experiment that were not modeled in the simulation. The robot would sometimes flip food pucks over with its wheels, thereby obscuring the retro-reflective markers and causing the food pucks to no longer be visible to the motion tracking system. These food would "disappear" from the environment, due to the robot driving into them and flipping them over. The robot would also occasionally push food pucks outside the boundary of the map, also causing them

to disappear from the environment. A wide range of complex, uncertain motion dynamics due to the interaction between the robot's wheels and the food pucks was observed in all of the trials.

Many of the experiment trials for all methods had long periods (or the entire length of the trial, in some cases) of experiencing failures, pushing food pucks around the table with their wheels and just generally performing poorly at the task. This is again, in agreement with the simulation results. The eCDF plots of all of the simulation scenarios showed long tails on the low performance end of the trials, showing that some very poor performances occurred in the simulations as well. This is expected, however, in a problem with not just many sources of uncertainty, but also ambiguity. And again, optimality at solving the task is not the intended outcome of the methods explored through this research. What was successfully demonstrated through the experiments, as was also done through the simulations, is that robots that implement the AT&E framework tend to have better resilience to ambiguity than the established baseline methods. Some final discussion and key take-aways from the experiment results are presented next.

## 4.23 Experiment Discussion and Key Take-Aways

Much of the same conclusions and key take-aways from the simulations, as were previously discussed in Section 4.21, also apply to the real-world experiments. The AT&E framework successfully demonstrates its expected outcomes and shows promise as a starting point for future research into resilience to ambiguity for autonomous robotic decision-making. One additional key take-away from the experiments is to further acknowledge the limitations of methods proposed in this research. The experimental setup provided many additional sources of ambiguity than simulations, however, it was not without some effort of fine tuning parameters that the experiment was brought up to the level of functionality to produce the results presented in this document. The design of the food pucks, for example, required several iterations before an "acceptable" amount of grab uncertainty could be achieved. Too small in diameter and the probability the robot could maneuver to a precise enough location to pick it up was so small that a successful pickup would rarely happen. Too large in diameter and the robot would pick them up too frequently to give the problem enough uncertainty to be challenging. While this is a very specific example, the broader concept is that this level of detail in setting up the environment is still required for an experiment that is intended to reduce the amount of prior knowledge about the distributions of uncertainty involved in the problem.

On the one hand, it is necessary to tune the parameters of problems to be "realistic and representative" of the scientific principles they are intended to investigate. On the other hand, requiring the distributions of uncertainty to be carefully designed for a problem in which the solution methods are intended to work with a lack of knowledge about the distributions of uncertainty may seem a bit contradictory. This field of decision-making under ambiguity, where the decision-maker has incomplete knowledge of the distributions of uncertainty, is an under-explored field, however, especially when applied to robotics and uncertainties arising from the physics of how robots perceive and interact with their environments. Therefore, there is not a large base of existing works upon which to design experiments and evaluation metrics. Most of the concepts behind designing the simulations, experiments, and evaluation metrics presented in this research come from the well established field of robotic decision-making under uncertainty. While decision-making under ambiguity is a fairly straightforward extension of decision-making under uncertainty into a more abstract realm, it does introduce many additional challenges not only in solving the problem, but also in understanding how the results are to be interpreted and quantified. In the end, the key take-away here is that not only is further research needed to find better methods of solving decision-making under ambiguity problems, but that more research is also needed in defining decision-making under ambiguity as a field of study in the context of robotics and in establishing commonly agreed upon descriptions of frameworks, metrics, and evaluation strategies. It is the author's hope that this research helps set the stage for decision-making under ambiguity to become an increasingly studied field in the context of robotics and that future research will work to solve some of the challenges described here in further formalizing this field of study.

# 5

# Concluding Remarks

## 5.1 Chapter Overview

This chapter summarizes the conclusions and key take-aways from the research presented in this document and draws additional conclusions about their broader impacts. Discussions are also presented about future work and how this research can be used as a starting point for further studies into the new concepts that have been explored thus far.

## 5.2 Overall Conclusions and Lessons Learned

As discussed in about the results of the simulations in Section 4.21 and the results of the real-world experiments in Section 4.23, it can be seen that the three factors of *physical feedback*, *diversity*, and *swarm local interactions* can be leveraged to provide *resilience to ambiguity* in decision-making problems that involve ambiguity. Recall that ambiguity is defined in this research as a lack of accurate knowledge about the distributions of uncertainty involved in a stochastic decision-making problem. The three key factors listed above are used to develop a novel decision-making framework known as Ambiguity Trial and Error (AT&E), which is then implemented in several robotic foraging case studies involving ambiguity, both simulated and as real-world experiments. These factors are used to inform the trial and error sampling of new candidate values for the unknown valued *ambiguity parameters* that define the ambiguity involved in the problem. The resilience to ambiguity of this framework comes from gathering information about outcomes that lead to success or failure through *physical feedback* with the environment, based on *diverse* experiences that occur through taking *diverse* exploratory actions, and communicated through *swarm local interactions* to gain a better understanding of what actions are more likely to lead to success, over the space of the problem. The implementations of the AT&E framework in this research are able to provide this resilience without significant computational expense, from which many other existing methods of decision-making under ambiguity would suffer. The low computational cost of these implementations of AT&E come from the use of informed trial and error sampling strategies that are not focused on accurately modeling why outcomes occur the way they do, but instead attempts to guide the selection of actions toward ones that are more likely to provide higher near-term payoffs.

While explored in this research through the context of solutions to a robot foraging problem, the lessons learned through this research can be related to the broader context of robotic decision-

making. One important lesson learned is that resilience to ambiguity requires gathering information about outcomes over a wide variety of conditions and across a wide range of the problem space. Given the existence of ambiguity, however, it cannot be assumed that this information is always accurate, and there must be a way to quickly unlearn "bad" or "outdated" information. This is especially necessary when ambiguity may vary parametrically (e.g., vary with location and/or time). Rather than trying to maintain a globally consistent estimate of the values of the unknown ambiguity parameters, what is more effective in terms of resilience, and less computationally costly, is to be able to quickly "adapt" to the *local* variations in ambiguity. Such globally consistent information may be necessary to achieve what could be considered more "optimal" outcomes, but such an objective for an autonomous robotic system may not be realistically achievable when dealing with complex parametrically varying ambiguity, as is likely to be encountered in many real-world situations. Suboptimal, but more *resilient* solutions are often a more appropriate and achievable objective in these cases. This relates to the concept of *satisficing*, which means that in situations where there is insufficient information or resources to find "optimal" solutions, the objective should not be to remain stuck in the paradigm of needing to find such an optimal solution, but that finding "good enough" solutions is instead the correct objective [120, 121]. This paradigm of "satisficing" instead of "optimizing" is an important foundational concept when making decisions under complex, parametrically varying ambiguity in real-world situations.

Quickly adapting to the local variations in ambiguity is the key challenge, however, and is where novel solutions, such as the ones presented in this research, are required. The gathering and exchange of information about the local ambiguity "conditions" through swarms of agents operating in the same environment or through recordings of prior experience of the same agent performing the same task, in the same environment, is the solution that was found to achieve this through this research. While it is likely there are other methods to achieve this as well, the key point here is that this information must come from actual outcomes of interactions with the true environment. One of the fundamental assumptions here is that the information the agent may have at any given time may be incorrect and it may need to forget that information and learn new information. The use of a "model" or "black box simulator" breaks this assumption and the "predictions" that are provided by such tools are not capable of providing informed estimates of complex, varying ambiguities. Remote sensing of the local environment is also not sufficient at addressing this either, because it cannot provide information about "outcomes," because there is no interaction with the local envi-

ronment. It can instead only provide "observations," which is not sufficient to gather information about the local variations in ambiguity. The use of swarms through the AT&E framework fills this gap which black box simulators and remote sensing cannot, because each swarm agent is another entity that exists within the same true environment and can take actions, outcomes, and then communicate that information to each other. It is acknowledged, however, that the use of swarms can be prohibitive in terms of system complexity and available resources and therefore swarms are not applicable to all problems. This is where the motivation for working to replicate the same properties of swarms with single agents, through the Groundhog Day method came from. Instead of using a swarm of multiple agents, the single agent's own prior experience at performing the same task can be used to provide the same type of "feedback about outcomes" when played back as a "virtual swarm." The class of problems to which the single agent Groundhog Day method is applicable is limited compared to the swarm method, however. The use of prior experience at performing the same task with Groundhog Day is based on the assumption that the ambiguity in the environment is identical at each time step. Spatial variations in ambiguity can be handled, but the ambiguity must be either stationary in terms of time (i.e., temporally invariant) or cyclo-stationary in terms of time, meaning that the temporal variations are periodic and occur cyclically at consistent time intervals. The swarm methods on the other hand are constantly gathering "up-to-date" feedback about outcomes and can be expected to remain resilient to a temporally non-stationary process. In the end, however it is achieved, the lesson here is that feedback about actual outcomes that occur through interaction with the environment, obtained from multiple sources is what is needed to inform the agent about the local variations in ambiguity that can be used to provide resilience in its decision-making strategy.

