Thesis

Learning based approaches to practical challenges in multi-agent active search

Arundhati Banerjee

CMU-ML-25-101

January 2025

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Jeff Schneider (Chair)
Geoffrey J. Gordon
Barnabás Póczos
Yisong Yue

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by Department of Energy award DESC0021414 and United States Army awards W911NF1820218, W911NF20F0017, W52P1J22F0103, W519TC23C0030 and W519TC23C0031.

Keywords: active search, multi-agent, sequential decision making, reinforcement learning, decision making under uncertainty, decentralized asynchronous multi-agent active search, Thompson sampling, Monte Carlo Tree Search, multi-objective decision making, cost-aware active search, Kalman filter, probability hypothesis density, PHD filter, diffusion models for decision making, gradient guided diffusion for active search, optimism bias, inductive bias in active search, Bayesian adaptive decision making

For my parents and my grandparents

Abstract

Interactive decision making is essential for the functioning of autonomous agents in both software and embodied applications. Typically, agents interact in a multi-agent environment with the goal of fulfilling individual or shared objectives. In this thesis, we study the multi-agent adaptive decision making problem in the framework of Multi-Agent Active Search (MAAS) with a focus on applications like search and rescue, wildlife patrolling or environment monitoring with multi-robot teams.

MAAS involves a team of robots (agents) deciding *when* and *where* to gather information about their surroundings, conditioned on their past observations, in order to estimate the presence and position of different objects of interest (OOIs) or targets. Agents communicate with each other asynchronously, without relying on a central controller to coordinate the agents' interactions. Realistically, inter-agent communications may be unreliable, and robots in the wild have to deal with noisy observations and stochastic environment dynamics. Our setup, described in Chapter 1, formalizes MAAS with practical models of real-world sensing, noise, and communication constraints for aerial and ground robots.

Part I of this thesis studies the benefits of non-myopic lookahead decision making in MAAS with Thompson sampling and Monte Carlo Tree Search. Additionally, we consider a multi-objective pareto-optimization setup for cost-aware active search, highlighting the challenges due to partial observability, decentralized multi-agent decision making, and computational complexity of combinatorial state and action spaces. In Part II, we focus on the practical challenges due to observation noise and dynamic targets in multi-agent active search and tracking. Our proposed algorithms using Bayesian filtering in these settings empirically demonstrate the importance of uncertainty modeling for inference and decision making. Part III shifts focus to generative models for decision making, particularly denoising diffusion sampling for lookahead MAAS with observation noise. Finally, we discuss the applicability and limitations of these methods in the context of multi-agent decision making in robotics and other applications with similar real world constraints.

Acknowledgments

I am extremely grateful for the many opportunities in life that led me here and the support and encouragement from so many teachers, mentors, peers, friends and family that sustained me along the way.

I would like to begin by thanking my advisor, Prof. Jeff Schneider for taking me on as a naïve PhD student and providing the support, mentorship and guidance to help me become a researcher. Jeff has taught me to think through asking the right questions, encouraged me to persevere through some daunting research problems and always provided the space and freedom to explore new ideas. Our research meetings have helped me gain perspective and grasp the bigger picture when I was too bogged down by pesky bugs in my experiments or flaws in my algorithms, and I am very grateful for all the kindness, patience and encouragement Jeff showed me in those times. Over the years, I have also received valuable career advice from Jeff that has given me the confidence to choose my own path. For all this, I am very thankful.

I would also like to thank my committee: Prof. Geoff Gordon, Prof. Barnabás Póczos and Prof. Yisong Yue for their time, support and encouragement. I am always inspired by Geoff's extensive expertise across ML research topics and I am grateful for his insightful questions and advice in shaping this thesis. As a TA, I learned a lot by attending the scalability in ML class taught by Barnabás and from his insights bridging ML theory and applications. During the course of my PhD, I have been inspired by several interesting research directions pursued in Yisong's lab and I am really grateful for his participation in my committee.

I am fortunate to have received mentorship from several internship supervisors and collaborators during and prior to my PhD. I spent the summer of 2022 interning with Stephan Zheng, Soham Phade and Prof. Stefano Ermon at Salesforce, where they supported me in exploring exciting research directions complementary to my thesis. I am also thankful to Prof. Yan Liu at USC, Prof. Ivan Kisel at FIAS and Dr. Subhasis Chattopadhyay at VECC for providing me with the opportunity to gain research experience as an undergraduate student, as well as encouraging me and paving the way for me to pursue a PhD.

I am very grateful to CMU SCS and especially the Machine Learning department for providing a collaborative and supportive atmosphere to students, that gave me great comfort during many stressful times. I am also deeply appreciative of the many professors and teaching staff at MLD for creating an enriching learning experience. Many thanks to the MLD staff for always looking out for us and especially to Diane Stidle who has been a constant friendly face on the 8th floor of GHC and continues to cheer us on through every milestone in the PhD program.

MLD has also given me the opportunity to interact with and learn from some brilliant, kind and truly amazing peers over the years. I am most grateful for the friendships that carried me through, especially in my cohort: Ashwini Pokle, Bingbin Liu, Stephanie Milani, Tanya Marwah, and Yusha Liu. I found a great roommate and foodie friend in Ashwini, a kind and joyful birthday twin in Bingbin, an inspiring and supportive force in Stephanie, a constant source of encouragement and growth in Tanya, and a wise and dependable friend in Yusha. They have made my time at CMU filled with memories that I will cherish for the rest of my life and for that I consider myself lucky.

Many thanks to the students and postdocs in the Auton Lab: Ramina Ghods, Conor Igoe, Tejus Gupta, Nikhil Bakshi, Ian Char, Youngseog Chung, Adam Villaflor, Brian Yang for always making time for the research chats and practice talks. Thank you to my office mates: Bingbin Liu, Yuda Song, Niki Hasrati, Tomas Gonzalez, Jacob Tyo and Haitian Sun for the many fun conversations over the years. Thanks also to the many friends and collaborators who helped me with research questions, career advice and guidance during my PhD: Biswajit Paria, Otilia Stretcu, Lisa Lee, Mohit Sharma, Che-Ping Tsai, Ritam Dutt, Ashrujit Ghoshal, Sagnik Saha and Saptarashmi Bandyopadhyay.

A special shoutout to my friends Akshita, Anjali, Anchita, Sutanuka and Mayurakshi who have known me from way back when and continue to catch up from time to time, no matter where in the world we are. To my alma maters, IIT Kharagpur and South Point High School, thank you for providing me with a strong foundation and fond memories.

Finally, I owe a debt of gratitude to my family. Thank you to my parents for believing in me, for giving me the strength to make my own choices and for teaching me to persevere in pursuit of my dreams. I would not be here without the many sacrifices they have made for my brother and me. Thank you to my brother for always being supportive, from helping me apply to grad school to helping me move cities for a new job after my PhD. Last, but not the least, I am very thankful for my grandmother, my dear Dida, for her unending love and wisdom and for being so proud of me. They have given me the roots to grow and the wings to fly. I truly could not have done this without them.

Contents

List of Figures

List of Tables

1 | Introduction

A large number of problems in scientific discovery can be characterized as the search for a set of rare, optimal solutions to a query in a large solution space. For example, in materials science and drug discovery, experiments are designed to identify compounds with certain desirable properties. In computer science and engineering, designing efficient search algorithms has been at the center of many of the advancements in technology over past decades. Such approaches are often bottlenecked by the cost of identifying or labeling a query response as optimal or not: for example, through synthetic experiments, clinical trials or user studies which can be quite expensive. This necessitates strategic exploration of the possible solution space with queries that can help maximize information or minimize uncertainty about unknowns.

Machine learning based approaches to searching for objects of interest encompass a number of related topics which can be characterized broadly as methods for sequential decision making under uncertainty. Adaptive experiment design proposes methods to maximize information about a task conditioned on already available observations. Similarly, active learning is concerned with identifying the most informative data for a task. Prior work in these topics has built on Bayesian optimization (BO) and Multi-Armed Bandit (MAB) algorithms, among others, for optimizing expensive functions. Bayesian optimization relies on a surrogate model, which is often a Gaussian Process, to represent the unknown function, along with an acquisition function to guide the search for the next function evaluation point with the goal of finding the global optimum. The Multi-Armed Bandit framework, on the other hand, provides a simplistic model for decision-making under uncertainty, where the goal is to maximize cumulative reward by balancing exploration and exploitation in the decision (or action) space. In spite of such broad range of approaches, there are several practical limitations to applying these methods in real world applications (Murphy, 2025).

The work presented in this thesis is motivated by the goal of developing autonomous agents capable of interacting in the physical word. Consider, for example, search and rescue missions using teams of robots looking for survivors in disaster zones. Although there have been studies showing the effectiveness of deploying autonomous robots in the aftermath of mining accidents (Murphy et al., 2009), forest fires (Wiki), earthquakes etc. (Murphy, 2004b; Murphy et al., 2011), we are yet to achieve scalable adoption of such systems (Carlson and Murphy, 2005). Focusing on the algorithms in use for similar applications, some of the immediate challenges to their scalable adoption include dependence on myopic or greedy decision making which prevents sufficient exploration in the agent's environment, misaligned or overlooked objectives that lead to undesirable agent behavior, inefficient coordination among agents leading to delays in achieving goals, as well as ineffective modeling of the robot's uncertainty about the environment or its task for active feedback and guidance in

unknown scenarios. Recently, even with the application of foundation models for different tasks in interactive decision making (Yang et al., 2023), there remain a number of open challenges in terms of model training and deployment in the real world.

In this thesis, we propose the multi-agent active search framework to abstract away the high level interactive decision making problem for active target recovery with multi-robot teams. Our problem formulation is designed to capture the fundamental characteristics of real world systems like unreliable inter-agent communication, accuracy-cost trade-off in agent actions and the need for efficient, real-time online decision making. We focus on a diverse range of methods spanning both machine learning and classical robotics approaches to analyze their strengths and weaknesses in multi-agent active search and propose novel advances toward developing robust autonomous agents for the real world.

Notation. We will assume the following conventions in the rest of this document. Non-bold characters represent scalars. Lowercase and uppercase boldface letters represent column vectors and matrices respectively. $\mathcal{X}$ denotes a set of $|\mathcal{X}|$ elements. $\boldsymbol{A}^T$ is the transpose for a matrix $\boldsymbol{A}$. $\boldsymbol{I}_n$ denotes an $n \times n$ identity matrix. The ith entry of a vector $\boldsymbol{a}$ is $[\boldsymbol{a}]_i$ and the (i, j)th entry of a matrix $\boldsymbol{A}$ is $[\boldsymbol{A}]_{ij}$. $\text{diag}(\boldsymbol{a})$ is a square matrix with $\boldsymbol{a}$ on the main diagonal. $q * A$ indicates multiplication of scalar q with every element of $\boldsymbol{A}$. $\mathbf{1}_{n\times 1}$ indicates a $n \times 1$ dimensional vector of ones. In addition, for each chapter, please note the respective use of notation as described therein.

1.1 Active Search

Active search focuses on online, adaptive sequential decision making for recovering targets or objects of interest (OOI) in unknown environments (Ghods et al., 2021b). Assuming the search space to be a set $\mathcal{X} \subseteq \mathbb{R}^{\text{dim}}$, with the dataset of previously collected query and observations as $\mathcal{D}_t = \{(x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t)\}$, an agent performing active search aims to select subsequent query location $x_{t+1} \in \mathcal{X}$ such that *all* targets $\mathcal{X}_k = \{x'_1, x'_2, \ldots x'_k\} \subset \mathcal{X}$ are recovered after a finite number of decision making steps T, i.e. $\{(x'_1, y'_1), ..., (x'_k, y'_k)\} \subset \mathcal{D}_T$. The number of targets k or the target set $\mathcal{X}_k$ are not known in advance and targets are sparsely distributed in the search space i.e. $|\mathcal{X}_k| \ll |\mathcal{X}|$. Moreover, gathering observations or labels y at query locations x is expensive, therefore $T \ll |\mathcal{X}|$.

In this thesis, agents are robots, for example, autonomous aerial drones or ground vehicles. Active search with robots has a wide range of applications. In disasters like earthquakes, hurricanes or gas leaks, robots deployed to rescue the human survivors or identify the leakage source should actively search the environment for targets, using their observations to guide the decision making in an unknown environment (Murphy et al., 2008). Similarly, unmanned aerial vehicles (UAVs) or drones engaged in wildfire monitoring (Viseras et al., 2019; Industries, 2022) should identify the high risk zones without requiring a set of pre-defined waypoints which might not always be available. In such settings, each robot selects its actions *interactively* in an *online* manner, based on its previous observations in its surroundings. Sensing actions cover contiguous regions of the search space which the agent observes with its sensors, for example, camera used to take pictures of the surroundings or lidar to measure distances to obstacles. Sensors provide noisy observations which affect the agent's subsequent decisions over sensing actions. This motivates the need for appropriate abstractions in the robotics active search formulation, in order to design

practical and scalable algorithms for real world applications.

1.1.1 Multi-Agent Active Search

When deployed at scale, robotic systems for search and rescue or environment monitoring typically consist of a *team* of robots aiming to achieve a common task. In order to leverage the benefit of such a multi-agent system, agents communicate with each other to share observations or their beliefs about the environment or subsequent intended actions (Balch and Arkin, 1995). But communication channels can be unreliable (Ramanathan and Redi, 2002) or require inter-agent coordinations to ensure communication connectivity (Zavlanos et al., 2013). Moreover, reliance on synchronized communication can lead to increased idle wait time for some agents or even failure of the entire system in extreme cases (Yan et al., 2013a). Accounting for these challenges, we require our multi-agent system to be robust and functional even with limited or unreliable communication. This further implies the need for decentralization so that the agents do not rely on a central controller for communication or decision making. Therefore in this thesis, we will focus on decentralized multi-agent active search in which the agents communicate asynchronously when possible, but can independently recover targets in the environment even when communication is unreliable.

1.1.2 Foundations of problem formulation

Following is an outline for understanding the environment and agent definitions that we will consider in the rest of this thesis.

Search space. We will consider a bounded search region of length n_ℓ and width n_w, typically discretized into grid cells. The discretized search space can be represented as a matrix of 0s and 1s - the k ground truth target locations indicated by 1s, with 0s elsewhere. The flattened vector representation of the search space $\boldsymbol{\beta} \in \mathbb{R}^n$ is the ground truth search vector that should be recovered by agents over T time steps. Here, $n = n_\ell \times n_w$ is the size of the search space.

Sensing action. Agents observe their surroundings with sensors that typically provide noisy measurements from sensing actions. Most common sensing algorithms assume either agents can always arbitrarily sense in the entire search region (Donoho, 2006b; Braun et al., 2014) or sense point-wise at a time (Carpin et al., 2015; Rolf et al., 2020). Instead we will consider a more realistic setup where the agents can choose from *contiguous region sensing actions* of different widths or viewing directions for different fields of view which provide different amounts of information about the environment (Ma et al., 2017). This allows an agent to trade-off between a broader sensing action with a wider field of view that can result in a noisier measurement, versus a sensing action over a smaller region leading to a more precise observation. In other cases, agents may actively choose different viewing directions to reduce uncertainty about observations over a sensed region. This is an important consideration in the active search setting - some sensing actions can help quickly reduce the agent's uncertainty about the unknown environment with fewer actions, whereas others can help eliminate false positives and accurately localize the targets in the search space. Considering a discretized search space, a sensing action $\boldsymbol{x}_t$ can be represented by a

matrix of 0s and 1s (described in details later), where 1s indicate the regions in the agent's current field of view.

Sensing model. At each time step t, an agent $j \in \{1, \ldots, J\}$ executes a region sensing action $\boldsymbol{x}_t$ and receives a noisy measurement $\boldsymbol{y}_t$. Throughout, we will assume a linear sensing model:

$$\underbrace{\boldsymbol{y}_t}_{\text{observation}} = \underbrace{\boldsymbol{x}_t\boldsymbol{\beta}}_{\text{ground truth}} + \underbrace{\epsilon_t}_{\text{measurement noise}} \tag{1.1}$$

where ϵ_t is an additive i.i.d random white noise. $\boldsymbol{x}_t\boldsymbol{\beta}$ indicates the ground truth location of targets in the agent's current field of view for $\boldsymbol{x}_t$.

Communication model. Following Section 1.1.1 and Fig. 1.1, agents asynchronously broadcast their sensing actions and observations to their teammates. We do not assume any central or distributed controller to coordinate agents' actions and synchronize their observations. The set of past sensing actions and measurements available to any agent j at time t is denoted by $\mathbb{D}_t^j = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_{t'}, \boldsymbol{y}_{t'})\}$, $t' \leq t-1$, $|\mathbb{D}_t^j| < t$. Note that $\mathbb{D}_t^j$ includes all the measurements from sensing actions that the agent itself executed, as well as action and observation pairs received from its peers in the multi-agent system.

Objective. At time t, agent j selects the sensing action $\boldsymbol{x}_t^j$ that maximizes its objective $\mathcal{R}\left(\boldsymbol{x}_t^j | \mathbb{D}_t^j\right)$ conditioned on the past measurements $\mathbb{D}_t^j$. In each chapter, we will describe the individual problem setup and the corresponding decision making objective for multi-agent active search.

We will measure the performance of our multi-agent active search algorithms in terms of average recall, or the average number of targets that were fully recovered by all the agents over T steps. We call this metric the *full recovery rate*.

1.1.3 Applications

In this thesis, we will primarily focus on understanding the challenges in multi-agent active search applications in robotics and propose algorithms to address some of those. Our proposed algorithms are evaluated in simulation with synthetic environments designed following realistic modeling assumptions. Separately, concurrent work has successfully deployed some of these algorithms on drones and ground vehicles for test applications (Bakshi et al., 2023; Tabib et al., 2024). This validates the practicality and potential of this line of work and the importance of addressing the remaining limitations in future work.

1.2 Thesis Overview

Chapter 2 sets the foundation for the multi-agent active search framework and introduces Thompson sampling as a decentralized decision making algorithm which is repeatedly drawn upon in the rest of the thesis.

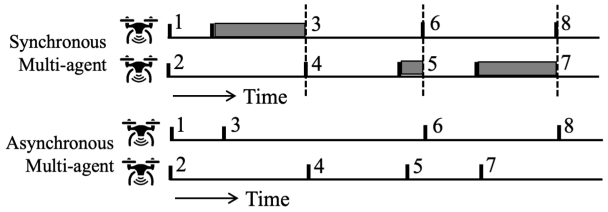


Figure 1.1: **Synchronous vs. asynchronous MAAS.** The small vertical lines indicate the start of the t-th task. Agents may differ in the time taken to complete their respective tasks. In a synchronous setup, agents have to wait for the entire team to complete their respective tasks at every time step (shown by the shaded grey blocks), leading to loss of processing time. In contrast, asynchronous MAAS ensures that agents can start their next assigned task as soon as they become available without any additional idle wait time.