As mentioned before, however, there are still many limitations to the methods proposed in this work. One of the most important is that knowledge about the space of parameters defining the ambiguity in the problem and the way in which different values of these parameters may affect outcomes is required. While knowing the value of these parameters accurately is not required, needing their definition to be known during the design of the decision-making algorithms is still a challenge that requires significant prior knowledge about the problem to be provided to the decision-making framework. Another limitation is that these frameworks only provide their intended benefit in terms of resilience to ambiguity when deployed onto distributed robotic swarms or onto single agents that can record prior knowledge of outcomes through its own experience at performing the

same task, repeatedly. Some tasks are not suitable for swarms and not all single agent tasks can be "practiced" repeatedly beforehand to gain knowledge from prior experience. Overall, the methods explored through the case studies in this research have worked towards reducing the amount prior knowledge robotic decision-making needs to have about the distributions of uncertainty involved in the problem. However, some knowledge, specifically the definition of the ambiguities and prior experience of outcomes from interacting with the real environment, and not a black box simulator or a predictive model, are required for these methods to provide resilience to ambiguity. Another important limitation in regards to this is the types of problems to which the methods discussed in this research are applicable and those to which they are not. These trial and error based methods, like many other decision-making methods for stochastic problems as well, must perform exploratory actions to gain information about the unknown aspects of the problem. This means that some detrimental outcomes, or partial failures, may occur. The severity of these partial failures that may be found through exploratory actions must be small enough that experiencing one or more of them is not catastrophic to the decision-maker's overall mission objective. In the robot foraging examples presented in this research, occasionally failing to pick up individual food is not a very severe failure, so it is acceptable for the robot to experience occasional failures in this regard and leverage the information from those failures to improve its decision-making in the future. Another hypothetical example application where partial failures are not very severe is a robot delivering mail in an office building. If the robot runs into a closed door and cannot deliver a particular piece of mail at that moment, it is not very detrimental if it has to come back and try again at a later time. In all of these cases, the trial and error based methods presented in this research are applicable, because experiencing a few partial failures is not catastrophic to the mission. However, an example application where these trial and error based methods would not be applicable would be a domain like autonomous driving. Failing to turn or to apply the brakes on a moving vehicle at the correct time could result in the injury or death of people in the vehicle or the surrounding area. Another example application where the failures are severe is motion planning for planetary rovers. If a rover operating on another celestial body gets stuck driving on terrain that was not known to be hazardous, there is likely no way to free the rover and such an event would likely end the rover's mission. These types of failures are catastrophic and unacceptable, so the trial and error based methods presented in this research are not applicable to problems where the severity of failures that may be experienced through exploratory actions is very high. Regardless of these limitations, however, the

case studies investigated in this research have provided great insight into the concept of resilience to ambiguity, but there are a number of directions for future work that build upon the outcomes of this research. These are discussed next.

## 5.3 Future Work

One of the challenges of this research was determining the correct methods to use as the baselines for comparison in the case studies. Decision-making under ambiguity is not currently a widely explored topic so there are few examples of comparable existing methods. Especially lacking are existing methods that operate under the same assumptions about the extent of the lack of knowledge of the distributions of uncertainty used in this research. The few that do tend to suffer from computational complexities and also require large amounts of prior data. Applying them to the simulated foraging problem used in the majority of the case studies, let alone the real-world foraging problem, was not found to be practical for the purposes of this research. However, as a direction for future work, an investigation into how other existing methods that may be applicable as points of comparison should be performed.

One method that has been identified as a candidate for comparison is Deep Reinforcement Learning, or Deep Q-Networks (DQN) [75]. In order to apply a DQN solution method, the definition of the foraging problem investigated in the swarm and Groundhog Day case studies must be in a format that is compatible with such methods. The problem is cast as an RL problem, because due to the ambiguities, it is assumed the state transition model $T$ is not known. The foraging problem used in the initial single agent case studies, where an MDP value iteration solution was applied, assumed the robot had full knowledge of the entire map, including the locations of all food. The updated problem used in the swarm and Groundhog Day case studies removed this assumption, however, and the robot could only perceive the map within a perception range around its current position. While it is assumed the robot can perceive anything in this perception range with certainty (i.e., the perceived locations of the food or other agents are accurate), the fact is that the rest of the environment still exists outside of the robot's perception range and the state of the environment beyond this range affects the outcomes of actions taken over time. This can be regarded as a form of partial observability, and therefore the problem now falls into the category of a POMDP. Some additional aspects of the problem (e.g., the approach direction ambiguity) may

also make the present description of the problem non-Markovian, so additional work may be required to define additional states that can make the problem Markovian. Regardless, assumming the problem can be cast as a POMDP, the dimensionality of the state space is going to be large and will present challenges for finding a solution. This is where the neural network aspect of a DQN becomes beneficial, since these methods are capable of solving problems with high dimensionality. It is hypothesized that a method similar to the Action-specific Deep Recurrent Q-Network (ADRQN) [122] method may be able to solve such a challenging POMDP RL problem. It is likely many changes to this framework will be required to make it applicable to the ambiguous foraging problem presented in this research, however, given the complexity of this problem. It is also possible that a method such as this may only be applicable to a single agent solution, rather than a swarm solution. Adding other agents into the environment significantly increases the dimensionality of the problem. The proper framing of an experiment comparing the single agent Groundhog Day solution to a solution using a method similar to ADRQN would be to see how the ADRQN-like method, as a purely single agent solution, would compare in terms of resilience to ambiguity to Groundhog Day with its virtual swarm of prior experience. Again, it is expected, however, that modifying the definition of this problem to be compatible with a POMDP RL framework is going to be challenging due to the curses of dimensionality and ambiguity and that finding a solution, even with the ability of DQN-like methods to handle high dimensionality problems, may be very computationally expensive.

Another direction of future work is to further investigate other methods of decision-making under ambiguity in order to gain a better understanding of how they might incorporate the concept of resilience to ambiguity. Much of the existing research on these topics has been in the context of other fields of study, such as economics, and it has not been significantly explored for robotics and engineering, so there is additional motivation to investigate how these concepts may apply to robotic decision-making. For example, there are methods for creating additive models that combine risk and ambiguity through linear combinations [123]. Decision-making under risk, uncertainty, and ambiguity may have conflicting definitions, depending on the literature, but in this context, decision-making under risk is defined as the same as what has been termed decision-making under uncertainty in this document. That is, decision-making under risk/uncertainty is when there is knowledge about the distributions of uncertainty. Ambiguity is the lack of knowledge about the distributions of uncertainty, but in many cases the problem may not be entirely ambiguous and

knowledge about the distributions of uncertainty in some aspects of the problem may be available. In these cases, such knowledge should be utilized where available and ambiguity about the other aspects of the problem needs to be considered in conjunction.

Additionally, many existing decision-making under ambiguity methods focus on handling ambiguity through "robust" decision-making or planning [124, 125], which maximize the worst-case outcomes over a set of known models of uncertainty. These include methods that incorporate ambiguity into MDPs [54], but the limitation of these methods is that ambiguity is factored into the states (e.g., parameters defining the ambiguity are included in the state space), increasing the dimensionality and therefore the complexity of the problem. Therefore, a direction for future work here is to investigate methods that do not require the addition of such complexities. This would be done by taking the concept of parameterizing a set of state transition models by a set of ambiguity parameters $\theta$, as was introduced with the AT&E framework in Section 4.11 and (4.6), and incorporating that into a robust decision-making framework. A further extension of this, though a challenging one, would be to apply this concept to problems with partial observability (e.g., POMDPs) as well.

Many of these methods suffer the common problem of high dimensional state spaces, especially in complex real-world problems, however. Finding ways to address this challenge is a key area of research. One of the methods used to do this is state abstraction (or state aggregation), where states that may be irrelevant in certain situations are grouped together into "abstract" states, reducing the dimensionality of the problem, compared to that of the the original "ground" representation [126, 127]. Different abstractions may be used in different contexts and interpreting information through different abstractions can lead to ambiguity. Albeit a different strategy of addressing the computational issues of complex real-world problems than the trial and error strategy of AT&E, investigating how to address ambiguity through having different state abstractions could be a way of approaching decision-making under ambiguity for complex real-world problems.

And finally, another direction of future work is to extend the work that was done on the concept of estimating the "belief of the uncertainty model" (BUM) as was presented in Section 4.6. Beyond simply improving the estimation framework used to compute BUM over a fixed set of initial models, an interesting way to reframe this problem would be to investigate robotic decision-making that can estimate its proficiency at the task it is performing and then modify its behavior to attempt to achieve better proficiency. Focusing on the estimation side, a promising approach

that could be leveraged is "assumption-alignment tracking" (AAT), where the robot can self-assess its performance at the task by estimating the veracity of the assumptions encoded into its decision-making algorithms, based on observing outcomes and evaluating how well it is performing at the task [128]. The robot can then use its own prior experiences and the estimate about how well its assumptions match its ability to perform the task to use alternate assumptions to change its behavior and attempt to increase its performance. The AAT method could likely be extended to include prior experience from multiple agents in a swarm scenario as well, and incorporate that information to help each agent in the swarm to collectively estimate the veracity of their assumptions. Additionally, as a longer-term goal, this concept could be further extended to enable the robot to propose new assumptions online, as it is running and discovering that its current assumptions may not be valid. Overall, however, this aligns with the core concept of AT&E, which is to update the robot's assumptions about the nature of the ambiguity in the problem, based on its own experience and the experience of other agents if part of a swarm. Such a framework could be effective at solving complex decision-making under ambiguity problems.

## 5.4 Broader Impacts and Final Remarks

As stated in Section 1.3, the broader impacts of this work are to make robot autonomy more generalizable to complex real-world environments and situations, without the need for close human supervision. The real-world is full of ambiguity and attempting to overcome this challenge with more modeling and more training data is simply not a scalable solution to making robots that can work in the real-world. While the exact methods presented in this research are not ready-to-deploy solutions to solve complex real-world problems, they help lay the foundation for future work by establishing an understanding of the problem and baseline frameworks. This investigative research into decision-making that is resilient to ambiguity describes a new type of objective that is not considered by most other existing works. Most decision-making research focuses on finding "optimal" solutions to problems, but given the difficulty presented by complex real-world situations, is optimality the correct objective to be pursuing? Based on the discussions in Section 5.2, the author argues that in many cases it is not the correct objective and that possibly suboptimal, but resilient solutions may be a better objective to pursue in these cases. Robot autonomy that is based on this objective may not be capable of performing complex or highly specialized tasks, but it will be capable of per-

forming simple tasks reliably and without close human operator supervision. Reliability in the face of real-world complexities that are impractical to model explicitly in the robot's decision-making algorithms is one of the key factors limiting the applicability of autonomous robots in real-world situations. Resilience to ambiguity does not fully capture a way of handling all of these real-world complexities, but it is an in incremental step towards improving the reliability of robots in these situations. The benefits of autonomous robots that are able to operate more reliably in complex real-world environments are that robots can be used to perform tedious, repetitive, or dangerous tasks that must currently be performed by humans. Human society will benefit from automating these types of tasks so that the people who would have been performing them otherwise can spend their time on more personally satisfying tasks instead. Robotics and automation is not the solution to every problem, but where appropriate, it can improve people's lives.

# References


[1] F. Cuzzolin, *The Geometry of Uncertainty: The Geometry of Imprecise Probabilities*. Springer Nature, 2020.

[2] A. J. Keith and D. K. Ahner, "A survey of decision making and optimization under uncertainty," *Annals of Operations Research*, vol. 300, no. 2, pp. 319–353, 2021.

[3] V. C. Müller and M. Hoffmann, "What is morphological computation? on how the body contributes to cognition and control," *Artificial life*, vol. 23, no. 1, pp. 1–24, 2017.

[4] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[5] P. M. Morse, G. E. Kimball, and S. I. Gass, *Methods of operations research*. Courier Corporation, 2003.

[6] B. O. Koopman, "The theory of search. i. kinematic bases," *Operations research*, vol. 4, no. 3, pp. 324–346, 1956.

[7] R. C. Larson and A. R. Odoni, *Urban operations research*. No. Monograph, 1981.

[8] T. L. Saaty, *Mathematical methods of operations research*. Courier Corporation, 2004.

[9] A. M. Turing and J. Haugeland, *Computing machinery and intelligence*. MIT Press Cambridge, MA, 1950.

[10] B. Khoussainov and A. Nerode, *Automata theory and its applications*, vol. 21. Springer Science & Business Media, 2012.

[11] Z. Kohavi and N. K. Jha, *Switching and finite automata theory*. Cambridge University Press, 2009.

[12] D. N. Arden, "Delayed-logic and finite-state machines," in *2nd Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1961)*, pp. 133–151, IEEE, 1961.

[13] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pp. 46–57, IEEE, 1977.

[14] M. Y. Vardi, "An automata-theoretic approach to linear temporal logic," in *Logics for concurrency*, pp. 238–266, Springer, 1996.

[15] M. J. Fischer and R. E. Ladner, "Propositional dynamic logic of regular programs," *Journal of computer and system sciences*, vol. 18, no. 2, pp. 194–211, 1979.

[16] N. J. Nilsson *et al.*, "Shakey the robot," 1984.

[17] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[18] D. Kortenkamp, R. Simmons, and D. Brugali, "Robotic systems architectures and programming," in *Springer handbook of robotics*, ch. 12, pp. 283–305, Springer, 2016.

[19] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[20] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[21] A. Stentz *et al.*, "The focussed d^* algorithm for real-time replanning," in *IJCAI*, vol. 95, pp. 1652–1659, 1995.

[22] R. Bellman, "On the theory of dynamic programming," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 38, no. 8, p. 716, 1952.

[23] R. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503–515, 1954.

[24] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[25] R. C. Arkin, R. C. Arkin, *et al.*, *Behavior-based robotics*. MIT press, 1998.

[26] R. J. Firby, *Adaptive execution in complex dynamic worlds*. PhD thesis, Yale University, 1989.

[27] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "An architecture for autonomy," *The International Journal of Robotics Research*, vol. 17, no. 4, pp. 315–337, 1998.

[28] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das, "The claraty architecture for robotic autonomy," in *2001 IEEE Aerospace Conference Proceedings (Cat. No. 01TH8542)*, vol. 1, pp. 1–121, IEEE, 2001.

[29] I. A. Nesnas, R. Simmons, D. Gaines, C. Kunz, A. Diaz-Calderon, T. Estlin, R. Madison, J. Guineau, M. McHenry, I.-H. Shu, *et al.*, "Claraty: Challenges and steps toward reusable robotic software," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, p. 5, 2006.

[30] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, pp. 679–684, 1957.

[31] R. A. Howard, "Dynamic programming and markov processes.," 1960.

[32] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[33] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*, pp. 282–293, Springer, 2006.

[34] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable markov processes over a finite horizon," *Operations research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[35] E. J. Sondik, "The optimal control of partially observable markov processes over the infinite horizon: Discounted costs," *Operations research*, vol. 26, no. 2, pp. 282–304, 1978.