Part I. In the first part of this thesis, we study the problem of multi-objective optimization (MOO) in multi-agent active search (MAAS). We particularly focus on cost-awareness in active search with autonomous aerial robots, eg. drones, where each agent incurs a sensing cost for executing its region sensing action and a travel cost for moving between successive sensing locations. In this setting, agents aim to fully recover all targets by optimizing over the number of measurements required as well as their associated cost.

Similar MOO problem settings have been addressed in prior work using scalarization (Best et al., 2020) and / or assuming myopic decision making (Ghods et al., 2021a). Moreover, the assumption of full observability in prior work (Chen and Liu, 2019) also simplifies the optimization problem. In contrast, we do not assume any knowledge about the number or position of targets in the search space. Therefore, the agent maintains a belief about the unknown environment using its noisy observations and must trade-off the expected reward of a sensing action under its current belief with the total cost that it will incur. Since the agent's action affects the obtained measurement, which subsequently influences the future sensing actions, so the MAAS agent would benefit from non-myopic multi-step lookahead over the evolution of its belief from possible actions *and* observations to balance the different optimization objectives.

In Banerjee et al. (2023a), we develop a cost-aware MAAS algorithm called CAST (Cost-aware Active Search of Sparse Targets) by combining principles from Monte Carlo Tree Search (for lookahead planning), Pareto-optimality (for multi-objective optimization) and Thompson sampling (for decentralized multi-agent decision making). Our experimental results demonstrate the need for cost-awareness in active search under different relative weights of cost components due to traveling and sensing. We show that CAST outperforms cost-agnostic and myopic baselines in multi-agent active search over unknown environments.

Part II. In the second part of this thesis, we will focus on designing MAAS algorithms that can adapt to different sources of uncertainty in the agent's observations. In Chapter 4, we consider observation uncertainty due to sensor noise both from object detectors and depth sensors when the targets are stationary. This is in contrast with prior work in signal processing (Malloy and Nowak, 2014a; Rajan et al., 2015; Marchant and Ramos, 2012a) and robotics (Miller et al., 2013; Zhou and Koltun, 2014; Chen et al., 2019a) which consider measurement noise due to only one of target detection uncertainty and target location

uncertainty respectively at a time. For example, sparse signal processing applications are concerned with addressing the uncertainty in the detection label, usually obtained from object detectors with some confidence score. But the perceived location of such detected targets is assumed to be accurate. On the other hand, robotics applications like SLAM (Simultaneous Localization and Mapping) (Huang et al., 2019) account for noisy depth sensors which can make an observed target appear to be located closer to or farther away from the agent than it actually is. In that case, the target detection uncertainty is abstracted away by thresholding the detection label to 0 / 1. But in reality, these two types of measurement noise almost never exist in isolation from one another in sensor observations. Moreover, to the best of our knowledge, prior work has primarily been concerned with state estimation given a set of measurements, but very rarely focused on the problem of observation uncertainty aware decision making to *actively* gather those measurements.

In Banerjee et al. (2023b) we develop an inference method called UnIK (Uncertainty-aware Inference with Kalman filter) that adapts the Kalman filter framework with a measurement noise covariance matrix constructed to account for both target detection and location uncertainty in the agent's measurements. The recursive nature of UnIK makes it computationally efficient and when augmented with a Thompson sampling (TS) decision making step, TS-UnIK is scalable to decentralized and asynchronous MAAS. We demonstrate in simulation the performance improvement obtained with UnIK and TS-UnIK compared to baselines that individually consider either target location or detection uncertainty.

In Chapter 5, we focus on the multi-agent active search and tracking setting where agents additionally face observation uncertainty due to dynamic targets. Prior work in target tracking algorithms assumes that agents continuously follow the targets being tracked (Dames et al., 2017; Papaioannou et al., 2020), thereby observing their location over successive time steps. Instead we consider the more realistic setting when there are fewer agents than targets, so that tracking by continuous coverage is not possible. We argue that it would be more practical if the agents were to adaptively switch between two modes - i.e. searching the environment for new or previously undetected targets and tracking the already detected targets by adaptively sensing to (re-)localize them in the search space. In Banerjee and Schneider (2024), we extend the framework of Probability Hypothesis Density (PHD) filters by proposing two Thompson sampling approaches that leverage the PHD posterior belief for multi-agent active search and tracking with non-stationary targets.

Part III. In the third part of this thesis, we shift our attention to generative sampling based lookahead decision making for active search. Motivated by the success of diffusion models for sequence modeling and planning (Song et al., 2021b; Janner et al., 2022), we propose a reward-gradient guided diffusion algorithm for cost-aware multi-agent active search. We outline some of the challenges involved in applying generative models for similar settings with partial observability and environment stochasticity. In particular, we highlight the optimism bias encountered during diffusion over the joint state-action space and appropriately modify our modeling assumptions to mitigate it. Moreover, we address the training sample inefficiency in higher dimensions by adopting an appropriate inductive bias in the network architecture. We also propose a simple Thompson sampling based framework for decentralized and asynchronous multi-agent active search with single-agent diffusion models. In simulation, our experiments demonstrate the advantage of non-myopic decision making in active search as well as improved (or competitive) inference-time cost-awareness compared

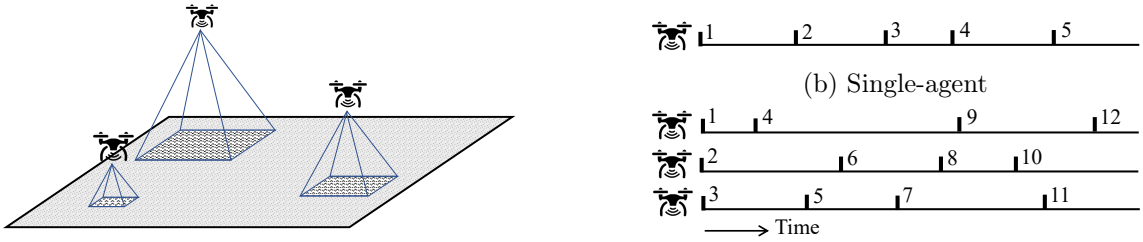
to the search based cost optimization in CAST (Banerjee et al., 2023a).

Finally, we conclude in Chapter 7 with insights from the completed work in this thesis and some outlines for future work in this direction.

2 | Thompson Sampling for Decentralized and Asynchronous Multi-Agent Active Search

2.1 Introduction

Agents (robots) deployed in active search applications like detecting gas leaks, pollution sources or search and rescue missions (Ma et al., 2017; Rolf et al., 2020; Flaspohler et al., 2019) have to adaptively decide how to act in their environments to successfully complete their tasks. Agents typically maintain a representation or model of the environment which can be learned and updated over time and which conditions the agents' behavior, especially in unknown or unstructured surroundings. Equally significant, and our focus in this chapter, is the decision making algorithm which selects the agent's actions conditioned on the current agent or environment state. In the active decision making domain, most of the existing algorithms are deterministic in nature and developed for single-agent settings, therefore not extendable to multi-agent scenarios. As an example, Braun et al. (2015) uses information greedy approaches to decide on best sensing actions for its agent. If we were to use multiple agents for this info-greedy method, all agents would make the same exact decision at each time step, leading to redundant observations and resource (for example, battery power) wastage. For other active learning algorithms that are extendable to multi agent scenarios, they usually need a central control system to coordinate the sensing actions of all agents (Azimi et al., 2012; Gu et al., 2014). Unfortunately, central coordination of agents is often impractical in certain applications of surveillance, search and rescue or localization and tracking (Sabattini et al., 2013). This is because in these applications connectivity maintenance is especially difficult (Yan et al., 2013b; Robin and Lacroix, 2016). Moving in an unknown or cluttered environment, it is very likely for robots to get trapped and temporarily lose their connection to the center (Sabattini et al., 2013). As a result, a central controller that expects synchronicity from all robots at all times is not feasible as any agent failure or communication delay could disrupt the entire process (Queralta et al., 2020; Best et al., 2019; Lauri et al., 2020; Murphy, 2004a; Feddema et al., 2002). To clarify, there is still communication between agents to share information, otherwise they are just independent actors and would not be a team. In this chapter, we are therefore primarily interested in a decentralized multi-agent decision making setting where agents can communicate asynchronously without a central controller. Our discussion on this topic in the following sections will form the basis of our active decision making framework in the rest of this thesis.



(a) (b) Single-agent (c) Multi-agent (asynchronous)

Figure 2.1: (a) **An illustration of multi-agent active search.** Multiple aerial robots are sensing an area looking for targets. Agents are free to move in all directions. If an agent moves farther from the region, it can cover a larger portion in one lower-resolution observation. Moving closer to the region covers a smaller region in one higher-resolution observation. (b), (c) **Single vs. multi-agent.** Here, the small vertical lines indicate the start of t'th task. In single-agent, tasks start sequentially. In asynchronous multi-agent, task t can start before all previous $t-1$ tasks are finished.

2.2 Problem Formulation

Figure 2.1a illustrates a multi agent active search problem for a two-dimensional environment. Our goal is to efficiently search for targets in an unknown environment by actively deciding sensing actions given all the observations thus far.

We assume k targets are sparsely distributed in the environment. In a two-dimensional space, we can represent the search region with a sparse matrix $\mathbf{B} \in \mathbb{R}^{n_\ell \times n_w}$, where n_ℓ is the length dimension and n_w is the width dimension. Agents have no knowledge of the true prior distribution of $\mathbf{B}$ other than knowing it is sparse. Defining $\boldsymbol{\beta} \in \mathbb{R}^n$ as a flattened (vectorized) version of matrix $\mathbf{B}$ with $n = n_\ell \times n_w$, we will refer to $\boldsymbol{\beta}$ as the true search vector. The agent's task is to recover this search vector $\boldsymbol{\beta}$ with minimum measurements.

Another consideration of our multi-agent active search setup is a realistic assumption on the sensing actions called region sensing, initially introduced by Ma et al. (2017). In this chapter, we will consider aerial agents, for example a drone equipped with a downward facing camera over its field of view. We assume that each agent senses an average value of a contiguous region (block) of the space at each time step. The size of the sensing block models the distance of the agent from the region. The agent's sensing model is given by

$$y_t = \mathbf{x}_t^{\mathrm{T}} \boldsymbol{\beta} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2),\ t = 1, ..., T. \tag{2.1}$$

Here, y_t is the observation and vector $\mathbf{x}_t \in \mathbb{R}^n$ is the sensing action at time step t. We call the set of $(\mathbf{x}_t, y_t)$ the measurement at time step t. In the d-dimensional search space, our sensing action will be a d-dimensional contiguous rectangle (region) with weights w_t inside the rectangle (i.e. the agent's field of view) and zeros outside. As an example, if $d = 1$, the sensing action becomes $\mathbf{x}_t = [0, ..., 0, w_t, ..., w_t, 0, ..., 0]^{\mathrm{T}}$. This constraint models a robot sensing a region of the search space as illustrated in Figure 2.1a.

We also model noise in the observations in accordance to this distance. Specifically, we dedicate a fixed amount of power to each sensing action by letting $\|\mathbf{x}_t\|_2 = 1$. This ensures that sensing a larger contiguous region at a farther distance from the region (i.e. larger field of view) inflicts a larger noise value on the resulting observation.

Remark 1. By dedicating a fixed amount of power to each sensing action, we are modeling noise as a function of distance from the region. In particular, by standing at a farther distance from the area, an agent can cover a larger region in an observation. Spreading the fixed amount of sensing power over this larger region would result in larger noise on the observation (a lower resolution observation). Similarly, sensing a smaller region at a closer distance to the environment would model smaller noise (higher resolution observation). Fig. 2.1a illustrates practicality of this model.

Communication Setup. In order to achieve the objective above with multiple agents, we need to first describe our communication setup which is motivated by real outdoor multi-aerial robot systems in field tests. Despite unreliability in unknown environments, communication becomes available sometimes and we want to take advantage of it when possible. That leads to the following constraints for our algorithm:

- Agents share their past actions and observations when possible.
- There can be no requirement that the set of available past measurements remains consistent across agents since communication problems can prevent it.
- here can be no part of the algorithm where an agent must wait for communication from its teammates before acting since this wait could be arbitrarily long and thus cause a loss of valuable sensing time.

We are now ready to describe the multi-agent setting. To actively locate targets, at each time step $t \leq T$, an agent j chooses a sensing action $\mathbf{x}_t^j$ given all the available measurements thus far in the set $\mathbf{D}_{t-1}^j$. The superscript j indicates the agent index. For a single agent this procedure is sequential as in Figure 2.1b where at time step t the agent uses all previous sequential measurements $\mathbf{D}_{t-1}^1 = \{(\mathbf{x}_{t'}, y_{t'}) | \, t' = \{1, .., t-1\}\}$ to make a decision. In this thesis, however, we are interested in an asynchronous parallel approach with multiple agents making data-collection decisions in a decentralized manner, as shown in Figure 2.1c. Here asynchronicity means that agents don't wait on results from other agents; instead, an agent starts a new query immediately after its previous data acquisition is completed using all the measurements available thus far. For example, in Figure 2.1c, the second agent queries $t = 6$'th action before tasks 4 and 5 are completed using available measurements $\mathbf{D}_6^2 = \{(\mathbf{x}_{t'}, y_{t'}) | \, t' = \{1, 2, 3\}\}$.

For easier computations, we can define a compact sensing model with all the available measurements in $\mathbf{D}_{t-1}^j$ for agent j. For example for sequential $\mathbf{D}_{t-1}^1$, by defining $\mathbf{y} = [y_1, ..., y_{t-1}]^{\mathrm{T}}$, $\mathbf{X} = [\mathbf{x}_1^{\mathrm{T}}, ..., \mathbf{x}_{t-1}^{\mathrm{T}}]^{\mathrm{T}}$ we can write the model in (2.1) as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_{t-1}). \tag{2.2}$$

2.3 Related Work

A prominent approach to estimating sparse signals is compressive sensing (CS) (Candès et al., 2006; Donoho, 2006a). There has been a large number of work on adaptive CS that enables the ability to make online and adaptive measurements to estimate sparse signals and thus is applicable to active search problems (Braun et al., 2015; Haupt et al., 2009a,b; Davenport and Arias-Castro, 2012; Malloy and Nowak, 2014b). Unfortunately, such adaptive CS methods are sequential and therefore not extendable to multi agent scenarios. Furthermore, CS algorithms in general assume that every measurement matrix can

sense the entire environment with arbitrary coefficients which is not a practical assumption for active search problems with region sensing constraints.

Another area of work are multi-armed bandits. Abbasi-Yadkori et al. (2012) and Carpentier and Munos (2012) propose multi-armed bandit algorithms that include a sparsity assumption on their hyperparameter. However, they do not focus on estimating the sparse parameter since the bandit algorithms optimize a different reward function. There have been other Bayesian Optimization (BO) and active learning methods proposed for active search. Marchant and Ramos (2012b) uses BO to develop a spatial mapping of a region whereas we are interested in locating targeted signals. Carpin et al. (2015) uses BO for localization of single wireless devices but only focuses on point sensing actions. Ma et al. (2017); Rajan et al. (2015); Jedynak et al. (2012) aim at locating targets by optimizing some notion of Shannon information. Unfortunately, all of the aforementioned active learning algorithms are developed for single agent applications, and except for Ma et al. (2017), they mostly lack any realistic assumptions on sensing actions.

In multi-agent active learning, algorithms in prior work generally require a central planner to optimize a batch of actions for all agents at each time step and therefore are not applicable to our problem setting (Azimi et al., 2012; Gu et al., 2014; Azimi et al., 2010). Another multi-agent area of work are mobile sensor networks (MSN) (Nguyen, 2019; Chen et al., 2019b; La et al., 2014) where multiple mobile sensors/agents reconstruct a scalar map of sensory values in an entire area. MSNs typically consider some form of region sensing assumption on their actions, however, they generally have a constricting sensor network with strict communication patterns which differentiates them from our applications.

In robotics research, methods that deal with active search generally aim at autonomously building topological (identify obstacles and clearways) and/or spatial maps of a region. Our active search problem differs from topological mapping techniques such as SLAM (Leonard and Durrant-Whyte, 1991; Huang et al., 2019) and can be most closely related to spatial mapping. For example, Rolf et al. (2018) identifies strong signals in environments with background information using trajectory planning with confidence intervals; but, unlike our problem setting, their algorithm is developed for a single agent performing point sensing observations. In the area of robotics information gathering, there has been more attention towards the need for decentralized solutions recently (Queralta et al., 2020; Zhang et al., 2021). However, existing methods for decentralized multi-agent systems either assume reliable communication requirements to share future plans (Best et al., 2019; Dames et al., 2017; Li and Duan, 2017) or assume centralized sharing of observations following decentralized execution (Lauri et al., 2020; Lowe et al., 2017). Instead, following our discussion so far, our algorithms should benefit from observation sharing when it occurs, but never depend on communication for coordination.

2.3.1 Naïve approach and challenges

In our multi-agent active search setup, agents communicate by broadcasting their measurements to their teammates. Therefore a naïve decentralized decision making algorithm could involve each agent selecting an action conditioned on its measurement set, which includes its own observations as well as those received from other agents. Following prior work, each agent may use an information-greedy objective for action selection. While this would enable decentralized and asynchronous active search with multi-agent teams, it still relies on a deterministic decision making objective. This means that agents which perhaps operate

at the same frequency and have identical measurement sets at a given time t would end up choosing the exact same sensing action in the search space. Not only would this lead to redundant observations, but agents have the possibility to collide as they compete for the same sensing locations. Instead, our main insight is to introduce stochasticity in the decentralized decision making setup so that agents implicitly select distributed sensing locations covering the entire search space. Since our sensing model also accounts for observation noise, we expect agents to collect multiple observations (distributed over the entire task duration) at the same sensing location to reduce the uncertainty about presence of targets in a particular region. This implies that the agents' decision making algorithm should be able to trade-off exploration (sensing locations not previously visited) with exploitation (repeated sensing at already visited location) in the search space. Moreover, recall that targets are sparsely distributed, further supporting the need for effective exlpore-exploit trade-off. While there have been many sparse recovery algorithms proposed in the literature, to the best of our knowledge there is no algorithm proposed that develops sparse estimators for active learning methods along with multi-agent structure and region sensing assumptions. In this chapter, we show how sparsity in its nature limits the exploration factor in active learning methods and how a practical region sensing assumption exacerbates this situation. Next we describe our proposed approach to strategically address such region sensing assumptions to successfully recover sparse signals.