[36] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

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

[38] M. T. Spaan and N. Spaan, "A point-based pomdp algorithm for robot planning," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 3, pp. 2399–2404, IEEE, 2004.

[39] T. Smith and R. Simmons, "Point-based pomdp algorithms: Improved analysis and implementation," *arXiv preprint arXiv:1207.1412*, 2012.

[40] H. Kurniawati, D. Hsu, and W. S. Lee, "Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces.," in *Robotics: Science and systems*, vol. 2008, Zurich, Switzerland., 2008.

[41] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based pomdp solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2013.

[42] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," in *Advances in neural information processing systems*, pp. 2164–2172, 2010.

[43] H. Bai, D. Hsu, W. S. Lee, and V. A. Ngo, "Monte carlo value iteration for continuous-state pomdps," in *Algorithmic foundations of robotics IX*, pp. 175–191, Springer, 2010.

[44] J. A. Boyan and M. L. Littman, "Exact solutions to time-dependent mdps," in *Advances in Neural Information Processing Systems*, pp. 1026–1032, 2001.

[45] E. Rachelson, P. Fabiani, and F. Garcia, "Timdppoly: An improved method for solving time-dependent mdps," in *2009 21st IEEE International Conference on Tools with Artificial Intelligence*, pp. 796–799, IEEE, 2009.

[46] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[47] L. Liu and G. S. Sukhatme, "A solution to time-varying markov decision processes," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1631–1638, 2018.

[48] M. P. Wellman, M. Ford, and K. Larson, "Path planning under time-dependent uncertainty," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 532–539, Morgan Kaufmann Publishers Inc., 1995.

[49] J. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, *et al.*, "Planning under continuous time and resource uncertainty: A challenge for ai," in *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pp. 77–84, Morgan Kaufmann Publishers Inc., 2002.

[50] H. J. Einhorn and R. M. Hogarth, "Decision making under ambiguity," *Journal of Business*, pp. S225–S250, 1986.

[51] J. Etner, M. Jeleva, and J.-M. Tallon, "Decision theory under ambiguity," *Journal of Economic Surveys*, vol. 26, no. 2, pp. 234–270, 2012.

[52] J. Stoye, "Statistical decisions under ambiguity," *Theory and decision*, vol. 70, no. 2, pp. 129–148, 2011.

[53] I. Gilboa and M. Marinacci, "Ambiguity and the bayesian paradigm," in *Readings in formal epistemology*, pp. 385–439, Springer, 2016.

[54] N. Bäuerle and U. Rieder, "Markov decision processes under ambiguity," *arXiv preprint arXiv:1907.02347*, 2019.

[55] Y. Pengqian, *Risk-averse and ambiguity-averse Markov decision processes*. PhD thesis, National University of Singapore (Singapore), 2016.

[56] S. Ross, J. Pineau, B. Chaib-draa, and P. Kreitmann, "A bayesian approach for learning and planning in partially observable markov decision processes.," *Journal of Machine Learning Research*, vol. 12, no. 5, 2011.

[57] S. Saghafian, "Ambiguous partially observable markov decision processes: Structural results and applications," *Journal of Economic Theory*, vol. 178, pp. 1–35, 2018.

[58] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[59] V. Gullapalli, J. A. Franklin, and H. Benbrahim, "Acquiring robot skills via reinforcement learning," *IEEE Control Systems Magazine*, vol. 14, no. 1, pp. 13–24, 1994.

[60] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, "Reinforcement learning for robot soccer," *Autonomous Robots*, vol. 27, no. 1, pp. 55–73, 2009.

[61] J. Kober, E. Oztop, and J. Peters, "Reinforcement learning to adjust robot movements to new situations," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[62] J. Morimoto and K. Doya, "Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning," *Robotics and Autonomous Systems*, vol. 36, no. 1, pp. 37–51, 2001.

[63] H. Nguyen and H. La, "Review of deep reinforcement learning for robot manipulation," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pp. 590–595, IEEE, 2019.

[64] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *2010 IEEE/RSJ international conference on intelligent robots and systems*, pp. 3232–3237, IEEE, 2010.

[65] N. ALTUNTAŞ, E. Imal, N. Emanet, and C. N. Öztürk, "Reinforcement learning-based mobile robot navigation," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 24, no. 3, pp. 1747–1767, 2016.

[66] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[67] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5129–5136, IEEE, 2018.

[68] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.

[69] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1341–1346, IEEE, 2014.

[70] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[71] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik, "Prediction-guided multi-objective reinforcement learning for continuous robot control," in *International Conference on Machine Learning*, pp. 10607–10616, PMLR, 2020.

[72] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029, IEEE, 2019.

[73] P. Abbeel, *Apprenticeship learning and reinforcement learning with application to robotic control*. Stanford University, 2008.

[74] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

[75] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[76] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[77] Y. Xiao, J. Wu, Z. Lin, and X. Zhao, "A deep learning-based multi-model ensemble method for cancer prediction," *Computer methods and programs in biomedicine*, vol. 153, pp. 1–9, 2018.

[78] C. Tebaldi and R. Knutti, "The use of the multi-model ensemble in probabilistic climate projections," *Philosophical transactions of the royal society A: mathematical, physical and engineering sciences*, vol. 365, no. 1857, pp. 2053–2075, 2007.

[79] T. N. Palmer, Č. Branković, and D. Richardson, "A probability and decision-model analysis of provost seasonal multi-model ensemble integrations," *Quarterly Journal of the Royal Meteorological Society*, vol. 126, no. 567, pp. 2013–2033, 2000.

[80] J. Sasiadek and P. Hartana, "Sensor data fusion using kalman filter," in *Proceedings of the Third International Conference on Information Fusion*, vol. 2, pp. WED5–19, IEEE, 2000.

[81] J. Z. Sasiadek, "Sensor fusion," *Annual Reviews in Control*, vol. 26, no. 2, pp. 203–228, 2002.

[82] S.-L. Sun and Z.-L. Deng, "Multi-sensor optimal information fusion kalman filter," *Automatica*, vol. 40, no. 6, pp. 1017–1023, 2004.

[83] X.-J. Sun, Y. Gao, Z.-L. Deng, C. Li, and J.-W. Wang, "Multi-model information fusion kalman filtering and white noise deconvolution," *Information Fusion*, vol. 11, no. 2, pp. 163–173, 2010.

[84] R. Caballero-Águila, I. García-Garrido, and J. Linares-Pérez, "Information fusion algorithms for state estimation in multi-sensor systems with correlated missing measurements," *Applied Mathematics and Computation*, vol. 226, pp. 548–563, 2014.

[85] M. H. Hansen and B. Yu, "Model selection and the principle of minimum description length," *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 746–774, 2001.

[86] P. D. Grünwald, *The minimum description length principle*. MIT press, 2007.

[87] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and systems magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[88] Y. Hou, P. Edara, and C. Sun, "Situation assessment and decision making for lane change assistance using ensemble learning methods," *Expert Systems with Applications*, vol. 42, no. 8, pp. 3875–3882, 2015.

[89] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*, pp. 1–15, Springer, 2000.

[90] R. J. Scherer and S.-E. Schapke, "A distributed multi-model-based management information system for simulation and decision-making on construction projects," *Advanced Engineering Informatics*, vol. 25, no. 4, pp. 582–599, 2011.

[91] E. A. Fulton, F. Boschetti, M. Sporcic, T. Jones, L. R. Little, J. M. Dambacher, R. Gray, R. Scott, and R. Gorton, "A multi-model approach to engaging stakeholder and modellers in complex environmental problems," *Environmental Science & Policy*, vol. 48, pp. 44–56, 2015.

[92] D. Lavanya and K. U. Rani, "Ensemble decision making system for breast cancer data," *International Journal of Computer Applications*, vol. 51, no. 17, 2012.

[93] A. Khamparia, A. Singh, D. Anand, D. Gupta, A. Khanna, N. Arun Kumar, and J. Tan, "A novel deep learning-based multi-model ensemble method for the prediction of neuromuscular disorders," *Neural computing and applications*, vol. 32, no. 15, pp. 11083–11095, 2020.

[94] S. Lessmann, J. Haupt, K. Coussement, and K. W. De Bock, "Targeting customers for profit: An ensemble learning framework to support marketing decision-making," *Information Sciences*, vol. 557, pp. 286–301, 2021.

[95] S. Biswas, G. Bandyopadhyay, B. Guha, and M. Bhattacharjee, "An ensemble approach for portfolio selection in a multi-criteria decision making framework," *Decision Making: Applications in Management and Engineering*, vol. 2, no. 2, pp. 138–158, 2019.

[96] P.-C. Chang, C.-H. Liu, C.-Y. Fan, J.-L. Lin, and C.-M. Lai, "An ensemble of neural networks for stock trading decision making," in *International conference on intelligent computing*, pp. 1–10, Springer, 2009.

[97] J. Campbell, S. Stepputtis, and H. B. Amor, "Probabilistic multimodal modeling for human-robot interaction tasks," *arXiv preprint arXiv:1908.04955*, 2019.

[98] C.-J. Hoel, K. Wolff, and L. Laine, "Tactical decision-making in autonomous driving by reinforcement learning with uncertainty estimation," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1563–1569, IEEE, 2020.

[99] J. Guérin, S. Thiery, E. Nyiri, and O. Gibaru, "Unsupervised robotic sorting: Towards autonomous decision making robots," *arXiv preprint arXiv:1804.04572*, 2018.

[100] L. N. Steimle, D. L. Kaufman, and B. T. Denton, "Multi-model markov decision processes," *IISE Transactions*, vol. 53, no. 10, pp. 1124–1139, 2021.

[101] K. J. Åström and B. Wittenmark, *Adaptive control*. Courier Corporation, 2013.

[102] K. Zhou and J. C. Doyle, *Essentials of robust control*, vol. 104. Prentice hall Upper Saddle River, NJ, 1998.

[103] B. Khaldi and F. Cherif, "An overview of swarm robotics: Swarm intelligence applied to multi-robotics," *International Journal of Computer Applications*, vol. 126, no. 2, 2015.

[104] A. Jevtić and D. Andina de la Fuente, "Swarm intelligence and its applications in swarm robotics," 2007.

[105] L. Bayındır, "A review of swarm robotics tasks," *Neurocomputing*, vol. 172, pp. 292–321, 2016.

[106] J. Harwell, L. Lowmanstone, and M. Gini, "Demystifying emergent intelligence and its effect on performance in large robot swarms," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 474–482, 2020.

[107] W. A. Just and M. E. Moses, "Flexibility through autonomous decision-making in robot swarms," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, IEEE, 2017.

[108] Y. Gu, N. Ohi, K. Lassak, J. Strader, L. Kogan, A. Hypes, S. Harper, B. Hu, M. Gramlich, R. Kavi, *et al.*, "Cataglyphis: An autonomous sample return rover," *Journal of Field Robotics*, vol. 35, no. 2, pp. 248–274, 2018.

[109] Y. Gu, J. Strader, N. Ohi, S. Harper, K. Lassak, C. Yang, L. Kogan, B. Hu, M. Gramlich, R. Kavi, *et al.*, "Robot foraging: Autonomous sample return in a large outdoor environment," *IEEE Robotics & Automation Magazine*, vol. 25, no. 3, pp. 93–101, 2018.

[110] N. Ohi, K. Lassak, R. Watson, J. Strader, Y. Du, C. Yang, G. Hedrick, J. Nguyen, S. Harper, D. Reynolds, *et al.*, "Design of an autonomous precision pollination robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7711–7718, IEEE, 2018.

[111] E. A. Hansen, "Solving pomdps by searching in policy space," *arXiv preprint arXiv:1301.7380*, 2013.

[112] H. P. Young, "Learning by trial and error," *Games and economic behavior*, vol. 65, no. 2, pp. 626–643, 2009.

[113] T. Winkler, S. Junges, G. A. Pérez, and J.-P. Katoen, "On the complexity of reachability in parametric markov decision processes," *arXiv preprint arXiv:1904.01503*, 2019.

[114] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[115] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[116] D. Anderson, E. Fredrickson, and R. Estell, "Managing livestock using animal behavior: mixed-species stocking and flerds," *Animal*, vol. 6, no. 8, pp. 1339–1349, 2012.

[117] H. Ramis (Director), "Groundhog day." Columbia Pictures, 1993.

[118] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3, pp. 293–321, 1992.

[119] J. E. Davidson, R. J. Sternberg, R. J. Sternberg, *et al.*, *The psychology of problem solving*. Cambridge university press, 2003.

[120] H. A. Simon, "Rational decision making in business organizations," *The American economic review*, vol. 69, no. 4, pp. 493–513, 1979.

[121] W. C. Stirling, *Satisficing Games and Decision Making: with applications to engineering and computer science*. Cambridge University Press, 2003.

[122] P. Zhu, X. Li, P. Poupart, and G. Miao, "On improving deep reinforcement learning for pomdps," *arXiv preprint arXiv:1704.07978*, 2017.

[123] Y. He, J. S. Dyer, J. C. Butler, and J. Jia, "An additive model of decision making under risk and ambiguity," *Journal of Mathematical Economics*, vol. 85, pp. 78–92, 2019.

[124] G. N. Iyengar, "Robust dynamic programming," *Mathematics of Operations Research*, vol. 30, no. 2, pp. 257–280, 2005.

[125] W. Wiesemann, D. Kuhn, and B. Rustem, "Robust markov decision processes," *Mathematics of Operations Research*, vol. 38, no. 1, pp. 153–183, 2013.

[126] L. Li, T. J. Walsh, and M. L. Littman, "Towards a unified theory of state abstraction for mdps.," in *AI&M*, 2006.

[127] D. Abel, D. Arumugam, L. Lehnert, and M. Littman, "State abstractions for lifelong reinforcement learning," in *International Conference on Machine Learning*, pp. 10–19, PMLR, 2018.

[128] A. Gautam, T. Whiting, X. Cao, M. A. Goodrich, and J. W. Crandall, "A method for designing autonomous robots that know their limits," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 121–127, IEEE, 2022.

[129] P. Koonce and L. Rodegerdts, "Traffic signal timing manual.," tech. rep., United States. Federal Highway Administration, 2008.

[130] X. Li, G. Li, S.-S. Pang, X. Yang, and J. Tian, "Signal timing of intersections using integrated optimization of traffic quality, emissions and fuel consumption: a note," *Transportation Research Part D: Transport and Environment*, vol. 9, no. 5, pp. 401–407, 2004.

[131] R. Mohajerpoor, M. Saberi, and M. Ramezani, "Analytical derivation of the optimal traffic signal timing: Minimizing delay variability and spillback probability for undersaturated intersections," *Transportation research part B: methodological*, vol. 119, pp. 45–68, 2019.

[132] E. Walraven, "Solvepomdp." https://www.erwinwalraven.nl/solvepomdp/.

[133] A. R. Cassandra, M. L. Littman, and N. L. Zhang, "Incremental pruning: A simple, fast, exact method for partially observable markov decision processes," *arXiv preprint arXiv:1302.1525*, 2013.

# A

# Traffic Intersection Controller Case Study

## A.1 Problem Description: Simulated Traffic Signal Controller

A traffic signal (i.e., a stoplight) controller is a simple, intuitive example of a problem that can be described and solved as a POMDP. Most traffic signal controllers in the real world are based on heuristic FSM or optimization problem implementations, solved offline for one set of fixed model parameters, with little consideration of uncertainty during execution [129–131]. This is not to imply these are poor solutions, however. It may be cost prohibitive, and potentially unsafe in the event of failure, to provide traffic signal controllers with the necessary computing hardware and sensing capabilities to implement more complex decision-making algorithms, such as POMDPs. Therefore, it is logical why simpler solutions are preferred. For the purposes of autonomous decision-making research, however, a simulated traffic signal problem, that can incorporate uncertainty, is a useful and intuitive case study to investigate.