2.4 Thompson sampling in Active Decision Making

Thompson Sampling is an exploration-exploitation algorithm originally introduced for clinical trials by Thompson (1933) and later rediscovered for multi-armed bandits (Wyatt, 1998; Strens, 2000; Russo et al., 2018). The key idea of TS is to balance exploration with exploitation by maximizing the expected reward of its next action assuming that a sample from the posterior is the true state of the world (Russo et al., 2018). This feature makes TS an excellent candidate for our asynchronous multi-agent setup. Essentially, by using TS's posterior samples in our reward function, we enable a calculated randomness in each agent's reward function. As a result, multiple agents can take independent samples and therefore solve for different reward values that equally contribute to the overall goal. Kandasamy et al. (2018) used this feature of TS for Bayesian Optimization (BO) to develop an asynchronous yet *centralized* parallel setting. We instead propose to develop TS in a decentralized and asynchronous setting where each agent independently makes decisions given the measurements available to it, i.e. $\mathbf{D}_t^j$.

Because TS was originally proposed for bandit problems, one might question its ability on active search and assume it might keep exploiting the same target. However, we are here interested in an adaptation of TS to parameter learning which is in fact perfect for active search. Having attracted a lot of attention in the past decade, TS has been successfully adapted to a variety of online learning problems (Russo et al., 2018). Our active search problem falls in the category of parameter estimation in active learning as developed by Kandasamy et al. (2019) with the name Myopic Posterior Sampling(MPS). Similar to MPS, our goal is to actively learn (estimate) parameter $\boldsymbol{\beta}$ by taking as few measurements as possible. Since the goal of MPS is to learn parameter $\boldsymbol{\beta}$, its reward function is designed to keep exploring the space as long as there are unexplored (or loosely explored) locations in the parameter space (i.e. it will not get stuck exploiting). We will next derive MPS for

an asynchronous multi-agent setting. For the sake of similarity, we will use TS to refer to MPS.

2.4.1 Decentralized Multi-Agent Thompson Sampling

First, we review TS for active learning and then develop it to a decentralized asynchronous (async.) multi-agent setting. We start with the single agent setting as introduced by Kandasamy et al. (2019). We are interested in recovering the n-dimensional vector $\boldsymbol{\beta} \sim p_0$.

An agent actively queries action $\mathbf{x}_t$ and observes outcome y_t where the likelihood $p(y_t|\mathbf{x}_t, \boldsymbol{\beta})$ is known. To query the best action, we maximize a reward function $\lambda(\boldsymbol{\beta}^*, \mathbf{D}_t^1)$ where $\boldsymbol{\beta}^*$ is our belief of the true $\boldsymbol{\beta}$. In other words, an agent follows a myopic policy which selects action $\mathbf{x}_t$ that maximizes the expected reward of timestep t, i.e.

$$\mathbf{x}_t = \underset{\mathbf{x}}{\operatorname{argmax}}\, \lambda^+(\boldsymbol{\beta}^*, \mathbf{D}_t^1, \mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmax}}\, \mathbb{E}_{y|\mathbf{x},\boldsymbol{\beta}^*}[\lambda(\boldsymbol{\beta}^*, \mathbf{D}_t^1 \cup (\mathbf{x}, y))]. \tag{2.3}$$

Here the reward that matches the ground truth value of selecting an action would be the one that has access to the true value of $\boldsymbol{\beta}$ i.e. for $\lambda^+(\boldsymbol{\beta}, \mathbf{D}_t^1, \mathbf{x})$ when $\boldsymbol{\beta}^* = \boldsymbol{\beta}$. Not knowing the true search vector $\boldsymbol{\beta}$, TS proposes to sample it from the current posterior distribution over $\boldsymbol{\beta}$ given the measurement set $\mathbf{D}_t^1$, i.e. $\boldsymbol{\beta}^* \sim p(\boldsymbol{\beta}|\mathbf{D}_t^1)$. Then the TS policy selects the sensing action $\mathbf{x}_t$ that maximizes Eq. (2.3) with the sampled $\boldsymbol{\beta}^*$.

In the multi-agent setting, consider J agents tasked with active discovery over T measurements in an environment. Suppose agent j executes its latest sensing action, communicates with its teammates and is ready to choose the t-th action. Using its measurement set $\mathbf{D}_t^j$ ($|\mathbf{D}_t^j| \leq t-1$), it updates the posterior and draws a sample (*posterior sampling*), selects its next sensing action to maximize the reward (*design*), evaluates its action and shares the next observation with other agents. Algorithm 1 summarizes this process.

Algorithm 1 Asynchronous Multi-Agent Thompson Sampling

Assume: prior $\boldsymbol{\beta} \sim p_0$ and likelihood $p(y|\mathbf{x}, \boldsymbol{\beta})$
For $t = 1, ..., T$
  Wait for an agent to finish; **For** the free agent j:
    Sample $\boldsymbol{\beta}^\star \sim p(\boldsymbol{\beta}|\mathbf{D}_t^j)$ — *Posterior Sampling*
    Select $\mathbf{x}_t^j = \operatorname{argmax}_{\mathbf{x}} \lambda^+(\boldsymbol{\beta}^\star, \mathbf{D}_{t-1}^j, \mathbf{x})$ — *Design*
    Observe y_t^j given action $\mathbf{x}_t^j$
    Update & share measurements $\mathbf{D}_{t+1}^j = \mathbf{D}_t^j \cup (\mathbf{x}_t^j, y_t^j)$

2.4.2 Thompson Sampling with Sparsity

To perform search and rescue, traditionally people have used coverage planning methods with exhaustive search (Lin and Goodrich, 2009; Chien et al., 2010; Ryan and Hedrick, 2005). However, with the availability of high and low resolution observation points, an optimized active search method can locate targets faster than exhaustive search in terms of number of observations (refer Section 2.6). Such faster recovery is achievable due to the concept of sparse signal recovery (compressive sensing) which says that we can recover a sparse signal with length n by taking less than n low or high resolution measurements (Candès et al., 2006; Donoho, 2006a). By using sparsity as the prior information for TS, we

can create the right balance between exploring larger regions with low resolution and then exploiting the ones we suspect of including a target with a closer look (higher resolution observation). We will next develop TS in Algorithm 1 for our active search problem using a sparse prior.

We start by first establishing the prior $p(\boldsymbol{\beta})$ and likelihood distribution $p(y_t|\mathbf{x}_t, \boldsymbol{\beta})$. As for the prior, our knowledge is limited to the presence of sparsity. First, let us assume $\boldsymbol{\beta}$ has a Laplace distribution with independent entries and a tunable parameter b, i.e. $p(\boldsymbol{\beta}) = \frac{1}{(2b)^n} \exp(-\frac{\|\boldsymbol{\beta}\|_1}{b})$. Laplace distribution translates to an ℓ_1-norm regularization term in the cost function which has been shown to introduce sparsity into the estimator (Williams, 1995; Tibshirani, 1996; Chen et al., 2001). For the likelihood distribution, the sensing model in Eq. (2.1) gives $p(y|\mathbf{X}, \boldsymbol{\beta}) = \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}_{t-1})$.

Design. Following Algorithm 1, agent j computes the expected reward $\lambda^+(\boldsymbol{\beta}^*, \mathbf{D}_t^j, \mathbf{x})$ and optimizes it for the next sensing action $\mathbf{x}_t^j$. Since our goal is to recover the unknown true search vector $\boldsymbol{\beta}$, we use the reward function

$$\lambda(\boldsymbol{\beta}^*, \mathbf{D}_t^j \cup (\mathbf{x}, y)) = -\|\boldsymbol{\beta}^* - \hat{\boldsymbol{\beta}}(\mathbf{D}_t^j \cup (\mathbf{x}, y))\|_2^2. \tag{2.4}$$

In order to compute the expected reward in Eq. (2.4), we need to design an estimator $\hat{\boldsymbol{\beta}}(\mathbf{D}_t^j \cup (\mathbf{x}, y))$ whose expectation we can compute. Unfortunately, for many of the well-known thresholding or iterative algorihms proposed in the sparse recovery literature (Pati et al., 1993; Needell and Tropp, 2010; Blumensath and Davies, 2009; Daubechies et al., 2004; Maleki, 2011), computing Eq. (2.4) is intractable. Instead we propose using the maximum a posteriori (MAP) estimate of the posterior $p(\boldsymbol{\beta}|\mathbf{D}_t^j)$ computed using an expectation-maximization approach. This allows us to compute a closed form analytical expression for Eq. (2.4) [1], but this preliminary implementation of a sparse estimator guided Thompson sampling for active search was unsuccessful and its performance was observed to be comparable to a point sensing algorithm that exhaustively searches all locations one at a time. Next we provide some intuition for this failure mode of TS in active decision making and then describe our approach to mitigating this behavior.

Failure Mode of TS. We can associate this poor performance of our preliminary approach described above with one of the failure modes of TS discussed in Sec. 8.2 of the tutorial by Russo et al. (2018). According to this tutorial, TS faces a dilemma when solving certain kinds of active learning problems. One such scenario includes problems that require a careful assessment of information gain. In general, by optimizing the expected reward, TS always restricts its actions to those that have a chance of being optimal which in our case are sparse sensing actions restricted further by the region sensing constraint. However, in active learning problems such as ours, suboptimal actions (i.e. non-sparse sensing actions) can carry additional information regarding the parameter of interest.

We provide the following example to better understand the challenge in applying TS with a sparse estimator in our problem formulation of active search. Let us assume there is no noise in the sensing model (Eq. (2.1)) i.e. $\epsilon_t = 0$ and that there is only one non-zero element ($k = 1$). Under such conditions, the active search task amounts to finding the location $\tilde{i}$ of this non-zero element in the search vector $\boldsymbol{\beta}$. Thus, for every action $\mathbf{x}_t$, the observation $y_t = \mathbf{x}_t^{\mathrm{T}}\boldsymbol{\beta}$ is non-zero if $\tilde{i}$ is in the support of $\mathbf{x}_t$ and it is zero otherwise. Clearly, a binary search agent can locate $\tilde{i}$ in $\log(n)$ steps. TS, however, in each timestep selects the

[1] For complete mathematical derivation of this posterior update algorithm, please refer to Sec 3.2 in Ghods et al. (2021b). The details of this approach are omitted here for brevity.

sensing action $\mathbf{x}_t$ which would maximize Eq. (2.4) by recovering the support of the sparse sample $\boldsymbol{\beta}^*$ in the estimate $\hat{\boldsymbol{\beta}}(\mathbf{D}_t^1 \cup (\mathbf{x}, \mathbf{x}^{\mathrm{T}}\boldsymbol{\beta}^*))$. So, unless $\boldsymbol{\beta}^*$ is the true $\boldsymbol{\beta}$, observation y_t from sensing action $\mathbf{x}_t$ would be zero and TS will only manage to eliminate support of $\boldsymbol{\beta}^*$ from the list of possibly correct supports. Therefore, TS ends up eliminating wrong supports one location at a time which explains its on par performance to an exhaustive point sensing policy. Similarly, for $k > 1$ TS is always limited to picking sensing actions $\mathbf{x}_t$ that are in the support of the sparse estimates of the samples. Adding the region sensing constraint will only aggravate this problem by further shrinking this support set. Next, we address these limitations and propose our novel multi-agent active search algorithm called SPATS for Sparse Parallel Asynchronous Thompson Sampling.

2.5 SPATS: Sparse Parallel Asynchronous Thompson Sampling

Per our discussion in Section 2.4.2, introducing sparsity with Laplace prior into TS algorithm limited its ability to explore queries. With this in mind, one might conclude that choosing non-sparse samples in the posterior sampling stage of Algorithm 1 should solve this problem. However, this strategy will still face the failure mode of TS because it is the sparse estimator in the design stage that is limiting the feasible sensing actions. The next logical solution would then be to make both the estimator and posterior sampling procedures non-sparse. Even though with this strategy we will avoid the failure mode of TS, without taking advantage of the prior information about sparsity, the resulting non-sparse TS will be performing no better than exhaustively searching the entire space. To overcome this issue, we propose making an assumption on the prior distribution of both the sampling and estimation procedures that the neighbouring entries of the sparse vector $\boldsymbol{\beta}$ are temporally correlated, i.e. $\boldsymbol{\beta}$ is block sparse. Such temporal correlation creates the most compatible results to the region sensing constraint which only approves sensing actions with a single non-zero block of sensors. Furthermore, we expect block sparsity to introduce exploration ability while also keeping sparsity a useful information in the recovery process. In particular, by gradually reducing the length of the blocks from a large value, we gently trade exploration with exploitation capability over time.

2.5.1 Belief Representation

Borrowing ideas from a block sparse Bayesian framework introduced in Zhang and Rao (2011), we use a block sparse prior

$$p(\boldsymbol{\beta}) = \mathcal{N}(\mathbf{0}_{n\times 1}, \boldsymbol{\Sigma}_0)$$

, where $\boldsymbol{\Sigma}_0 = \mathrm{diag}\left([\gamma_1\mathbf{B}_1, ..., \gamma_M\mathbf{B}_M]\right)$, with γ_m and $\mathbf{B}_m \in \mathbb{R}^{L\times L}$ $(m = 1, ..., M)$ as hyperparameters. γ_m controls the sparsity of each block as is the case in sparse Bayesian learning methods (Tipping, 2001; Wipf and Rao, 2004), i.e. when $\gamma_m = 0$, the corresponding block m is zero. L is the length of the blocks that we will gradually reduce in the TS process. To avoid overfitting while estimating these hyperparameters, Zhang and Rao (2013) suggests one matrix $\mathbf{B}$ to model all block covariances, namely $\boldsymbol{\Sigma}_0 = \mathrm{diag}(\boldsymbol{\gamma}) \otimes \mathbf{B}$, where, $\boldsymbol{\gamma}$ is the vector containing all elements of γ_m for $m = 1, ..., M$. As before, $\boldsymbol{\gamma} \in \mathbb{R}^M$ and $\mathbf{B} \in \mathbb{R}^{L\times L}$ $(M = n/L)$ are hyperparameters.

For any agent j, the posterior with a Gaussian likelihood and Gaussian prior becomes:

$$p(\boldsymbol{\beta}|\mathbf{D}_t^j, \boldsymbol{\gamma}, \mathbf{B}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\beta}}(\boldsymbol{\gamma}, \mathbf{B}), \boldsymbol{\Sigma}_{\boldsymbol{\beta}}(\boldsymbol{\gamma}, \mathbf{B})), \tag{2.5}$$

with

$$\begin{aligned}\boldsymbol{\Sigma}_{\boldsymbol{\beta}}(\boldsymbol{\gamma}, \mathbf{B}) &= ((\boldsymbol{\Sigma}_0)^{-1} + \frac{1}{\sigma^2}\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1},\\ \boldsymbol{\mu}_{\boldsymbol{\beta}}(\boldsymbol{\gamma}, \mathbf{B}) &= \frac{1}{\sigma^2}\boldsymbol{\Sigma}_{\boldsymbol{\beta}}(\boldsymbol{\gamma}, \mathbf{B})\mathbf{X}^{\mathrm{T}}\mathbf{y}.\end{aligned}$$

Using Expectation-Maximization proposed by Zhang and Rao (2011), we can estimate the hyperparameters $\boldsymbol{\gamma}$ and $\mathbf{B}$ for $p = 1, \ldots, P$ iterations as follows.

$$\begin{aligned}\textbf{E-step}: \boldsymbol{\mu}_{\boldsymbol{\beta}}^{(p)} &= \boldsymbol{\mu}_{\boldsymbol{\beta}}(\boldsymbol{\gamma}^{(p-1)}, \mathbf{B}^{(p-1)}),\\ \boldsymbol{\Sigma}_{\boldsymbol{\beta}}^{(p)} &= \boldsymbol{\Sigma}_{\boldsymbol{\beta}}(\boldsymbol{\gamma}^{(p-1)}, \mathbf{B}^{(p-1)})\\ \textbf{M-step}: \gamma_m^{(p)} &= \frac{\mathrm{tr}(\mathbf{B}^{-1}(\boldsymbol{\Sigma}_{\boldsymbol{\beta}}^m + \boldsymbol{\mu}_{\boldsymbol{\beta}}^m(\boldsymbol{\mu}_{\boldsymbol{\beta}}^m)^{\mathrm{T}}))}{L}, \quad (m = 1, \ldots, M)\\ \mathbf{B}^{(p)} &= \frac{1}{M}\sum_{m=1}^{M}\frac{\Sigma_{\boldsymbol{\beta}}^m + \boldsymbol{\mu}_{\boldsymbol{\beta}}^m(\boldsymbol{\mu}_{\boldsymbol{\beta}}^m)^{\mathrm{T}}}{\gamma_m}\end{aligned} \tag{2.6}$$

with $\boldsymbol{\mu}_{\boldsymbol{\beta}}^m = \mu_{\boldsymbol{\beta}}^{(p)}[(m-1)\times L : m\times L]$ and $\boldsymbol{\Sigma}_{\boldsymbol{\beta}}^m = \boldsymbol{\Sigma}_{\boldsymbol{\beta}}^{(p)}[(m-1)\times L : m\times L, (m-1)\times L : m\times L]$.

2.5.2 Decision making in SPATS

Next, we will compute the objective in Eq. (2.4) in expectation over the observation likelihood following Section 2.5.1. For the estimator $\hat{\boldsymbol{\beta}}(\mathbf{D}_t^j)$ of agent j, we use the MAP estimate $\boldsymbol{\mu}_{\boldsymbol{\beta}}^{(P)}$. Specifically, we have

$$\hat{\boldsymbol{\beta}}(\mathbf{D}_t^j \cup (\mathbf{x}, y)) = \underbrace{(\sigma^2{\boldsymbol{\Sigma}_0}^{-1} + \begin{bmatrix}\mathbf{X}^{\mathrm{T}} & \mathbf{x}\end{bmatrix}\begin{bmatrix}\mathbf{X}\\ \mathbf{x}^{\mathrm{T}}\end{bmatrix})^{-1}}_{\mathbf{Q}}\begin{bmatrix}\mathbf{X}^{\mathrm{T}} & \mathbf{x}\end{bmatrix}\begin{bmatrix}\mathbf{y}\\ y\end{bmatrix} \tag{2.7}$$

with $\boldsymbol{\Sigma}_0 = \mathrm{diag}(\boldsymbol{\gamma}^{(P)}) \otimes \mathbf{B}^{(P)}$. Now, following Eq. (2.3), Eq. (2.4), we can derive the expected reward as follows:

$$\begin{aligned}\lambda^+(\boldsymbol{\beta}^*, \mathbf{D}_t^j, \mathbf{x}) &= \mathbb{E}_{y|\mathbf{x},\boldsymbol{\beta}^*}[-\|\boldsymbol{\beta}^* - \hat{\boldsymbol{\beta}}(\mathbf{D}_t^j \cup (\mathbf{x}, y))\|_2^2]\\ &= -\mathbb{E}_{y|\mathbf{x},\boldsymbol{\beta}^*}[\|\mathbf{Q}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \mathbf{Q}\mathbf{x}y - \boldsymbol{\beta}^*\|_2^2]\\ &= -\|\mathbf{Q}\mathbf{X}^{\mathrm{T}}\mathbf{y} - \boldsymbol{\beta}^*\|_2^2 - \|\mathbf{Q}\mathbf{x}\|_2^2\mathbb{E}_{y|\mathbf{x},\boldsymbol{\beta}^*}[y^2] - 2(\mathbf{Q}\mathbf{X}^{\mathrm{T}}\mathbf{y} - \boldsymbol{\beta}^*)^{\mathrm{T}}\mathbf{Q}\mathbf{x}\mathbb{E}_{y|\mathbf{x},\boldsymbol{\beta}^*}[y]\\ &= -\|\mathbf{Q}\mathbf{X}^{\mathrm{T}}\mathbf{y} - \boldsymbol{\beta}^*\|_2^2 - \|\mathbf{Q}\mathbf{x}\|_2^2(\sigma^2 + (\mathbf{x}^T\boldsymbol{\beta}^*)^2) - 2(\mathbf{Q}\mathbf{X}^{\mathrm{T}}\mathbf{y} - \boldsymbol{\beta}^*)^{\mathrm{T}}\mathbf{Q}\mathbf{x}\mathbf{x}^{\mathrm{T}}\boldsymbol{\beta}^*\end{aligned} \tag{2.8}$$

For best empirical performance, we need to initialize parameter $\boldsymbol{\Sigma}_0$ relative to signal power. This concludes our specification of the *Posterior sampling* and *Design* stages from Algorithm 1. Algorithm 2 describes our proposed algorithm called SPATS for asynchronous and decentralized multi-agent active search.

Algorithm 2 SPATS

Assume: Sensing model (2.2); sparse signal β; J agents; block length L
Set: $\mathbf{D}_0^j = \varnothing$; $L = n/J$; $\gamma_m = 1$; $\mathbf{B}$: random highly correlated covariance matrix
For $t = 1, ..., T$
  Wait for an agent to finish; **For** the freed agent j:
    Sample $\boldsymbol{\beta}^\star \sim p(\boldsymbol{\beta}|\mathbf{D}_t^j, \boldsymbol{\gamma}, \mathbf{B})$ from Eq. (2.5) in Section 2.5.1
    Select $\mathbf{x}_t = \text{argmax}_{\mathbf{x}} \lambda^+(\boldsymbol{\beta}^\star, \mathbf{D}_t^j, \mathbf{x})$ using Eq. (2.8) in Section 2.5.2
    Observe y_t given action $\mathbf{x}_t$
    Update & share measurements $\mathbf{D}_{t+1}^j = \mathbf{D}_t^j \cup (\mathbf{x}_t, y_t)$
    Update hyper parameters $\boldsymbol{\gamma}$ and $\mathbf{B}$ using EM algorithm in Eq. (2.6) in Section 2.5.1
    if $t \,\%\, J = 0$ **then** $L = L/2$

2.5.3 Theoretical Bounds for a Sparse Model

We now provide theoretical analysis testifying to the benefits of SPATS. SPATS has two aspects that distinguish it from a naïve TS developed for sparse signals. First, the posterior belief update step in SPATS utilizes a block sparse prior with varying block length. Second, SPATS builds on parallel asynchronous TS for multi-agent active search. Next we introduce two theorems to investigate the benefits of each aspect separately. In this chapter, we will explain the main takeaway from each in context of the subsequent work presented in this thesis. Please refer to Ghods et al. (2021b) for complete proofs of both the theorems.

First in Theorem 1, for a sparse model with single agent setting we compute and compare upper bounds on the expected regret of two TS algorithms with a 1-sparse and a 1-block sparse prior with one nonzero block. The 1-block sparse prior closely imitates SPATS's performance with a region sensing assumption.

Theorem 1. *Consider an active search problem with a 1-sparse true parameter $\boldsymbol{\beta} \in \mathbb{R}^n$ and reward function $\mathcal{R}(\mathbf{x}, \boldsymbol{\beta}) = (\mathbf{x}^{\boldsymbol{\beta}})^2$ for action $\mathbf{x} \in \mathbb{R}^n$ chosen from set of actions $\mathcal{X}$ that satisfy region sensing in Section 2.2. Consider two single agent TS algorithms where one assumes a 1-sparse prior and another uses a 1-block sparse prior with varying block length as defined in Algorithm 2. Then, the expected regret $\mathbb{E}[Reg(T)] = \mathbb{E}\left[\sum_{t=1}^T \mathcal{R}(\mathbf{x}^\star, \boldsymbol{\beta}) - \mathcal{R}(\mathbf{x}_t, \boldsymbol{\beta})\right]$ for each algorithm is upper-bounded by:*

$$\textit{1-sparse prior}: \mathbb{E}[Reg(T)] \leq \left(\log(|\mathcal{X}|) \sum_{t=1}^{\min\{T, n-1\}} \frac{\left(1 - \frac{t}{n}\right)\left(1 - \frac{1}{n-t+1}\right)}{\left(\frac{n-t-1}{n-t} \log\left(\frac{n-t}{n-t-1}\right) + \frac{1}{n-t} \log(n-t)\right)} \right)^{1/2} \tag{2.9}$$

$$\textit{1-block sparse prior}: \mathbb{E}[Reg(T)] \leq \left(\log(|\mathcal{X}|) \sum_{t=1}^{\min\{T, \log_2(n)\}} \left(1 - \frac{1}{n - \left(\sum_{t'=1}^{t-1} \frac{n}{2^{t'}}\right)}\right)^2 / \log(2) \right)^{1/2} \tag{2.10}$$

A simple comparison of (2.9) and (2.10) in Theorem 1 shows that using TS with a block sparse prior and varying block length significantly reduces the regret bounds comparing to TS that is using the true 1-sparse prior. Next, we will compute and compare an upper bound on the expected regret of a single-agent and an asynchronous multi-agent TS algorithm. To the best of our knowledge, only theoretical analysis for asynchronous parallel TS has been provided by Kandasamy et al. (2018) which is limited to Gaussian Processes. In the

following theorem, we provide theoretical guarantees for an asynchronous multi-agent active search problem with a sparse model.

Theorem 2. *Consider the active search problem in Theorem 1. Let us propose two TS algorithms with a 1-sparse prior where one is single agent and another uses g agents in an asynchronous parallel setting. Then, the expected regret as defined in Theorem 1 for each algorithms is:*

$$\textit{single agent:} \quad \mathbb{E}[Reg(T)] = T_n - \frac{T_n(T_n+1)}{2n}, \quad T_n = \min\{T, n\} \tag{2.11}$$

$$\textit{multi agent:} \quad \mathbb{E}[Reg(T)] \leq T_n - \frac{T_n(T_n+1)}{2n} + \frac{T_n(2g-1)}{n}, \quad T_n = \min\{T, n+g\} \tag{2.12}$$

A simple analysis of (2.12) shows that for $g \ll n$ and $g \ll T$ (which is a reasonable assumption), the third term in the bound will be upper bounded by $2g+1$. As a result, the difference in expected regret between single agent and asynchronous multi agent is negligible in terms of number of measurements T. Hence, we can conclude that by dividing the same number of measurements T between g agents, multi agent algorithm achieves same regret g times faster than single agent setting.

Remark 1. *Theorem 2 shows that our asynchronous multi agent algorithm performs on par with an optimal multi agent system with a central planner. This result is a consequence of the central planner's regret being bounded by the single agent in terms of number of measurements T (Kandasamy et al., 2017).*

2.6 Experiments

Next, we compare the performance of SPATS against the following competitive active search baselines.

2.6.1 Baselines

Region Sensing Index (RSI). RSI is a single agent active search algorithm designed to locate sparse signals by actively making data-collection decisions. Similar to our problem formulation in Section 2.2, RSI makes a practical assumption that at each time step the agent senses a contiguous region of the space. To decide on the next action, RSI at each time step chooses the sensing action $\mathbf{x}_t$ that maximizes the mutual information between the next observation ${}_yt$ and the signal of interest $\boldsymbol{\beta}$, i.e.

$$\mathbf{x}_t = \underset{\mathbf{x}}{\operatorname{argmax}}\, I(\boldsymbol{\beta}; y|\mathbf{x}, \mathbf{D}_t^1)$$

where the mutual information is computed using posterior distribution

$$p(\boldsymbol{\beta}|\mathbf{D}_t^1) = p_0(\boldsymbol{\beta}) \prod_{\tau=1}^{t-1} p(y_\tau|\mathbf{x}, \boldsymbol{\beta})$$

with a k-sparse uniform prior $p_0(\boldsymbol{\beta})$ and same likelihood distribution as in Section 2.5.1.

Unfortunately, computing the mutual information $I(\boldsymbol{\beta}; y|\mathbf{x}, \mathbf{D}_t^1)$ has high complexity for sparsity rates of $k > 1$. In order to reduce this complexity for $k > 1$ RSI recovers the support of $\boldsymbol{\beta}$ by repeatedly applying RSI assuming $k = 1$. Note that this reward function is

deterministic, therefore in a multi-agent setting, all agents will solve for the same sequence of sensing actions without any randomness in their decision making objective.

Laplace TS with Information gain (LATSI). LATSI, proposed in Ghods et al. (2021b), is an asynchronous parallel Thompson Sampling algorithm using a sparse Laplace prior (LaplaceTS), without the assumption of block-sparsity. The decision making objective combines λ^+ from Eq. (2.3) and Eq. (2.4) with the mutual information computed by RSI to mitigate the challenges of TS with sparsity as discussed in Section 2.4.2.

Point Sensing (PS). Sequential PS is a deterministic search baseline where an agent exhaustively searches the environment one location at a time following a predetermined route (sometimes also referred to as lawnmower pattern in the path-planning literature).

2.6.2 2D search space discretized into 8×16 grid cells

In this section, we focus on 2-dimensional search spaces where we estimate a k-sparse signal $\boldsymbol{\beta}$ with length $n = 8 \times 16$ and two sparsity rates of $k = 1, 5$. Here, $\boldsymbol{\beta}$ is generated with a randomly uniform sparse vector. We set the noise variance to $\sigma^2 = 1$. Note that in all of the algorithms considered, agents are not aware of the true uniform sparse prior or sparsity rate k. We then vary the number of measurements T and plot the mean and standard error of the full recovery rate over 50 random trials. Recall that the full recovery rate is defined as the rate at which an algorithm correctly recovers the entire vector $\boldsymbol{\beta}$ over random trials.

Single Agent

In a single agent setting, Figure 2.2 shows that for $k = 1$, RSI and LATSI outperform SPATS. The reason is that RSI has a very accurate approximation of mutual information for $k = 1$ and consequently it is difficult for SPATS to win over the information-optimal algorithms of RSI and LATSI. All algorithms significantly outperform exhaustive search (PS). On the other hand, for higher sparsity rate of $k = 5$, SPATS outperforms RSI and LATSI. This is a result of poor approximation of mutual information for $k > 1$ by RSI. Specifically, for $k > 1$ RSI recovers the support of $\boldsymbol{\beta}$ by repeatedly applying RSI assuming $k = 1$. The authors use this strategy to avoid the large cost of computing mutual information for $k > 1$. This strategy even allows PS to catch up and outperform RSI. Finally, since LATSI is a combination of RSI and LaplaceTS, its performance is tied to that of both RSI and SPATS.

Multi-Agent

Figure 2.4, 2.5 and 2.6 show the performance of SPATS, RSI and LATSI respectively in a multi-agent setting. Each figure consists of 4 sub-figures where the top row illustrates the full recovery rate for $k = 1$ and $k = 5$ as a function of number of measurements (T) taken by all the agents. To better demonstrate the multi-agent performance, in the bottom row we plot full recovery rate as a function of time which is computed by dividing the number of measurements T by the number of agents J. In each sub-figure we vary the number of available agents between $1, 2, 4$ and 8. Note that in all subsequent plots, LATSI-J, RSI-J or SPATS-J indicate the corresponding algorithm with J agents.

SPATS: As evident in the two sub-figures in the bottom row of Figure 2.4, SPATS becomes J times faster by using J number of agents compared to the single-agent setting. From the two sub-figures in the top row, we can draw a similar conclusion. That is,

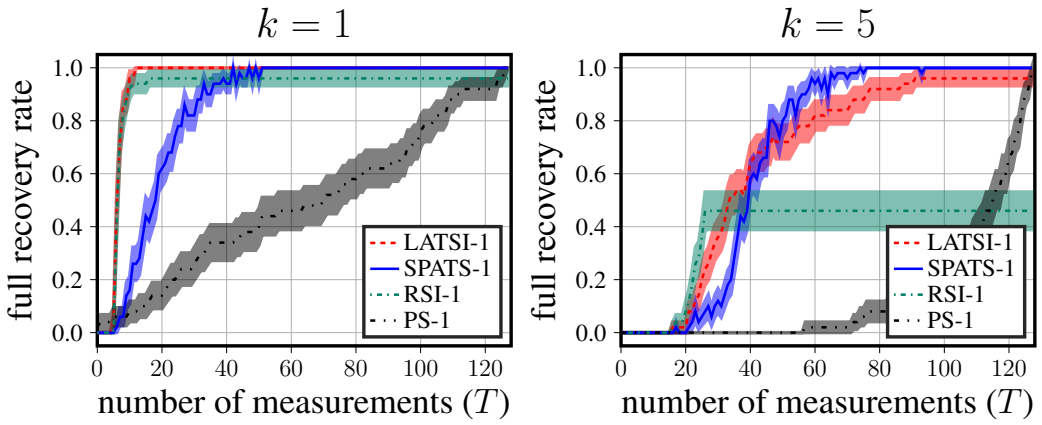


Figure 2.2: Full recovery rate of SPATS, LATSI, RSI and PS (exhaustive) for a single agent for sparsity $k = 1, 5$.

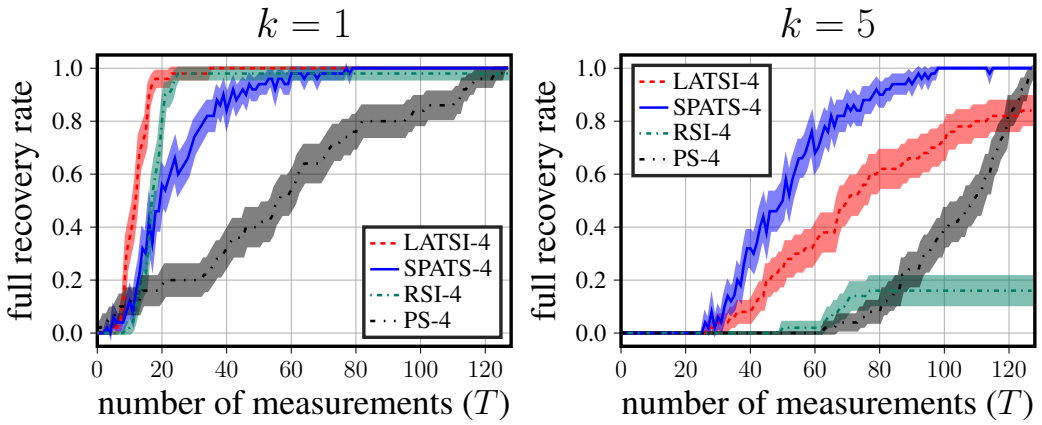


Figure 2.3: Full recovery rate of SPATS, LATSI, RSI and PS (exhaustive) for 4 agents for sparsity $k = 1, 5$.

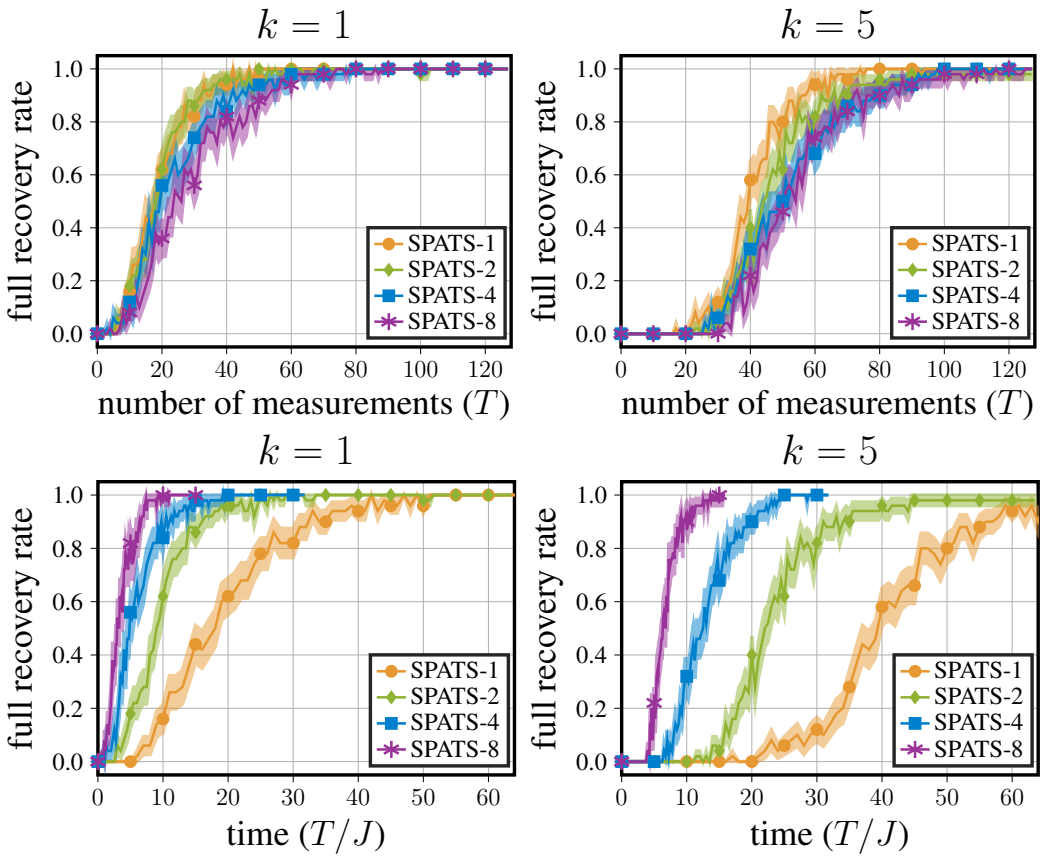


Figure 2.4: Full recovery rate of SPATS with 1, 2, 4 and 8 agents for sparsity rates $k = 1, 5$.