This simulated traffic signal controller case study consists of a single, isolated traffic light in a four-way road intersection. In order to keep the problem easily solvable as a discrete state POMDP, the number of states, actions, and observations must be kept small. Therefore, the models of traffic flow through the intersection used in this work are very simple. Cars turning at the intersection are not modeled and the only two possible states of the traffic light are considered. The arrival and departure of cars at the traffic intersection through the the north-south (NS) direction and the east-west (EW) direction is stochastic. The states consist of three elements: the state of the traffic light $L$ and the total number of cars waiting in the NS direction and that of the EW direction, $m$ and $n$, respectively. The state of the light $L$ can be one of two symbols: $l_0$ (NS open, EW closed) and $l_1$ (NS closed, EW open) and $m$ and $n$ range from zero to a maximum number of cars $c_{\max}$. The two possible actions are to set the state of the light to $l_0$ ($a_0$) or to $l_1$ ($a_1$). An illustration of this concept is shown in Fig. A.1.1.

The possible observations represent the outputs of uncertain, low-fidelity traffic sensors in both the NS and EW directions. The traffic sensor outputs can be one of the discrete symbols: none, low, or high number of cars waiting. Therefore, an observation event $o$ at each time step consists of an ordered pair of symbols representing the measurement from the NS and EW traffic sensors, respectively (e.g., $o = \{\text{low}, \text{high}\}$). The state transition model $T(s', a, s)$ describes the probability of how the number of cars waiting in each direction $(m, n)$ may change at each time step. For

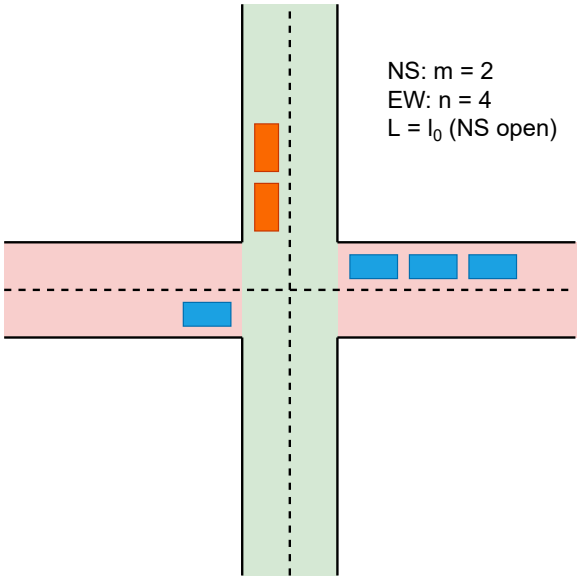



**Figure A.1.1:** Illustration of four-way traffic signal problem, showing an example set of states

simplicity, the problem is constrained such that at each time step, there is a probability that the number of cars waiting at the intersection may increase by one, stay the same, or decrease by one. Therefore, the transition model is of the following form, where all other transition probabilities not explicitly defined are equal to zero.

$$T(s, a, s') = \begin{bmatrix} P(m' = m-1, n' = n-1) & P(m' = m-1, n' = n) & P(m' = m-1, n' = n+1) \\ P(m' = m, n' = n-1) & P(m' = m, n' = n) & P(m' = m, n' = n+1) \\ P(m' = m+1, n' = n-1) & P(m' = m+1, n' = n) & P(m' = m+1, n' = n+1) \end{bmatrix} \quad \text{(A.1)}$$

In the case where the state of the light is $l_0$, meaning NS open, all entries where $n' = n - 1$ are zero, because the number of cars waiting in the EW direction cannot decrease. And similarly, when the state of the light is $l_1$, meaning EW open, all entries where $m' = m - 1$ are zero, because the number of cars waiting in the NS direction cannot decrease. When a traffic direction is open, the number of cars waiting in that direction is likely to decrease rapidly to zero and very likely to stay at

zero. When a traffic direction is closed, the number of cars cannot decrease, but is similarly likely to stay the same, or increase. This represents cars arriving and needing to wait at the closed direction of the intersection at a reasonable rate, over time. To make the decision-making problem more interesting, the transition probabilities for the NS and EW directions are not symmetric, meaning there is more traffic in one direction than the other. In this case, it was chosen that the EW direction generally has heavier traffic than the NS direction.

The observation model $O(s', a, o)$ describes the uncertainty of the traffic sensors, meaning the probability that the observation will be a pair of the traffic sensor symbols (none, low, or high), for each direction, given that the number of cars waiting in each direction is a particular value. In other words, the observation model is defined as follows.

$$O(s', a, o) = P(o|m, n) \tag{A.2}$$

Since the objective is to minimize the number of cars waiting at the intersection, the reward function is defined simply as follows, applying an incremental penalty at each time step based on the total number of cars waiting at the intersection in both directions.

$$R(s) = -m - n \tag{A.3}$$

## A.2 Case Study: Simulated Traffic Signal Controller

The traffic signal problem described previously in Section A.1 is implemented in simulation as described here. As a baseline, a traditional POMDP solution, based on a single model of traffic uncertainty, is compared to the MM-POMDP solution, based on multiple models.

The single POMDP solution assumes a single state transition model, describing the "expected" traffic conditions, and a single observation model describing the uncertainty of the traffic sensors. The *normal traffic* conditions described by the transition model states that when a traffic direction is open, the number of cars waiting in that direction is likely to decrease rapidly to zero and very likely to stay at zero. When a traffic direction is closed, the number of cars cannot decrease, but is similarly likely to stay the same, or increase. This represents cars arriving and needing to wait at the closed direction of the intersection at a reasonable rate, over time. The traffic flow models for the

NS direction and the EW direction are identical in this case. The observation model represents a *low uncertainty* in the traffic sensors.

The MM-POMDP solution, however, uses multiple transition and observation models to solve for separate POMDP policies, representing different traffic conditions and traffic sensor uncertainty, assuming that it is possible the single models used by the single POMDP solution may be incorrect. All of these models follow the same structure, as described in Section A.1, but they differ in terms of parameters. The first transition model $T_0$ is the same *normal traffic* conditions as described above for the single POMDP. The second transition model $T_1$ represents *rush hour traffic* conditions. Compared to the normal traffic model, this model is more likely for the number of cars waiting to increase while a direction is closed and the number of cars waiting in the open direction is less likely to decrease to zero as quickly. This represents a higher volume of traffic moving through the intersection, in both directions, due to rush hour traffic. The third transition model $T_2$ represents *game day traffic* conditions, a hypothetical situation of a major sporting event happening in the city that day, where traffic volume is elevated in the EW direction, but not in the NS direction. The state transition models in this case are not symmetric. The traffic volume in the EW direction is more similar to that of the rush hour conditions, but the traffic volume in the NS direction remains similar to normal traffic conditions. The fourth transition model $T_3$ represents *late night traffic* conditions, where the number of cars out on the road is minimal. This model is symmetric, like the first two, but the probability of cars accumulating at the intersection when a direction is closed is much smaller than normal traffic conditions. The fifth and final transition model $T_4$ represents an additional *unknown traffic* condition, for which a POMDP policy is not solved, representing that the traffic intersection controller has no prior knowledge of this situation. This model may represent an unexpected situation, such as road work, or a traffic accident and is similar to the game day traffic conditions, except that NS is biased more heavily instead of EW. Three observation models are also used, to represent different levels of traffic sensor uncertainty. The first model $O_0$ has low uncertainty, the second model $O_1$ has higher uncertainty. The specific numeric values of these models are included in Section A.3.