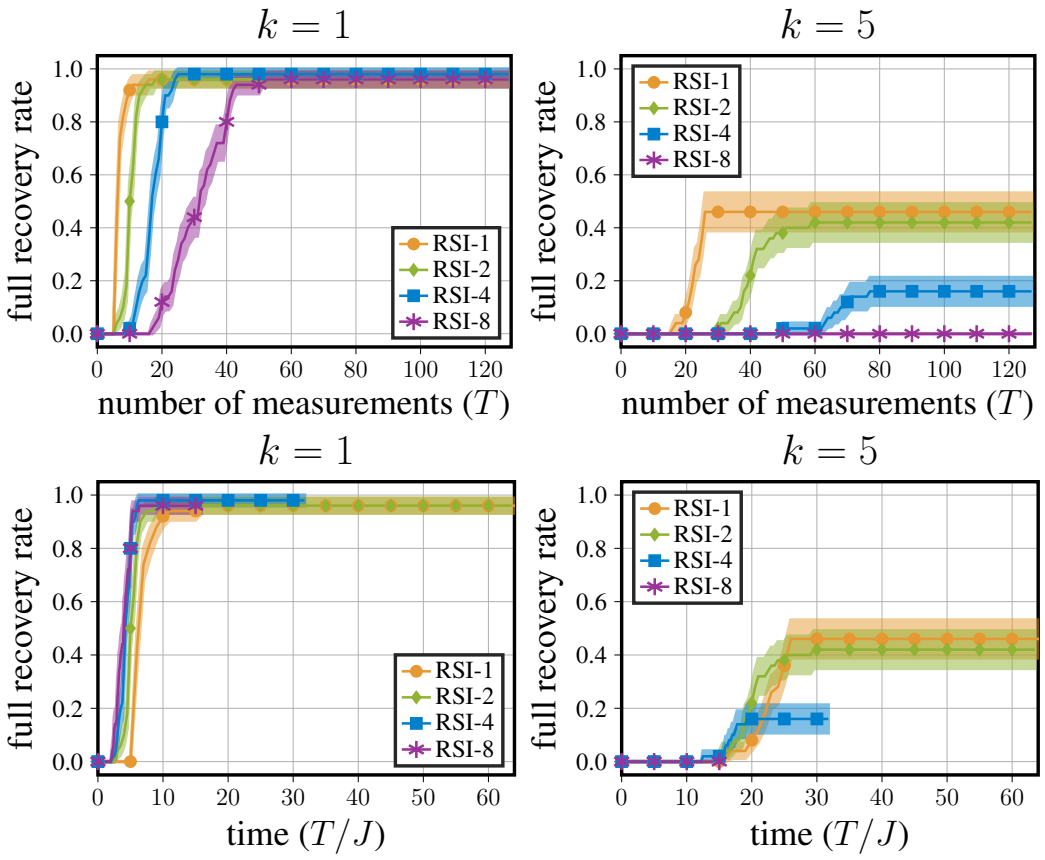


Figure 2.5: Full recovery rate of RSI with 1, 2, 4 and 8 agents for sparsity rates $k = 1, 5$.

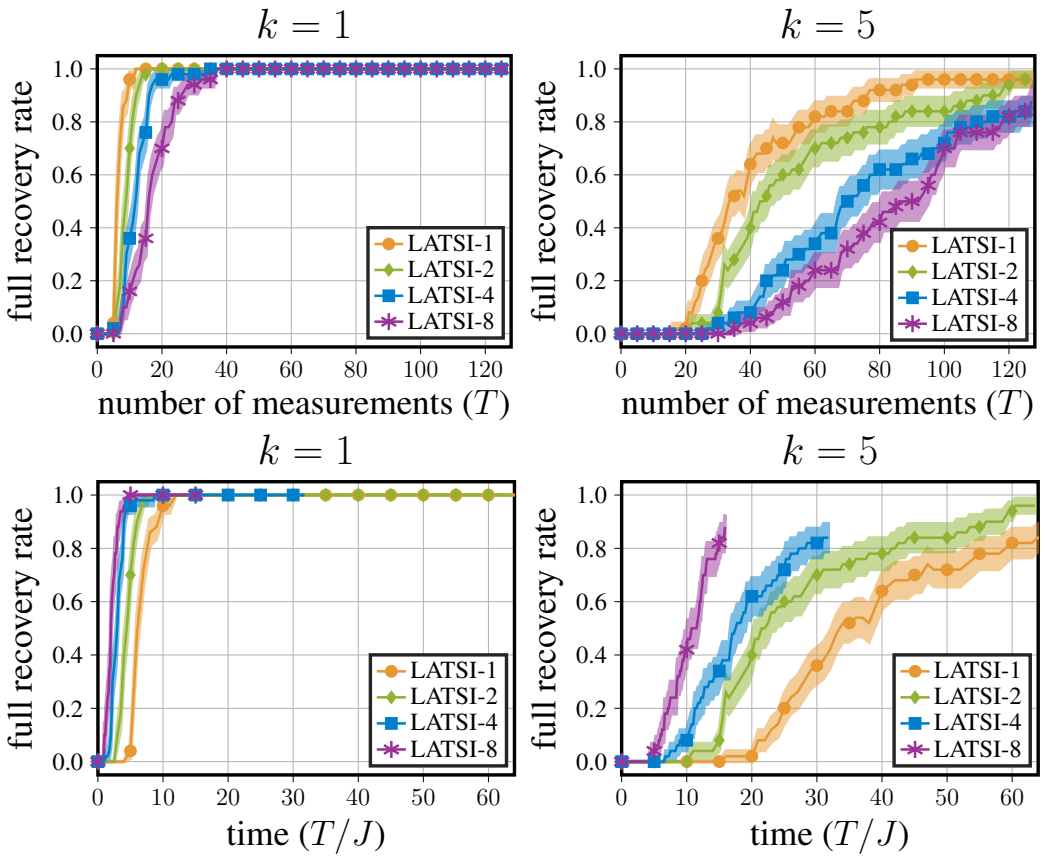


Figure 2.6: Full recovery rate of LATSI with 1, 2, 4 and 8 agents for sparsity rates $k = 1, 5$.

increasing the number of agents from 1 to 2 to 4 and 8 hardly changes the total number of measurements required for a given recovery rate, i.e. the average number of sensing actions per agent is improved about J times. This result demonstrates that SPATS can efficiently perform active search in an asynchronous decentralized fashion.

RSI: We extend the RSI algorithm of Ma et al. (2017) to multi-agent setting by allowing each agent to independently choose its sensing action given RSI's acquisition function and utilizing the available measurements from other agents. Looking at Figure 2.5 for both $k = 1$ and $k = 5$, we see a significant deterioration in full recovery rate as a function of T as the number of agents increases. The reason is that without randomness in RSI's reward function, agents that are working at the same time are repeating the same sensing actions. For $k = 5$, this performance reduction is also obvious as a function of time. However, for $k = 1$ RSI performs slightly better in time by increasing agents. The reason for this contradicting behavior is that RSI's performance for $k = 1$ is so close to optimal (binary search) that it reaches recovery rate of 1 before the multi-agent system can negatively affect it.

LATSI: Looking at Figure 2.6, we see that similar to SPATS, LATSI's multi-agent performance improves in time by increasing the number of agents.

SPATS vs. LATSI vs. RSI: In Figure 2.3, we plot all four algorithms against each other for 4 agents. Here, for $k = 1$, RSI and LATSI outperform SPATS due to their information-theoretic approach in computing the reward function. For $k = 5$, SPATS outperforms both RSI and LATSI. This is because SPATS is carefully designed to use randomness from TS in its reward function such that multiplying the number of agents would multiply its recovery rate. Furthermore, LATSI performs significantly better than RSI due to the probabilistic exploration aspect of TS in its reward function. All algorithms outperform PS except for RSI with $k > 5$ due to its poor information approximation.

2.6.3 Robustness to unreliable inter-agent communication

Next we provide experimental results showing the robustness of SPATS to unreliable communication among agents in asynchronous multi-agent active search. Using the same parameters and setting described above, we focus on a 2-dimensional search space where we estimate a k-sparse signal $\boldsymbol{\beta}$ with length $n = 8 \times 16$ and two sparsity rates of $k = 1, 5$. Assuming J number of agents are actively searching for the targets and asynchronously communicating their measurements to each other, we introduce unreliability in inter-agent communication by randomly dropping, for each agent, up to three of its last received new measurement messages from other agents. We measure the mean and standard error of the full recovery rate as a function of the number of measurements T as well as time over 50 random trials. We vary the number of agents J between 2, 4 and 8. Figure 2.7 shows that the agents are able to fully recover all targets even in this additionally introduced unreliable communication setup. The probabilistic nature of SPATS allows the agents to maintain their performance even when they cannot access all measurements from other agents.

2.7 Discussion

In the remainder of this thesis, we will repeatedly leverage asynchronous parallel Thompson sampling as a decentralized decision making algorithm for multi-agent active search. TS ensures stochasticity in decision making by sampling different plausible realizations of

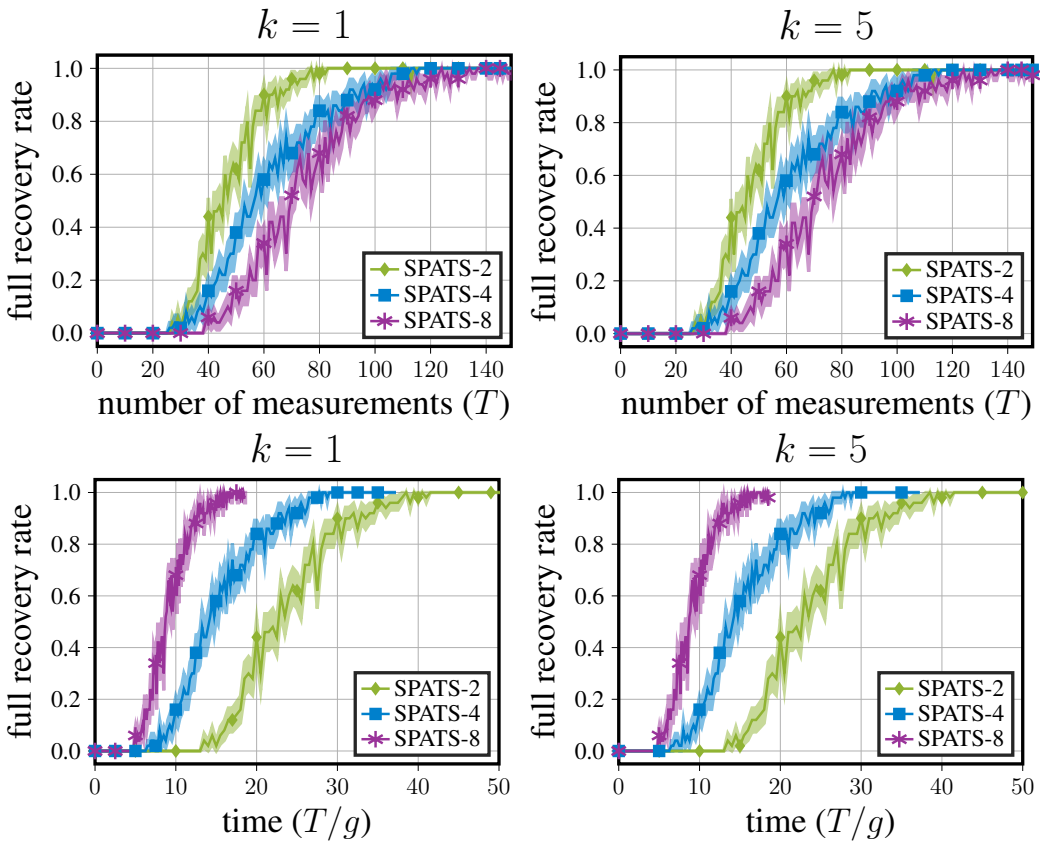


Figure 2.7: Full recovery rate of SPATS in the presence of communication failure with 2, 4 and 8 agents for sparsity rates $k = 1, 5$

the ground truth from the posterior belief and selecting the best action to maximize the desired reward for a particular sample. The uncertainty in the agent's belief over the state space is reflected in the posterior samples, which makes TS suitable for driving exploration and exploration. As we have discussed in Section 2.2, this explore-exploit trade-off is an essential aspect of multi-agent active search and it is one that we will repeatedly encounter in subsequent chapters arising from different practical constraints on the problem setup. In each case, we will observe that a TS-based approach with an appropriate belief model is scalable, robust to unreliable communication and enables decentralized and asynchronous active target recovery with multi-robot teams.

Part I

Non-myopic multi-agent decision making

3 | Multi-Agent Multi-objective optimization with lookahead

3.1 Introduction

Interactive decision making with autonomous agents is typically underlain by the need to balance different, often conflicting, constraints over the task objective. For example, many e-commerce or social media platforms are known to use algorithms that trade-off different factors like user engagement or satisfaction, ads revenue, etc. in recommending users' products or posts. Similarly, in search and rescue missions with multi-robot teams, agents have limited battery capacity which affects their choice of sensing actions to explore the environment in search of survivors. This is because sensing a wider area might have a different energy requirement than point sensing, whereas traveling between consecutive sensing locations can incur additional energy costs. In such scenarios, agents are faced with a multi-objective optimization problem where they must make cost-aware decisions, for example, by trading-off the expected future reward of detecting a survivor with the overall cost of traveling to the appropriate location and executing the sensing action. Unfortunately, Lim et al. (2016) show that even in the single-agent setting, this problem is NP-hard. Moreover there is limited focus in prior work on the additional challenges introduced by multi-objective decision making in multi-agent settings, especially for sequential decision making and under partial observability (Rădulescu et al., 2019). In this chapter, we focus on this problem formulation with the goal of enabling cost-awareness in decentralized and asynchronous multi-agent active search.

3.2 Problem Formulation

Fig. 3.1 depicts the idea behind cost-aware active search. Each agent is actively sensing regions of the search space looking for targets indicated by '×'. To plan its next sensing action, the cost-aware agent has to trade-off the expected future reward of detecting a target with the overall costs it will incur in travelling to the appropriate location and executing the action. Given previous observations, it adaptively makes such data-collection decisions online while minimizing the associated costs as much as possible.

Sensing model. We consider a gridded search environment of size n described by a sparse matrix which is recovered through multi-agent active search. $\boldsymbol{\beta} \in \{0,1\}^{n\times 1}$ denotes the flattened vector representation of the environment having k non-zero entries at the true locations of the k OOIs. $\boldsymbol{\beta}$ is the ground truth search vector. Both the number and location

of targets is unknown to the agents. The sensing model for an agent j at time t is

$$y_t^j = {\mathbf{x}_t^j}^{\mathrm{T}} \boldsymbol{\beta} + \epsilon_t^j, \text{ where } \epsilon_t^j \sim \mathcal{N}(0, \sigma^2). \tag{3.1}$$

$\mathbf{x}_t^j \in \mathbb{R}^n$ is the (flattened) sensing action at time t, with a non-zero support over the sensing region and zeros elsewhere. y_t^j is the agent's observation and ϵ_t^j is a random, i.i.d added noise. The tuple $(\mathbf{x}_t^j, y_t^j)$ is agent j's measurement at time t. We define our action space $\mathcal{A}$ ($\mathbf{x}_t^j \in \mathcal{A}$) to include only hierarchical spatial pyramid sensing actions (Lazebnik et al., 2006). For example in Fig. 3.1, the two agents in the bottom right half of the search space are sensing hierarchical 2×2 and 1×1 regions respectively. Since our search space is discretized into grid cells, the hierarchical actions provide coverage over the search space while reducing the size of the action space (Figs. 3.1b to 3.1d). Considering the search vector $\boldsymbol{\beta} \in \mathbb{R}^n$ of size n, the hierarchical pyramid action space $\mathcal{A}$ is of size $\mathcal{O}(n)$ rather than $\mathcal{O}(n^2)$ had we considered all possible contiguous region sensing actions.

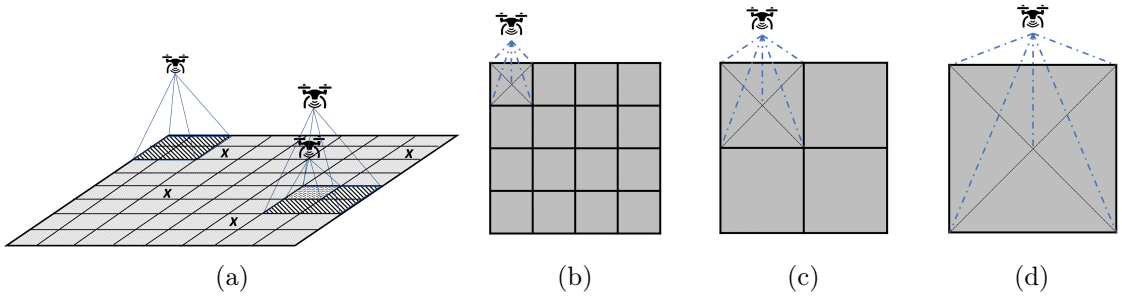
Each sensing action $\mathbf{x}_t \in \mathbb{R}^n$ is a flattened vector whose non-zero support indicates the grid cells being sensed, with $0s$ elsewhere. The support of the vector $\mathbf{x}_t$ is appropriately weighted so that $\|\mathbf{x}_t\|_2 = 1$ to ensure each sensing action has a constant power. This helps us in modeling observation noise as a function of the agent's distance from the region (Ghods et al., 2021b). Since each action has a constant power and every observation y has an i.i.d added noise with a constant variance σ^2, the signal to noise ratio in the unit (1×1) squares comprising the rectangular sensing block reduces as the size of the sensing region is increased with increasing distance between the agent and that region. Therefore agents using this region sensing model must trade-off between sensing a wider area with lower accuracy versus a highly accurate sensing action over a smaller region.

Cost model. Sensing actions for the agents are typically associated with different sources of incurred costs. We consider the following two cost types: *travel cost* and *sensing cost*. First, we consider that the agent travelling from location a to location b incurs a travel time cost $c_d(a, b)$. For this, we assume a constant travelling speed and compute the time taken to traverse the Euclidean distance between locations a and b. Second, we assume that executing each sensing action at location b incurs a time cost $c_s(b)$. In our implementation, we consider a constant value of c_s. Therefore, T steps after starting from location x_0 (corresponding to sensing action $\boldsymbol{x}_0$), an agent j executes actions $\left\{\boldsymbol{x}_t^j\right\}_{t=1}^{T}$ and incurs a total cost defined by

$$C^j(T) = \sum_{t=1}^{T} \Big(c_d\Big(x_{t-1}^j, x_t^j\Big) + c_s \Big).$$

Note that the cost-aware active search strategy may differ based on the relative per unit costs for traveling or sensing: for example, if travelling per unit distance is much more expensive than sensing, the agent might prefer sensing successive locations that are not too far apart depending on the cumulative reward vs. cost trade-off.

In the multi-agent setting, we follow the same communication model outlined in Section 1.1.2 in order to enable decentralized decision making with asynchronous inter-agent communication.



(a) (b) (c) (d)

Figure 3.1: **Sensing model.** (a) Active search with UAVs. '×' indicate true targets. Pyramids with shaded base indicate field of view (FOV) for each agent. (b), (c), (d) Hierarchical pyramid region sensing actions of size 1×1, 2×2 and 4×4 respectively in a 4×4 search space.

3.3 Related Work

Previous studies have used various constraints and reductions to achieve resource efficient adaptive search. Adaptive sensing applications in robotics typically reduce it to a planning problem assuming full observability of the environment (Pěnička et al., 2019; Kent and Chernova, 2020). Imposing a cost budget in such applications is modeled as constrained path planning between the known start and goal locations. Unfortunately, this is in contrast with the real world where the agent's environment, the number of targets and their locations may be unknown and the agent may have access only to noisy observations from sensing actions. All these factors increase the difficulty of cost-aware active search.

Our active search formulation has close similarities with planning under uncertainty using a Partially Observable Markov Decision Process (POMDP) (Kaelbling et al., 1998). While computing exact optimal policies for POMDPs is generally intractable, they can often be approximated using algorithms like Point-Based Value Iteration (PBVI) (Pineau et al., 2003) or Monte Carlo methods. Monte Carlo Tree Search (MCTS) (Kocsis and Szepesvári, 2006; Browne et al., 2012) is an online algorithm that combines tree search with random sampling in a domain independent manner and has found success as a generic online planning algorithm in large POMDPs (Silver and Veness, 2010). Prior work (Flaspohler et al., 2019; Fischer and Tas, 2020) has proposed single agent MCTS algorithms for (single objective) adaptive decision making using information theoretic reward in continuous state and observation domain POMDPs.

Decentralized POMDP (Dec-POMDP) (Bernstein et al., 2002; Oliehoek et al., 2016) is another framework for decentralized active information gathering using multiple agents which is typically solved using offline, centralized planning followed by online, decentralized execution (Lauri et al., 2020; Lauri and Oliehoek, 2020). Decentralized MCTS (Dec-MCTS) algorithms have also been proposed for multi-robot active perception under a cost budget (Sukkar et al., 2019; Best et al., 2020) but they typically rely on each agent optimizing for a known global objective while maintaining a joint probability distribution over its own belief as well as those of the other agents, that helps ensure inter-agent coordination.

Prior work in reinforcement learning and planning has typically focused on single objective optimization problems. In the multi-objective optimization (MOO) setting, some algorithms consider a scalarized form of separate reward and cost value functions (Lee

et al., 2018), but scalarization requires that the different objectives be weighted appropriately and the policy has to be updated each time the scalarized objective changes. Several multi-objective reinforcement learning (MORL) and planning algorithms have been proposed that learn a single policy or multiple policies for different objectives, or predict a pareto-front for the MOO problem. For a brief overview, pareto-optimization builds on the idea that some solutions to the MOO problem are categorically worse than others and are dominated by a set of pareto-optimal solution vectors forming a pareto front for the optimization objective. Considering a set of D-dimensional vectors $\mathbf{v} \in \mathcal{V}$, we define the following:

- $\mathbf{v}$ dominates $\mathbf{v}'$ (denoted $\mathbf{v} \succ \mathbf{v}'$) iff (1) $\forall d \in \{1, \ldots, D\}$, $[\mathbf{v}]_d \geq [\mathbf{v}']_d$ (2) $\exists d \in \{1, \ldots, D\}$, $[\mathbf{v}]_d > [\mathbf{v}']_d$
- $\mathbf{v}$ and $\mathbf{v}'$ are incomparable (denoted $\mathbf{v}||\mathbf{v}'$) iff $\exists d_1, d_2 \in \{1, \ldots, D\}$ s.t. $[\mathbf{v}]_{d_1} > [\mathbf{v}']_{d_2}$ *and* $[\mathbf{v}]_{d_2} < [\mathbf{v}']_{d_2}$
- $\mathcal{V}^* \subseteq \mathcal{V}$ is the pareto-front of $\mathcal{V}$ iff (1)$\forall \mathbf{v} \in \mathcal{V}$ and $\forall \mathbf{v}' \in \mathcal{V}^*$, $\mathbf{v} \not\succ \mathbf{v}'$ (2) $\forall \mathbf{v}, \mathbf{v}' \in \mathcal{V}^*$, $\mathbf{v}||\mathbf{v}'$.

Typically, each element of the vector $\mathbf{v}$ indicates an objective in the MOO problem and the pareto-optimal set of vectors $\mathbf{v}^* \in \mathcal{V}^*$ comprises feasible solutions that are pairwise non-dominated or incomparable. Previously, Wang and Sebag (2012) introduced multi-objective MCTS (MO-MCTS) for discovering global pareto-optimal decision sequences in the search tree. Unfortunately, MO-MCTS is computationally expensive and unsuitable for online planning. Further, Chen and Liu (2019) proposed the Pareto MCTS algorithm for multi-objective informative path planning but they ignore uncertainty due to partial observability in the search space. We refer to Hayes et al. (2022) for a comprehensive overview of prior approaches to MORL.

Most of these methods are however limited to the single agent setting with full observability over the state space. In contrast, for multi-objective multi-agent active search, learning to predict pareto-optimal actions would face the additional challenge of reasoning over the agent's belief about the environment under partial observability, as well as adapting to different team sizes and asynchronous inter-agent communication. Therefore computing the entire pareto-optimal solution set may be intractable especially in non-deterministic environments or with Monte Carlo methods that simulate rollouts over a sequence of timesteps. Moreover, the reward from successive region sensing actions in our active search setup follows adaptive submodularity (Golovin and Krause, 2011) and at any timestep, the reward is also invariant to actions that are symmetric in terms of the agent's posterior belief updates. These distinctive characteristics of our problem setting compared to prior work further motivate our study of cost-aware multi-agent active search.

3.3.1 Naïve approach and challenges

Given our previously described approach to multi-agent active search using Thompson sampling and a sparse posterior belief update algorithm SPATS in Ghods et al. (2021b), one might think of an extension to pareto-optimization over reward-cost vectors as an initial approach to enabling cost-awareness in the agents. But this would still be limited to myopic decision making which has been previously demonstrated to lead to worse performance than lookahead based policies (Jiang et al., 2019). Following the MCTS-based algorithms, a different approach could combine search-tree based belief-action space rollouts using the

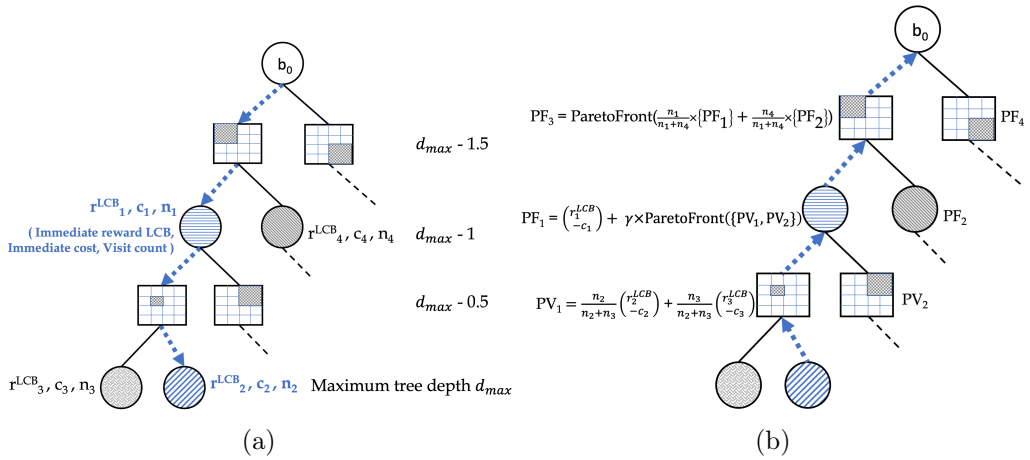


Figure 3.2: Illustration of a search tree $\mathcal{T}_t$ with $d_{\max} = 2$. $\boldsymbol{b}_0$ is the belief at the root node. The action nodes (rectangles) indicate region sensing actions. The belief nodes (circles) are shaded to indicate the evolving posterior belief in the search tree. (a) Top-down traversal for episode m'. r_1^{LCB}, c_1, n_1, r_2^{LCB}, c_2 and n_2 are updated. (b) Bottom-up traversal for episode m'. $PV_{\{1,2\}}$ are vectors, $PF_{\{1,2,3,4\}}$ are the pareto fronts at the respective belief and action nodes. γ is the discount factor. ParetoFront() obtains the pareto-optimal vectors from an input set. During backpropagation, only PV_1, PF_1 and PF_3 are updated corresponding to nodes encountered during top-down traversal.

UCT algorithm over lookahead expected reward-cost vectors for decision making. Note that this method would lead to deterministic action selection in a multi-agent setting and without a central controller, agents might end up choosing the same sensing action thereby leading to redundant observations. Instead, our main insight is to leverage the decentralized decision making ability of Thompson sampling with pareto-optimization over lookahead reward-cost vectors estimated using finite horizon rollouts in MCTS.

3.4 CAST: Cost-Aware Active Search of Sparse Targets

Our proposed algorithm, CAST (Cost-aware Active Search of Sparse Targets) (Algorithm 3) enables cost-aware lookahead multi-agent active search. It combines Thompson sampling with Monte Carlo Tree Search for lookahead planning and multi-agent decision making, along with Lower Confidence Bound (LCB) style pareto optimization to tradeoff expected future reward with the associated costs.

3.4.1 Notation for CAST

Table 3.1 summarizes the various symbols and notations used in describing CAST.

3.4.2 Belief representation in CAST

Since the number of targets k and their locations are unknown, each agent must maintain a belief or probability distribution over the targets in the search space. Assuming that

Algorithm 3 CAST: Cost-Aware Active Search of Sparse Targets

procedure MAIN ▷ Executed on each agent j
  for t in $\{1, 2, \ldots, T\}$ **do**
    $\boldsymbol{x}_t^j$ = SEARCH($\mathbb{D}_t^j, x_{t-1}^j, \boldsymbol{b}_t^j$)
    Execute $\boldsymbol{x}_t^j$. Observe $\boldsymbol{y}_t^j$. $\mathbb{D}_{t+1}^j = \mathbb{D}_t^j \cup \{\boldsymbol{x}_t^j, \boldsymbol{y}_t^j\}$
    Share $\{\boldsymbol{x}_t^j, \boldsymbol{y}_t^j\}$ asynchronously with teammates.
    Update belief $\boldsymbol{b}_{t+1}^j$ and estimate $\hat{\boldsymbol{\beta}}(\mathbb{D}_{t+1}^j)$.
procedure SEARCH($\mathbb{D}, x, \boldsymbol{b}$)
  Search tree $\mathcal{T} = \phi$
  for each episode $m' \in \{1 \ldots m\}$ **do**
    Sample $\tilde{\boldsymbol{\beta}} \sim \boldsymbol{b}$. Discretize $\tilde{\boldsymbol{\beta}}$ to get $\tilde{\boldsymbol{\beta}}_{m'}$.
    SIMULATE($\tilde{\boldsymbol{\beta}}_{m'}, \mathbb{D}, x, 0$)
  $\mathcal{A}^*$ =ParetoOptimalActionSet($\mathcal{T}$)
  $\boldsymbol{x}^* = \text{argmax}_{\boldsymbol{x}}\{\frac{\boldsymbol{x}.r^{LCB}}{\boldsymbol{x}.cost} | \boldsymbol{x} \in \mathcal{A}^*\}$
  return $\boldsymbol{x}^*$
procedure SIMULATE($\tilde{\boldsymbol{\beta}}, \mathbb{D}, x, \text{depth}$)
  $n(h) \leftarrow n(h) + 1$ ▷ Denote root (belief) node as h
  if depth $= d_{\max}$ **then return** $0, 0$ ▷ Reached leaf node
  if $\lfloor n(h)^{\alpha_s} \rfloor > \lfloor (n(h)-1)^{\alpha_s} \rfloor$ **then**
    add new child action node (h, a)
  else select action node (h, a) using (3.9)
  $n(h, a) \leftarrow n(h, a) + 1$ ▷ a is the location corresponding to action $\boldsymbol{a}$
  $\boldsymbol{o} \leftarrow \boldsymbol{a}^{\mathrm{T}}\boldsymbol{\beta}$, $\mathbb{D}' \coloneqq \mathbb{D} \cup \{\boldsymbol{a}, \boldsymbol{o}\}$
  if $\boldsymbol{o}$ was not previously observed at (h, a) **then**
    append new node h' due to $\boldsymbol{o}$ in branch hah'
  $r_{h'} = \lambda^-(\boldsymbol{\beta}, \mathbb{D}'), c_{h'}(x, a) = c_d(x, a) + c_s$
  Update $r_{h'}^{LCB}$ and $\boldsymbol{g}_{h'} = \begin{bmatrix} r_{h'}^{LCB} & -c_{h'}(x, a) \end{bmatrix}^{\mathrm{T}}$
  $r', c' =$ SIMULATE($\boldsymbol{\beta}, \mathbb{D}', a, \text{depth} + 1$)
  $r'' = r_{h'} + \gamma \times r'$, $c'' = c_{h'} + c'$
  $\bar{Q}^{UCT}(h, a) = \frac{\bar{Q}^{UCT}(h,a) \times (n(h,a)-1) + \frac{r''}{c''}}{n(h,a)}$
  LCBParetoFrontUpdate(h')
  LCBParetoFrontUpdate((h, a))
  return r'', c''

Table 3.1: Symbols and notations in Section 3.4

Notation	Definition
$\boldsymbol{b}_t(\boldsymbol{\beta})$	Posterior belief over $\boldsymbol{\beta}$ at time step t
$\mathbb{D}_t^j$	Set of past actions and observations available to the agent j at time t
λ^-	One-step lookahead reward for a CAST agent
$d_{\max}$	Maximum lookahead depth of the search tree. Successive levels differ by a depth of 0.5.
n	Dimension of $\boldsymbol{\beta}$
$n(h)$	Number of times belief node h is visited in search tree $\mathcal{T}_t$
$n(h,a)$	Number of times action node (h,a) is visited in search tree $\mathcal{T}_t$
x	Position of an agent in the search space after executing sensing action $\boldsymbol{x}$
a	Position of an agent in the search space after executing sensing action $\boldsymbol{a}$
$c_d(x,a)$	Time cost of agent travelling from position x to position a
c_s	Time cost of agent executing a sensing action $\mathbf{x}$
γ	Discount factor for computing multi-step lookahead reward over a finite horizon
α_s	Progressive widening parameter
m	Total number episodes of tree building in every decision making time step
J	Total number of agents performing active search

the targets are sparsely distributed in the environment, an agent's belief over search vector $\boldsymbol{\beta} \in \mathbb{R}^n$ is modeled by a sparse prior $\boldsymbol{b}_0$:

$$\boldsymbol{b}_0 = p(\boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0), \tag{3.2}$$

where $\boldsymbol{\mu}_0 = \frac{1}{n} * \mathbf{1}_{n\times 1}$ and $\boldsymbol{\Sigma}_0 = \text{diag}(\boldsymbol{\tau})$, with hyperparameter $\boldsymbol{\tau} \in \mathbb{R}^n$. For any agent j, given the measurement set $\mathbb{D}_t^j = \{\boldsymbol{x}_i^j, y_i^j\}_{i=1}^{t-1}$ we define $\boldsymbol{X}_t^j$ as the matrix $[\boldsymbol{x}_1^{j\,\mathrm{T}} \ ... \ \boldsymbol{x}_{t-1}^{j\,\mathrm{T}}]^{\mathrm{T}}$ and $\boldsymbol{y}_t^j$ as the column vector $[y_1^j \ ... \ y_{t-1}^j]^{\mathrm{T}}$. From Eq. (3.1), the likelihood function is $p(\boldsymbol{y}_t^j | \boldsymbol{X}_t^j, \boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{X}_t^j \boldsymbol{\beta}, \sigma^2 * \mathbf{I}_{t-1})$. Therefore, the posterior belief over $\boldsymbol{\beta}$ is

$$p(\boldsymbol{\beta} | \mathbb{D}_t^j, \boldsymbol{\tau}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\beta}}^j(\boldsymbol{\tau}), \boldsymbol{\Sigma}_{\boldsymbol{\beta}}^j(\boldsymbol{\tau})) \tag{3.3}$$

where

$$\boldsymbol{\Sigma}_{\boldsymbol{\beta}}^j(\boldsymbol{\tau}) = ((\boldsymbol{\Sigma}_0)^{-1} + \frac{1}{\sigma^2} \boldsymbol{X}_t^{j\,\mathrm{T}} \boldsymbol{X}_t^j)^{-1} \tag{3.4}$$

$$\boldsymbol{\mu}_{\boldsymbol{\beta}}^j(\boldsymbol{\tau}) = \boldsymbol{\mu}_0 + \frac{1}{\sigma^2} \boldsymbol{\Sigma}_{\boldsymbol{\beta}}^j(\boldsymbol{\tau}) \boldsymbol{X}_t^{j\,\mathrm{T}} (\boldsymbol{y}_t^j - \boldsymbol{X}_t^j \boldsymbol{\mu}_0). \tag{3.5}$$

The hyperparameter $\boldsymbol{\tau}$ can be estimated using Expectation Maximization following Zhang and Rao (2011) over $p = 1, 2, \ldots, n_{EM}$ iterations as follows.

$$\textbf{E-step: } \boldsymbol{\mu}_{\boldsymbol{\beta}}^{(p)} = \boldsymbol{\mu}_{\boldsymbol{\beta}}(\boldsymbol{\tau}^{(p-1)}) \ \ \boldsymbol{\Sigma}_{\boldsymbol{\beta}}^{(p)} = \boldsymbol{\Sigma}_{\boldsymbol{\beta}}(\boldsymbol{\tau}^{(p-1)}) \tag{3.6}$$

$$\textbf{M-step: } \forall i = 1, \ldots, n, \ \ [\boldsymbol{\tau}^{(p)}]_i = [\boldsymbol{\Sigma}_{\boldsymbol{\beta}}^{(p)}]_{ii} + ([\boldsymbol{\mu}_{\boldsymbol{\beta}}^{(p)}]_i - \frac{1}{n})^2 \tag{3.7}$$

At time step t, after the agent executes action $\boldsymbol{x}_t^j$ and receives an observation y_t^j, we update its estimate $\hat{\boldsymbol{\beta}}(\mathbb{D}_t^j \cup \{\boldsymbol{x}_t^j, y_t^j\})$ using the MAP estimator given by:

$$\hat{\boldsymbol{\beta}}(\mathbb{D}_t^j \cup \{\boldsymbol{x}_t^j, y_t^j\}) = \big(\sigma^2 * {\boldsymbol{\Sigma}_0}^{-1} + \begin{bmatrix} {\mathbb{D}_t^j}^{\mathrm{T}} & \boldsymbol{x}_t^j \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_t^j \\ {\boldsymbol{x}_t^j}^{\mathrm{T}} \end{bmatrix} \big)^{-1} \begin{bmatrix} {\boldsymbol{X}_t^j}^{\mathrm{T}} & \boldsymbol{x}_t^j \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_t^j \\ y_t^j \end{bmatrix} \tag{3.8}$$

where $\boldsymbol{\Sigma}_0 = \mathrm{diag}(\boldsymbol{\tau}^{(n_{EM})})$.