The true underlying transition and observation models used in the simulation for all three decision-making methods (FSM, single POMDP, and MM-POMDP) are varied across all combinations of each transition model $\{T_0, \ldots, 4\}$ and each observation model $\{O_0, O_1\}$, forming the set of uncertainty models $\Xi = \mathbb{T} \times \mathbb{O} = \{\xi_0, \xi_1, \ldots, \xi_9\}$. Uncertainty models $\{\xi_0, \ldots, \xi_4\}$ represent transi-

tion models $\{T_0, \ldots, 4\}$ for observation model $O_0$ and $\{\xi_5, \ldots, \xi_9\}$ represent the same transition models for observation model $O_1$. Each POMDP policy is solved offline using the `SolvePOMDP` program [132], implementing the incremental pruning method [133]. For the single POMDP, a single policy is solved using $T_0$ and $O_0$. For the MM-POMDP, transition models $T_0$ through $T_3$ are considered known to the decision-maker, as are all the observation models and each combination of these pairs of transition and observation models make up the the set of models for which policies are solved. Transition model $T_4$ is considered to be unknown to the decision-maker and it is not contained in any of the decsion-maker's models and therefore none of the policies have knowledge of this transition model. This case of the unknown model is used to evaluate the resilience of the MM-POMDP decision-maker to unmodeled forms of uncertainty. Each decision-making method was tested in simulation by performing 1000 Monte Carlo trials, for 200 timesteps, of the fifteen different traffic scenarios, representing each model combination. For the MM-POMDP, two action arbitration methods, majority voting and weighted majority voting, are tested for each scenario. The performance of each decision-making strategy is compared by examining the distribution of the accumulated reward $U$, defined in (4.2), for each time step, over the Monte Carlo trials and each scenario.

Since the states are partially observable, the decision-makers cannot evaluate this for themselves, but since the truth states can be easily recorded from simulation trials, this can be evaluated externally, as is done here. The accumulated reward $U$ is compared across all of these scenarios in terms of the empirical cumulative distribution (eCDF) $\widehat{F}_n(U)$, which represents the fraction of Monte Carlo trials that perform at less than or equal to a certain value of accumulated reward. These results are presented in Fig. A.2.1.

It can be seen that some true model scenarios expose the sensitivity of certain arbitration strategies more than others. For example, the single POMDP decision-maker, based on the "normal" traffic model, clearly performs worse in the "rush hour" scenario, but it does not perform as poorly, relative to the two MM-POMDP arbitration methods, in others. It can also be seen, however, that when the model matches the true model, the single POMDP performs well, as is seen in the "normal" traffic scenario. This is expected, since the single POMDP matches the true model. Interestingly, however, the majority voting MM-POMDP strategy performs poorly in the "normal" traffic scenario. This can be attributed to all decision-makers, based on all known models, having an equal

vote, and the other decision-makers out-voting the likely more informed actions that would have been selected by the single decision-maker based on the correct model. The "normal" and "game day" scenarios show sensitivity to this, while the other scenarios do not show it as strongly; however, it is expected that not all scenarios have the same types of sensitivities. The addition of the BUM weights to the majority voting scheme shows that it helps the MM-POMDP behave more like a single POMDP that matches the true model, since the BUM is able to converge on the true model and weight the actions selected by the decision-maker based on the correct model more highly than others. The scenarios with the higher observation model uncertainty tend to follow mostly the same trends, but with degraded performance, as expected when the uncertainty is higher.

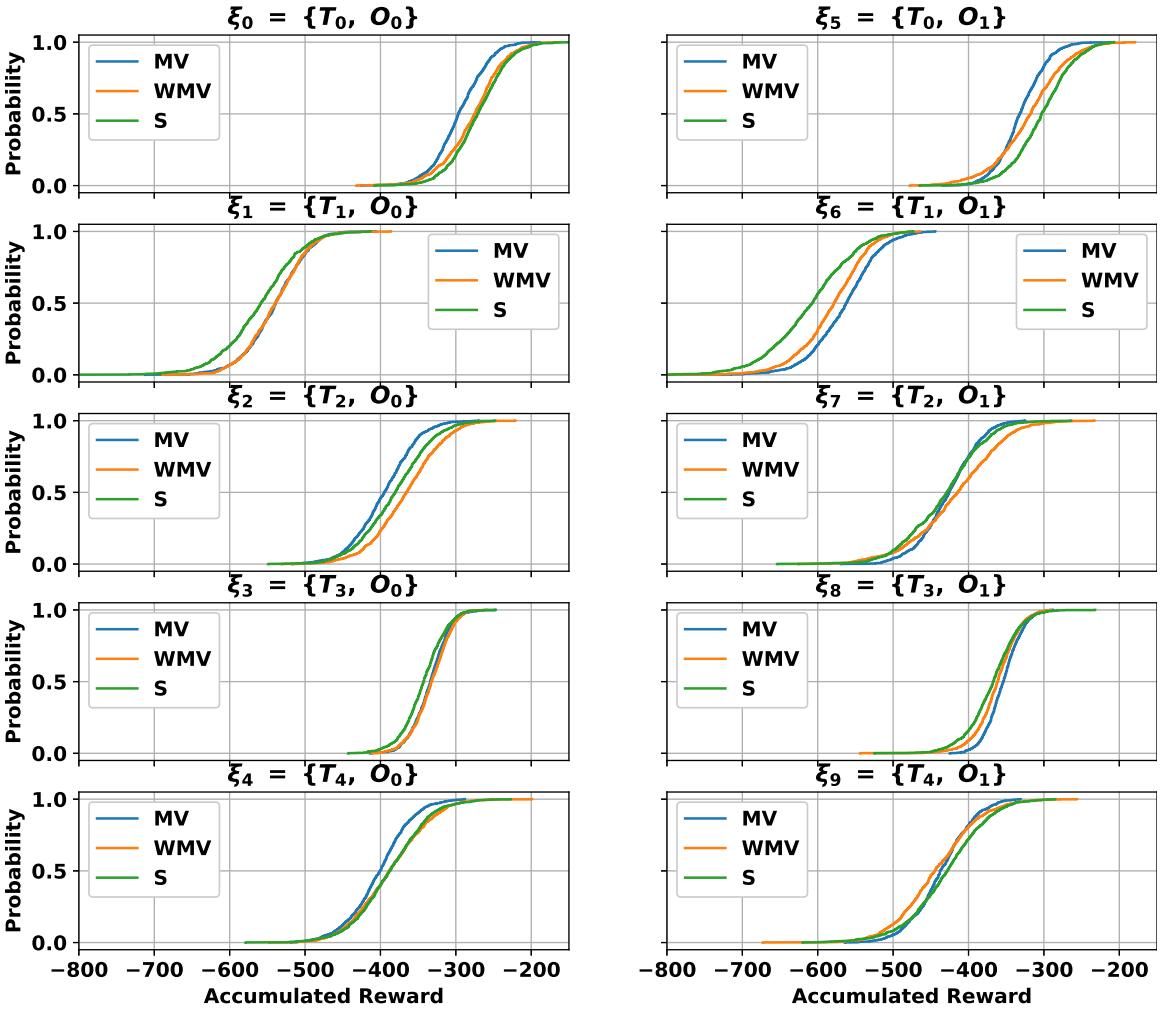



**Figure A.2.1:** Traffic signal results for each true model scenario and for each action arbitration strategy. S = single POMDP, MV = majority voting, WMV = weighted majority voting.

## A.3 Traffic Intersection Uncertainty Models

| Action: NS Open | | | |
|---|---|---|---|
| | $n' = n - 1$ | $n' = n$ | $n' = n + 1$ |
| $m' = m - 1$ | 0.00 | 0.73 | 0.20 |
| $m' = m$ | 0.00 | 0.05 | 0.01 |
| $m' = m + 1$ | 0.00 | 0.01 | 0.00 |

| Action: EW Open | | | |
|---|---|---|---|
| | $n' = n - 1$ | $n' = n$ | $n' = n + 1$ |
| $m' = m - 1$ | 0.00 | 0.00 | 0.00 |
| $m' = m$ | 0.73 | 0.05 | 0.01 |
| $m' = m + 1$ | 0.20 | 0.01 | 0.00 |

**Table A.3.1:** Normal traffic conditions transition probabilities

| Action: NS Open | | | |
|---|---|---|---|
| | $n' = n - 1$ | $n' = n$ | $n' = n + 1$ |
| $m' = m - 1$ | 0.00 | 0.50 | 0.40 |
| $m' = m$ | 0.00 | 0.05 | 0.04 |
| $m' = m + 1$ | 0.00 | 0.01 | 0.00 |

| Action: EW Open | | | |
|---|---|---|---|
| | $n' = n - 1$ | $n' = n$ | $n' = n + 1$ |
| $m' = m - 1$ | 0.00 | 0.00 | 0.00 |
| $m' = m$ | 0.50 | 0.05 | 0.01 |
| $m' = m + 1$ | 0.40 | 0.04 | 0.00 |