3.4.3 Overview of CAST

We now describe our proposed algorithm CAST. At each time step t, on the basis of its history $\mathbb{D}_t^j$ of past measurements, an agent j decides its next region sensing action $\boldsymbol{x}_t^j$ using the SEARCH procedure of Algorithm 3. It starts with an empty tree $\mathcal{T}_t^j$ having just a root node and gradually builds it up over m episodes. We assume a maximum tree depth $d_{\max}$.

The search tree has two types of nodes - *belief nodes* and *action nodes*. A belief node h is identified by the history of actions and observations accumulated in reaching that node. An action node (h, a) is identified by the action $\boldsymbol{a}$ taken at the immediately preceding belief node h in the search tree. The root as well as the leaves are belief nodes.

Each episode $m' \in \{1, \ldots, m\}$ comprises the following:

1. *Sampling*: First, a posterior sample is drawn at the root node from the belief $\boldsymbol{b}_t^j = p\Big(\boldsymbol{\beta} | \mathbb{D}_t^j\Big)$ and discretized into a binary vector $\boldsymbol{\beta}_{m',t}^j \in \{0,1\}^n$ (Line 10).
2. *Selection and Expansion*: Starting at the root node, a child action node selection policy (tree policy, described later) is applied at every belief node h in a top-down depth-first traversal till a leaf node is reached. In order to prevent tree width explosion with increasing size of the action space, the progressive widening parameter α_s (Line 18) (Coulom, 2007) determines when a new action node is added to the tree. Arriving at any action node (h, a), the corresponding maximum likelihood observation $o = \boldsymbol{a}^{\mathrm{T}} \boldsymbol{\beta}_{m',t}^j$ is computed (Line 22) which helps transition to its child belief node h'. The one-step reward $\lambda_{m'}^-$ for $\boldsymbol{\beta} = \boldsymbol{\beta}_{m',t}^j$ and associated execution cost is computed at each belief node visited in m' (Line 25). Every belief and action node in m' also updates the number of times it has been visited so far (Lines 16 and 21).
3. *Backpropagation*: Once the maximum depth $d_{\max}$ is reached, the lookahead rewards and associated costs are backpropagated up from the leaf to each belief and action node visited in m' (Line 17). Each action node stores the discounted cumulative reward per unit cost averaged over $n(h, a)$ simulations in the subtree rooted at that node (Line 29). Further, each belief and action node builds a reward-cost pareto front (Lines 30 and 31) using the backed up values from their respective subtrees which is utilized in deciding $\boldsymbol{x}_t$ after m episodes (Line 12).

Reward computation in CAST.

Our proposed formulation of active search for recovering the true search vector $\boldsymbol{\beta}$ is closely related to the parameter estimation problem in active learning. Kandasamy et al. (2019) proposed the Myopic Posterior Sampling (MPS) objective which selects actions that maximally reduce mean squared error between a posterior sample and the one-step lookahead

(myopic) estimate. In other words, MPS chooses $\boldsymbol{x}_t^j$ that maximizes

$$\mathbb{E}_{\boldsymbol{y}_t^j|\boldsymbol{x}_t^j,\tilde{\boldsymbol{\beta}}_t^j}\left[\lambda\Big(\tilde{\boldsymbol{\beta}}_t^j, \mathbb{D}_t^j \cup \Big\{\boldsymbol{x}_t^j, \boldsymbol{y}_t^j\Big\}\Big)\right] = \mathbb{E}_{\boldsymbol{y}_t^j|\boldsymbol{x}_t^j,\tilde{\boldsymbol{\beta}}_t^j}\left[-\left\|\tilde{\boldsymbol{\beta}}_t^j - \hat{\boldsymbol{\beta}}_{t+1}^j\right\|_2^2\right]$$

where $\tilde{\boldsymbol{\beta}}_t^j \sim \boldsymbol{b}_t^j$ and $\hat{\boldsymbol{\beta}}_{t+1}^j$ is the MAP estimate

$$\hat{\boldsymbol{\beta}}_{t+1}^j = \left(\sigma^2 * {\boldsymbol{\Sigma}_0}^{-1} + \begin{bmatrix}{\boldsymbol{X}_t^j}^{\mathrm{T}} & \boldsymbol{x}_t^j\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_t^j \\ {\boldsymbol{x}_t^j}^{\mathrm{T}}\end{bmatrix}\right)^{-1}\begin{bmatrix}{\boldsymbol{X}_t^j}^{\mathrm{T}} & \boldsymbol{x}_t^j\end{bmatrix}\begin{bmatrix}\boldsymbol{Y}_t^j \\ \boldsymbol{y}_t^j\end{bmatrix}$$

where $\Big\{\boldsymbol{X}_t^j, \boldsymbol{Y}_t^j\Big\}$ constitute the measurements in $\mathbb{D}_t^j$. Essentially, $\lambda\Big(\tilde{\boldsymbol{\beta}}_t^j, \mathbb{D}_t^j \cup \Big\{\boldsymbol{x}_t^j, \boldsymbol{y}_t^j\Big\}\Big)$ is designed so that the agent will choose $\boldsymbol{x}_t^j$ to keep exploring the search space as long as there is uncertainty in the posterior samples $\tilde{\boldsymbol{\beta}}_t^j$. Simultaneously, the posterior belief distribution will contain uncertainty as long as there are unexplored or less explored locations in the search space.

In this setting, we note that $\lambda\Big(\tilde{\boldsymbol{\beta}}_t^j, \mathbb{D}_t^j \cup \Big\{\boldsymbol{x}_t^j, \boldsymbol{y}_t^j\Big\}\Big) \leq 0$, therefore if we simply extend the MPS reward over multiple lookahead steps and try to maximize the value of cumulative discounted reward divided by total incurred cost, it would erroneously favor costlier actions for the same reward. Instead, we propose using

$$\lambda^-\Big(\tilde{\boldsymbol{\beta}}_t^j, \mathbb{D}_t^j \cup \Big\{\boldsymbol{x}_t^j, \boldsymbol{y}_t^j\Big\}\Big) = \max\left\{0, \left\|\tilde{\boldsymbol{\beta}}_t^j - \hat{\boldsymbol{\beta}}_t^j\right\|_2^2 - \left\|\tilde{\boldsymbol{\beta}}_t^j - \hat{\boldsymbol{\beta}}_{t+1}^j\right\|_2^2\right\}$$

as the one-step lookahead reward (*Line* 25). We design λ^- to encourage information gathering by favoring actions $\boldsymbol{x}_t$ that reduce the uncertainty in the posterior sample $\tilde{\boldsymbol{\beta}}_t^j$ over consecutive time steps. Additionally, $\lambda^- \geq 0$. Therefore, we can compute the u-step lookahead reward $\mathcal{R}^u\Big(\boldsymbol{x}_t, \tilde{\boldsymbol{\beta}}_t^j\Big)$ over the action sequence $\boldsymbol{x}_{t:t+u}$ as the γ-discounted expected sum of λ^- over u steps.

$$\mathcal{R}^u\Big(\boldsymbol{x}_t, \tilde{\boldsymbol{\beta}}_t^j\Big) = \mathbb{E}_{\boldsymbol{y}_{t:t+u}}\left[\sum_{\delta t=1}^{u}\gamma^{\delta t-1}\lambda^-\Big(\tilde{\boldsymbol{\beta}}_t^j, \mathbb{D}_{t+\delta t}^j\Big)\right]$$

We observe that $\mathcal{R}^u\Big(\cdot, \tilde{\boldsymbol{\beta}}_t^j\Big)$ is dependent on the posterior sample $\tilde{\boldsymbol{\beta}}_t^j$. Particularly, $\lambda^-\Big(\tilde{\boldsymbol{\beta}}_t^j, \mathbb{D}_{t+1}^j\Big)$ is higher for sensing actions $\boldsymbol{x}_t$ that identify the non-zero support elements of the vector $\tilde{\boldsymbol{\beta}}_t^j$. Moreover, maximizing $\mathcal{R}^u\Big(\boldsymbol{x}_t, \tilde{\boldsymbol{\beta}}_t^j\Big)$ over all sequences $\boldsymbol{x}_{t:t+u}$ for a sampled $\tilde{\boldsymbol{\beta}}_t^j$ would exacerbate this problem by choosing a series of point-sensing actions that identify the non-zero support of the particular sample. Section 8.2 of Russo et al. (2017) also highlights this drawback of employing TS based exploration in active learning problems that require a careful assessment of the information gained from actions. In order to overcome these challenges, we propose generalizing the posterior sampling step to a sample size greater than one and combine the information from these samples using confidence bounds over λ^- to evaluate the corresponding sensing actions (described later).

Tree policy.

UCT (Upper Confidence Bound applied to trees) (Kocsis and Szepesvári, 2006) is the tree policy used in most MCTS implementations to balance exploration-exploitation in building the search tree. UCT exploits action nodes based on their lookahead reward estimates averaged over past episodes but does not account for the inter-episode variance in such rewards. Particularly in our setting, the lookahead reward at any action node in an episode m' depends on the posterior sample $\boldsymbol{\beta}^j_{m'}$ drawn at the root node and this stochasticity leads to sample variance especially when the particular action node has been visited in only a few episodes. We can account for this variance using the UCB-tuned policy (Audibert et al., 2006) to guide action node selection. Separately, Shah et al. (2020) formalize a correction to the UCT formula in an MDP framework replacing its logarithmic exploration term with an appropriate polynomial. We extend it to our tree policy in CAST, called CAST-UCT (3.9), by combining it with UCB-tuned to balance exploration with exploitation while building the search tree in our partially observable state space. Specifically, CAST-UCT chooses

$$\boldsymbol{a}^* = \underset{\boldsymbol{a}}{argmax}\, Q(h,a) + \sqrt{\frac{2\sigma_{h,a}\sqrt{n(h)}}{n(h,a)} + \frac{16\sqrt{n(h)}}{3n(h,a)}}. \tag{3.9}$$

$\sigma^2_{h,a}$ is the variance of the terms averaged in $Q(h,a)$ (Line 29).

Pareto front construction with confidence bounds.

During the selection and expansion phase in any episode m', the one-step lookahead reward $\lambda^-_{m'}$ is computed at each visited belief node h (Line 25). We note that $\lambda^-_{m'}$ depends on the posterior sample $\boldsymbol{\beta}_{m'}$ drawn for that episode. Assuming that a belief node h is visited in $n(h)$ episodes so far while building the search tree $\mathcal{T}_t$, we account for the stochasticity in the computed λ^- by maintaining the Lower Confidence Bound (LCB) of these rewards (denoted r^{LCB}_h) using the Student's t-distribution to estimate a 95% confidence interval (Line 26). Denoting the cost of executing the action that transitions into the belief node h as c_h (Line 25), we define a LCB based immediate (one-step lookahead) reward-cost vector at h, $\boldsymbol{g}_h = \begin{bmatrix} r^{LCB}_h & -c_h \end{bmatrix}^{\mathrm{T}}$ which is essential to our multi-objective decision making as described next. Fig. 3.2a highlights, in blue, these variables updated during the selection and expansion phase in one episode.

Next, we compute the pareto front over the multi-step lookahead reward-cost vectors at tree nodes visited during the backpropagation phase in episode m'. Fig. 3.2b illustrates this process. Note that the search tree depth at the leaf nodes is $d_{\max}$ and consecutive action and belief nodes differ in depth by 0.5. The lookahead reward-cost vector at the action node at depth $d_{\max} - 0.5$ is the weighted average of the reward-cost vectors $\boldsymbol{g}_{h_\ell}$ of all its leaves h_ℓ, weights being in proportion of their visits. Next, the belief node at depth $d_{\max} - 1$ builds a pareto front from the lookahead reward-cost vectors of all its children action nodes. It then takes a discounted sum of its immediate reward-cost vector with this pareto front to build its lookahead reward-cost vector *set* since the pareto front may comprise multiple non-dominated pareto-optimal vectors.

Repeating these steps in episode m' all the way up to the root node, we alternate between the following: 1) every action node builds its lookahead reward-cost vector set as the pareto front computed from the weighted average of the lookahead vectors of its children

belief nodes 2) every belief node builds its lookahead reward-cost vector set by taking the discounted sum of its immediate reward-cost vector with the pareto front obtained from its children action nodes. Note that all reward-cost vectors use the LCB of the rewards. Therefore, at the end of m episodes, each child action node of the root has an LCB based pareto front of lookahead reward-cost vectors. $\mathcal{A}_t^*$ (Line 12) is the pareto front at the root node comprising the non-dominated lookahead reward-cost vectors among its children action nodes. Finally, the agent selects the action node at the root having the maximum value of reward per unit cost among its vectors in $\mathcal{A}_t^*$ (Line 13). This completes the agent's decision making step at time t.

Remark 1. *In summary, posterior sampling based lookahead planning enables decentralized and asynchronous multi-agent decision making in CAST so that each agent can select and execute region sensing actions using its current posterior belief, which is updated with its own previous measurements and those received from other agents. Additionally, LCB based pareto front construction helps select actions taking into account the sample variability in multi-step reward-cost trade-off computation.*

3.5 Experiments

We now evaluate CAST by comparing in simulation the total cost incurred during multi-agent active search using cost-aware agents against the cost agnostic active search algorithms SPATS (Ghods et al., 2021b) and RSI (Ma et al., 2017). SPATS is a TS based algorithm for asynchronous multi-agent active search, whereas RSI chooses sensing actions that maximize its information gain. We also consider sequential point sensing (PS) as a baseline for exhaustive coverage. Next we briefly review their key features.

3.5.1 Baselines

Region Sensing Index (RSI) (Ma et al., 2017) is a single agent myopic active search algorithm wherein at any time step t, the agent selects the region sensing action $\boldsymbol{x}_t$ which would maximize the mutual information between the resulting observation $\boldsymbol{y}_t$ and the search vector $\boldsymbol{\beta}$ i.e.

$$\boldsymbol{x}_t = \underset{\boldsymbol{x}}{argmax}\, I(\boldsymbol{\beta}; \boldsymbol{y}|\boldsymbol{x}, \mathbb{D}_t^1). \tag{3.10}$$

The mutual information I is computed using the posterior distribution

$$p(\boldsymbol{\beta}|\mathbb{D}_t^1) = b_0 \prod_{t'=1}^{t-1} p(\boldsymbol{y}_{t'}|\boldsymbol{x}_{t'}, \boldsymbol{\beta})$$

with a k-sparse uniform prior b_0, assuming k is known to the agent. Unfortunately, computing I is cumbersome for sparsity $k > 1$ and RSI addresses this by iteratively identifying the most likely target locations from its belief assuming $k = 1$ at each iteration.

Sparse Parallel Asynchronous Thompson Sampling (SPATS) (Ghods et al., 2021b) is a multi-agent decentralized and asynchronous active search algorithm which uses Thompson sampling (TS) to determine the next sensing action, i.e.

$$\boldsymbol{x}_t^j = \underset{\boldsymbol{x}}{argmax}\, \mathbb{E}_{\boldsymbol{y}|\boldsymbol{x},\tilde{\boldsymbol{\beta}}}[-\|\tilde{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}(\mathbb{D}_t^j \cup \{\boldsymbol{x}, \boldsymbol{y}\})\|_2^2] \tag{3.11}$$

where $\tilde{\boldsymbol{\beta}}$ is a sample drawn from the posterior $p(\boldsymbol{\beta}|\mathbb{D}_t^j)$ which is a normal distribution assuming a block sparse prior. Unfortunately, SPATS is myopic in nature and relies on a carefully tuned block length reduction schedule in its posterior belief update to overcome the limitations of purely TS based exploration strategy in active search.

Sequential Point Sensing (PS) is an exhaustive coverage baseline where an agent starts from one corner on the grid and traverses every grid cell sequentially, executing point sensing actions to cover the entire search space. In the multi-agent case, every agent follows the same trajectory, so the incurred cost is expected to increase with larger team sizes due to repetitive sensing actions.

Remark 2. *Our purpose in contrasting these baselines with CAST is to demonstrate the benefit of an explicit cost-aware approach to multi-agent active search. To the best of our knowledge, there are no other cost-aware baselines for active search that we could compare to CAST. Moreover, pareto-optimality in multi-agent multi-objective decision making under uncertainty is not a well studied setting, so comparison to such algorithms is not considered given our proposed problem setup.*

Simulation setup.

In our experiments, we focus on 2-dimensional (2D) search spaces discretized into square grid cells of width 10m. An agent can move horizontally or vertically at a constant speed of 5m/s. Each sensing action incurs a fixed cost of c_s seconds (s), in addition to the travel time between sensing locations. We note that the cost-aware active search strategy may differ depending on the relative magnitudes of per action sensing cost and per unit travel cost. Hence, for each setting, we will vary $c_s \in \{0\text{s}, 50\text{s}\}$ to simulate high travel cost and high sensing cost respectively. Throughout, any agent is allowed to consider only hierarchical spatial pyramid sensing actions. Our goal is to estimate a k-sparse signal $\boldsymbol{\beta}$ by detecting all the k targets with J agents.

The search vector $\boldsymbol{\beta}$ is generated as a randomly uniform k-sparse vector in the search space. The agents are unaware of k and the generative prior. We set the signal to noise variance to 16. For CAST, we set $\gamma = 0.97$ and $\alpha_s = 0.5$. The hyperparameters in SPATS and RSI follow Ghods et al. (2021b); Ma et al. (2017). We allow the agents to continue searching the space until all targets have been recovered. Then, across 10 random trials we measure the mean and standard error (s.e.) of the total cost incurred by the team in recovering all k targets. We also plot the mean and s.e. of the full recovery rate achieved as a function of the total cost incurred. Recall that the full recovery rate is defined as the fraction of targets in $\boldsymbol{\beta}$ that are correctly identified. All agents start from the same location at one corner of the search space, fixed across trials. However, the exact instantiation of the search space varies across trials in terms of the position of the targets.