**Table A.3.2:** Rush hour traffic conditions transition probabilities

| Action: NS Open | | | |
|---|---|---|---|
| | $n' = n - 1$ | $n' = n$ | $n' = n + 1$ |
| $m' = m - 1$ | 0.00 | 0.50 | 0.40 |
| $m' = m$ | 0.00 | 0.05 | 0.04 |
| $m' = m + 1$ | 0.00 | 0.01 | 0.00 |

| Action: EW Open | | | |
|---|---|---|---|
| | $n' = n - 1$ | $n' = n$ | $n' = n + 1$ |
| $m' = m - 1$ | 0.00 | 0.00 | 0.00 |
| $m' = m$ | 0.73 | 0.05 | 0.01 |
| $m' = m + 1$ | 0.20 | 0.01 | 0.00 |

**Table A.3.3:** Game day traffic conditions transition probabilities (EW worse than NS)

| Action: NS Open | | | |
|---|---|---|---|
| | $n' = n - 1$ | $n' = n$ | $n' = n + 1$ |
| $m' = m - 1$ | 0.00 | 0.90 | 0.07 |
| $m' = m$ | 0.00 | 0.02 | 0.01 |
| $m' = m + 1$ | 0.00 | 0.00 | 0.00 |

| Action: EW Open | | | |
|---|---|---|---|
| | $n' = n - 1$ | $n' = n$ | $n' = n + 1$ |
| $m' = m - 1$ | 0.00 | 0.00 | 0.00 |
| $m' = m$ | 0.90 | 0.02 | 0.00 |
| $m' = m + 1$ | 0.07 | 0.01 | 0.00 |

**Table A.3.4:** Late night traffic conditions transition probabilities

| Action: NS Open | | | |
|---|---|---|---|
| | $n' = n - 1$ | $n' = n$ | $n' = n + 1$ |
| $m' = m - 1$ | 0.00 | 0.73 | 0.20 |
| $m' = m$ | 0.00 | 0.05 | 0.01 |
| $m' = m + 1$ | 0.00 | 0.01 | 0.00 |

| Action: EW Open | | | |
|---|---|---|---|
| | $n' = n - 1$ | $n' = n$ | $n' = n + 1$ |
| $m' = m - 1$ | 0.00 | 0.00 | 0.00 |
| $m' = m$ | 0.50 | 0.05 | 0.01 |
| $m' = m + 1$ | 0.04 | 0.04 | 0.00 |

**Table A.3.5:** Unknown traffic conditions (i.e., road work) transition probabilities (NS worse than EW)

| | $o_{\text{none}}$ | $o_{\text{low}}$ | $o_{\text{high}}$ |
|---|---|---|---|
| num cars < 1 | 0.95 | 0.2 | 0.01 |
| 1 >= num cars < 5 | 0.2 | 0.75 | 0.15 |
| num cars > 5 | 0.1 | 0.25 | 0.75 |

**Table A.3.6:** Low uncertainty observation model

| | $o_{\text{none}}$ | $o_{\text{low}}$ | $o_{\text{high}}$ |
|---|---|---|---|
| num cars < 1 | 0.75 | 0.25 | 0.05 |
| 1 >= num cars < 5 | 0.25 | 0.5 | 0.2 |
| num cars > 5 | 0.2 | 0.35 | 0.6 |

**Table A.3.7:** High uncertainty observation model

# B

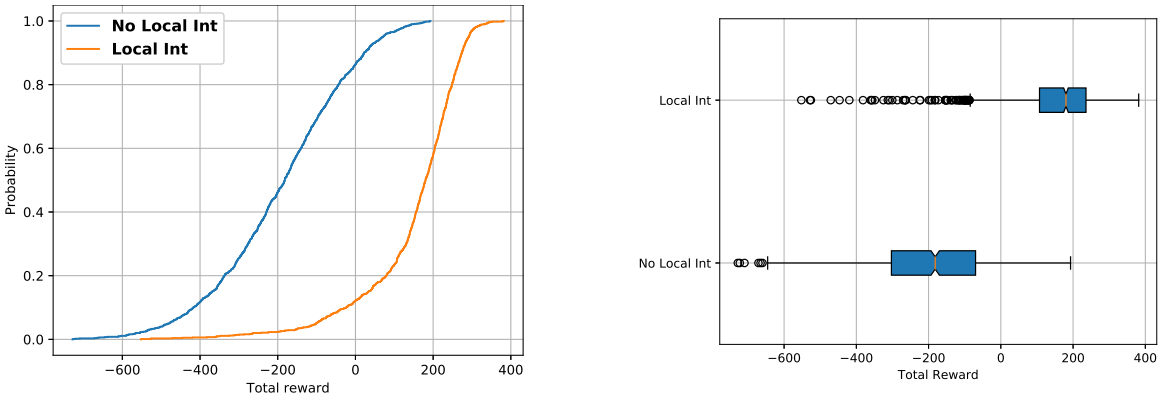
# Interpreting Resilience to Ambiguity Results

## B.1 Evaluating Resilience to Ambiguity from Distributions over Monte Carlo Trials

The performance of any decision-making problem involving stochastic outcomes must be evaluated in terms of the distribution of outcomes over a large number of Monte Carlo trials. In this work, the outcomes are evaluated in terms of the total accumulated reward $U$, as defined in (4.2). Distributions of $U$ over the Monte Carlo trials are represented as the empirical cumulative distribution function (eCDF) $\widehat{F}_n(U)$ of the accumulated reward, as defined in (B.1).

$$\widehat{F}_n(U) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{u_i \leq U} \tag{B.1}$$

Where $n$ is the number of trials and $\mathbb{1}_{u_i \leq U}$ is an indicator function, returning 1 if the trial value $u_i$ is less than or equal to the query value $U$ and 0 otherwise. An example pair of eCDF plots of $U$ comparing two decision-making strategies is shown in Fig. B.1.1a. Another representation of these same distributions, in terms of a box and whisker plot (or just box plot) is shown in Fig. B.1.1b.




**(a)** Example eCDF of $U$

**(b)** Example box plot of $U$

**Figure B.1.1:** Example distributions of outcomes of two decision-making strategies being compared.

These plots show a comparison between two decision-making strategies, performing the same task. In this case, the *No Local Int* strategy is considered the baseline, while the *Local Int* strategy is the one being evaluated. Multiple characteristics of this plot are used to judge whether the strategy

in question is more resilient than the baseline. The first is to compare the difference between the median and lower quartile (0.25 eCDF probability) regions. The further to the right (i.e., the higher value of $U$) these regions of the strategy in question are than those of the baseline, the more resilient this strategy is considered to be, relative to the baseline. If these regions had similar values or if they were to the left of the baseline, then this strategy would not be considered resilient, relative to the baseline. Another characteristic that is considered in conjunction with this is the slope of the curve between the lower and upper quartile (0.25 to 0.75 eCDF probability). The more "vertical" the slope, compared to that of the baseline, the less variance in the performance of the trials. This means that multiple trials are more consistent and the probability of degraded performance, compared to other trials is low. In the case study problems considered in this research, there is a chance of very low performing trials, so the sharp tails at the minimum end of the eCDF plot, representing all of the low outliers as can be seen in the box plots, are not considered when evaluating resilience. Bounding the worst case performance, as is done in robust decision-making, is a different objective than resilience to ambiguity.

Also in this example, the upper quartile region and the max region is also to the right of the baseline, so this strategy is also higher performance, in addition to being more resilient to ambiguity. A strategy can be both more resilient to ambiguity and higher performance, but it does not need to be higher performance in order to be more resilient to ambiguity. The upper regions of the eCDF may be similar, but as long as the lower regions tend to be further to the right than the baseline and the slope of the lower to upper quartile region is at least as steep as the baseline, then that strategy is considered to be resilient to ambiguity.