Remark 3. *The size of the action space in our experiments is larger than what MCTS algorithms commonly deal with, unless they are augmented with a neural policy network (Silver et al., 2016, 2017). Having a continuous state vector gives rise to additional challenges of exploding width at the belief nodes, making the tree too shallow to be useful and may cause collapse of belief representations resulting in overconfidence in the estimated policy. Further, the added observation noise would exacerbate these challenges. To address these, Section 3.5.4 outlines some of the implementation strategies for CAST in our setup.*

3.5.2 2D search space discretized into 8×8 grid cells

Table 3.2 compares the total cost incurred in fully recovering $\boldsymbol{\beta}$ at 2 different sparsity rates $k \in \{1, 3\}$ with $J = 4$ agents in an 8×8 search space. For CAST, we varied the tree depth $d_{\max} \in \{2, 3, 4\}$ and the number of simulation episodes $m \in \{5, 7.5, 10, 20\} \times 10^4$. Table 3.2 reports the results corresponding to the best performing $d_{\max}$ and m in each case. CAST-1 indicates the performance with one-step lookahead i.e. $d_{\max} = 1$ over $m = 20 \times 10^4$ episodes to emphasize the importance of multi-step lookahead over a finite horizon in our cost-aware algorithm. Fig. 3.3 plots the corresponding full recovery rate across trials as a function of the total cost incurred. Each agent can choose from $|\mathcal{A}| = 85$ region sensing actions over successive time steps. We observe that CAST outperforms SPATS, RSI and PS by incurring a lower total cost. RSI being information-greedy and modeling the assumption $k = 1$ in its hypothesis space is at an advantage in the single target setting. The stochastic nature of TS based active search in SPATS favours it in the multi-target setting when sensing is more expensive than travelling. Exhaustive coverage in PS is comparable only in a single target setting if travelling is expensive. But in cases that do not match their most favorable scenarios, all of them exhibit poor cost efficiency. In contrast, CAST's ability to perform adaptive lookahead plannning together with posterior sampling helps it achieve cost efficiency across single-target, multi-target and different cost scenarios.

Table 3.2: Total cost (s) (mean and s.e. over 10 trials) to achieve full recovery in an 8×8 grid with $J = 4$ agents.

Algorithm	k	$c_s = 0$s	$c_s = 50$s
CAST	1	**93.10 (12.35)**	**1268.98 (255.76)**
CAST-1		168.83 (12.40)	2125.62 (74.97)
SPATS		238.84 (39.42)	1570.91 (225.71)
RSI		124.24 (13.43)	1321.21 (124.18)
PS		186.61 (51.27)	4887.01 (1330.73)
CAST	3	**147.65 (16.11)**	**2392.21 (142.16)**
CAST-1		186.61 (8.13)	2678.65 (176.75)
SPATS		343.25 (44.69)	2454.76 (221.91)
RSI		233.12 (15.25)	2851.48 (182.40)
PS		368.93 (29.92)	9774.93 (843.40)

3.5.3 2D search space discretized into 8×16 grid cells

Table 3.3 compares the total cost incurred in fully recovering $\boldsymbol{\beta}$ at 2 different target sparsity rates $k \in \{1, 5\}$ with $J = 3$ agents in an 8×16 search space. Fig. 3.4 plots the corresponding full recovery rate across trials as a function of the total cost incurred. We fixed the tree depth in CAST at $d_{\max} = 2$ and the number of simulation episodes $m = 10^5$. Each agent can choose from $|\mathcal{A}| = 170$ region sensing actions over successive time steps. In almost all the settings, CAST outperforms SPATS, RSI and PS by choosing cost-aware actions that reduce its total incurred cost. When sensing is so expensive that traveling cost is negligible ($c_s = 50s$), especially in the single target setting, we observe that the information gain based algorithm RSI is at an advantage compared to the shallow lookahead in CAST. But when

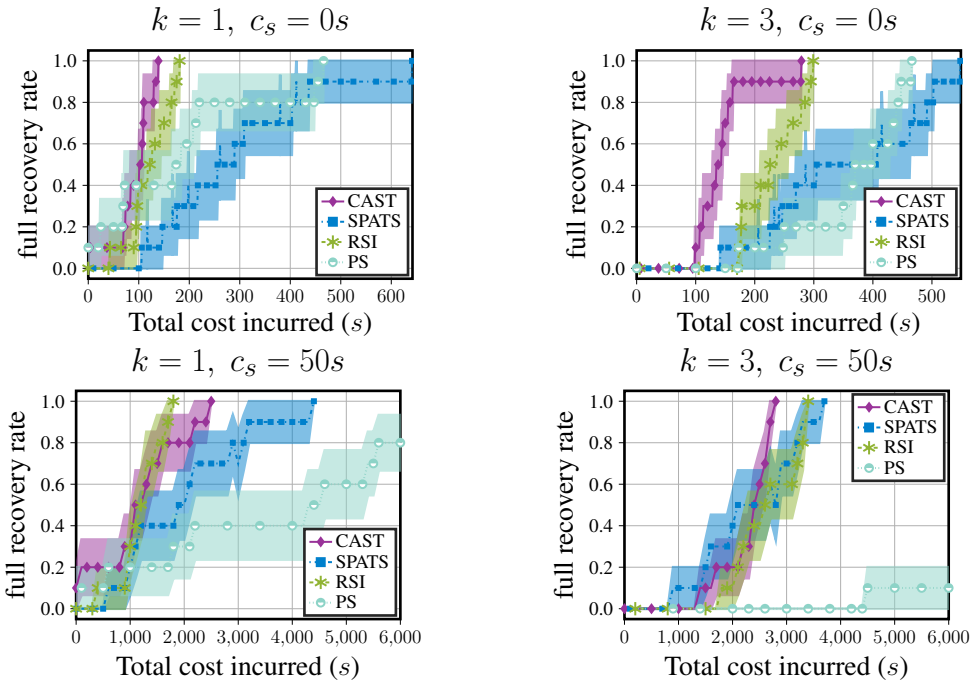


Figure 3.3: Full recovery rate versus total cost incurred in seconds in a 8×8 grid with $J = 4$ agents and k targets. Shaded regions indicate s.e.

Table 3.3: Total cost in seconds (mean and s.e. over 10 trials) to achieve full recovery in an 8×16 grid with $J = 3$ agents.

Algorithm	k	$c_s = 0$s	$c_s = 50$s
CAST	1	**110.09 (30.48)**	2616.90 (213.31)
SPATS		551.74 (120.52)	1931.28 (211.27)
RSI		135.56 (9.96)	**1166.69 (84.72)**
PS		183.80 (83.04)	7383.20 (2146.24)
CAST	5	**258.04 (16.55)**	**3705.41 (211.81)**
SPATS		1010.94 (89.86)	5250.94 (442.07)
RSI		398.01 (8.99)	4243.57 (95.50)
PS		631.80 (34.71)	16472.80 (894.30)

travelling is expensive, even with a lookahead horizon of 2 actions, CAST enables better cost efficiency than RSI. Moreover, exhaustive coverage in PS also incurs lower cost compared to SPATS when travelling is expensive, further indicating the need for cost awareness in active search.

3.5.4 Mitigating computational complexity in CAST

The computational complexity of CAST (Algorithm 3) increases with the maximum depth of the search tree ($d_{\max}$) and the size n of the search vector $\boldsymbol{\beta} \in \mathbb{R}^n$. For a larger n, the size of the action space being $O(n)$ (considering spatial hierarchical pyramid sensing

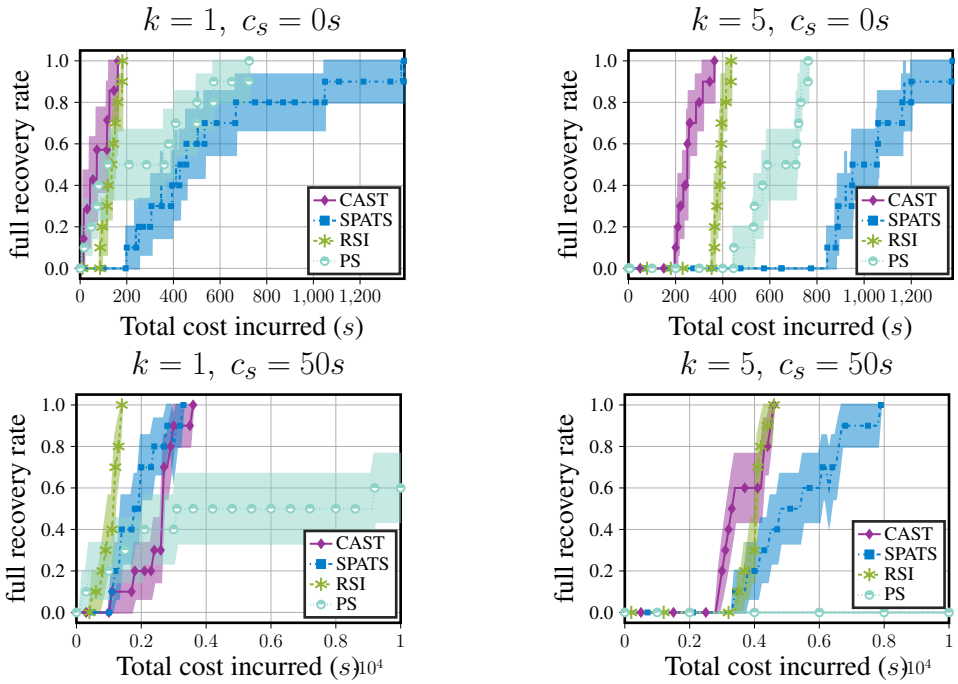


Figure 3.4: Full recovery rate versus total cost incurred in seconds in a 8×16 grid with $J = 3$. Shaded regions indicate s.e.

actions) implies that the tree policy as well as the new action node addition policy has to evaluate a larger set of feasible actions at every belief node encountered during the selection and expansion phase and this quickly becomes computationally expensive with increasing n. Moreover, increasing $n_{\max}$ further exacerbates the time complexity since it expands the space of lookahead action sequences and as a result, more episodes are required for effective exploration within the search tree. Additionally, as the tree width increases with completion of more episodes, it also leads to an increase in the pareto front computation and update time at each tree node. In what follows, we describe two heuristic strategies that we implemented to scale CAST to a 16×16 search space (results shown in Section 3.5.5).

Sampling from actions. In order to select the new action node to be added to the search tree (Line 19,Algorithm 3), we iterate over all feasible next actions at a belief node h (denoting the set by $\mathcal{A}_h$) and for each such action $\boldsymbol{x} \in \mathcal{A}_h$, we compute the change in entropy of the belief distribution per unit immediate cost incurred if $\boldsymbol{x}$ were executed. The action $\boldsymbol{x} \in \mathcal{A}_h$ with the maximum value of this quantity is selected and the tree expands to include the new action node (h, a). This strategy leads to more directed exploration within the search space than simple random sampling from $\mathcal{A}_h$. Unfortunately, it becomes computationally expensive as the size of the action space increases. Therefore, we propose sampling a subset $\mathcal{A}_h^s$ of size s from the feasible action pool ($\mathcal{A}_h^s \subset \mathcal{A}_h$) and select the action $\boldsymbol{x}' \in \mathcal{A}_h^s$ with the maximum change in entropy of the belief distribution per unit immediate incurred cost. This not only reduces the computational cost of CAST, it also introduces additional stochasticity in the search tree building phase in the multi-agent setting.

Pruning the tree. In contrast to the progressive widening strategy followed while adding children action nodes at the interior belief nodes in the search tree $\mathcal{T}_t$, we observed that CAST performs better when throughout the m episodes, the root node has as its children *all* the feasible action nodes at time step t. Although it helps our CAST-UCT tree policy to balance exploration-exploitation with the knowledge of the entire feasible action set, it would not be scalable in terms of the number of episodes needed as the size of the action space increases. Therefore, we propose a pruning technique using which we can prune the action nodes at the root level of $\mathcal{T}_t$ after a pre-determined number of simulation episodes are completed. Note that the particular episodes when we prune the tree is a tunable hyperparameter that also determines the cost-aware performance. In order to achieve this, in the backpropagation phase of each episode, we additionally maintain the upper confidence bound (UCB) based reward-cost pareto front using the backed up values. In any episode m', the 1-step lookahead reward λ^- is computed at a belief node h. We maintain the UCB (r_h^{UCB}) of these rewards over $n(h)$ episodes using the Student's t-distribution to compute a 95% confidence interval. c_h is the immediate cost of executing the action that would result in transitioning to belief node h. At the leaf node h_ℓ for episode m', we define the UCB based immediate reward-cost vector $\boldsymbol{g}'_{h_\ell} = \begin{bmatrix} r_{h_\ell}^{UCB} & -c_{h_\ell} \end{bmatrix}^{\mathrm{T}}$. To distinguish it from the LCB based vector $\boldsymbol{g}_h$ defined in Section 3.4, we will refer to $\boldsymbol{g}'_h$ as our UCB pruning vector. During backpropagation, at each tree node visited in m', we update the UCB based lookahead reward-cost pareto-front in the same way as described for the LCB in Section 3.4. Assuming that T_t is to be pruned at the m''-th episode, we remove all those child actions at the root whose UCB based pareto-front is dominated by the LCB based pareto-front over all actions at the root. This pruned T_t is then used over subsequent simulation episodes at time t.

Following are experimental results in a 2D 16×16 search space following the implementation strategies outlined above for computational efficiency.

3.5.5 2D search space discretized into 16×16 grid cells

Fig. 3.5 shows the mean and s.e. of the full recovery rate versus total cost incurred over 10 trials with J agents looking for $k = 5$ targets in a 16×16 search space. We vary the team size $J \in \{4, 8, 12\}$. Table 3.4 indicates the corresponding mean and s.e. of the total cost to correctly detect all targets. CAST simulates $m = 25000$ episodes with a lookahead horizon of 2 actions ($d_{\max} = 2$). Each agent can choose from $|\mathcal{A}| = 341$ region sensing actions over successive time steps.

We observe that CAST outperforms SPATS, RSI and PS, incurring a lower cost and a higher full recovery rate across different team sizes and cost scenarios. RSI is information greedy and deterministic in its decision making, so all agents choose the same actions leading to an increasing total cost with larger team sizes. On the other hand, the stochastic nature of TS based active search in SPATS is suited to the asynchronous and decentralized multi-agent setup and becomes competitive especially when sensing actions are more expensive than travelling ($c_s = 50s$) which aligns best with the objective of active information gathering. Exhaustive coverage in PS is comparable only with a smaller team size in case when travelling is expensive ($J = 4, c_s = 0s$) but outperforms SPATS in that setting, showing the need for cost-awareness in active search. Unfortunately in cases that do not match their most favorable scenarios, all of these algorithms exhibit poor cost efficiency. In contrast, the cost-aware agents using CAST's posterior sampling based lookahead pareto-

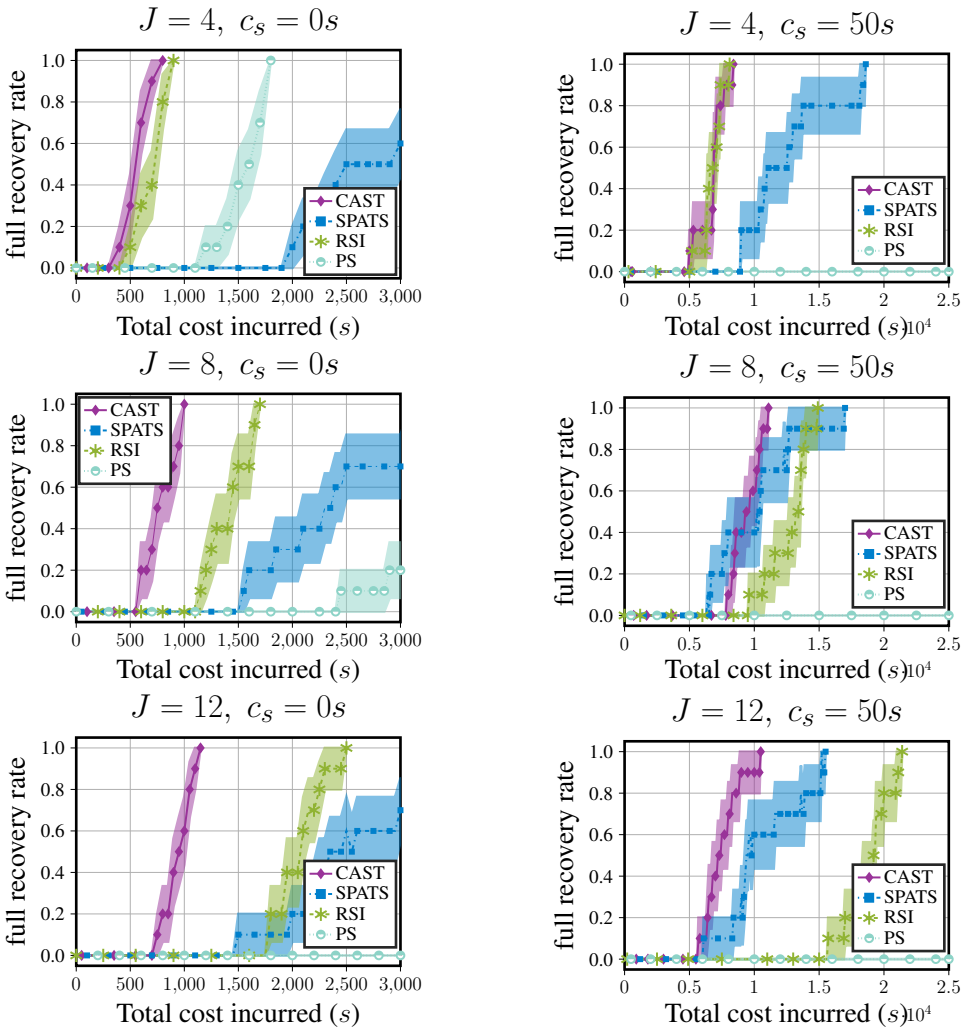


Figure 3.5: Full recovery rate versus total cost incurred in seconds in a 16×16 grid with J agents, $k = 5$ targets. Shaded regions indicate standard error over 10 trials. Compared to baselines, CAST achieves a full recovery rate of 1 at a lower total cost, both when traveling is costlier than sensing ($c_s = 0$s) and when sensing is more expensive ($c_s = 50$s).

Table 3.4: Total cost (mean and s.e. over 10 trials) to achieve full recovery in a 16×16 grid with J agents, $k = 5$ targets. CAST outperforms all other baselines for different team sizes and different relative costs for travelling and sensing.

Algorithm	J	$c_s = 0$s	$c_s = 50$s
CAST	4	**655.9 (39.4)**	**6852.3 (314.1)**
SPATS		2988.8 (285.6)	12563.8 (1132.7)
RSI		797.4 (37.2)	6862.4 (252.8)
PS		1654.1 (64.4)	42753.3 (1742.6)
CAST	8	**827.0 (48.4)**	**9529.7 (350.6)**
SPATS		2482.3 (255.7)	10242.3 (1033.6)
RSI		1455.5 (59.8)	12815.5 (513.6)
PS		3414.9 (143.7)	88839.9 (3735.3)
CAST	12	**991.6 (39.6)**	**7647.59 (445.4)**
SPATS		2699.2 (240.1)	10764.2 (948.8)
RSI		2118.9 (71.0)	19023.9 (551.1)
PS		4827.0 (167.4)	125582.0 (4352.2)

optimal planning and stochastic decision making are able to achieve cost efficiency across different cost scenarios with teams of varying sizes.

We also evaluate the robustness of CAST by comparing the total cost incurred to correctly identify all targets as the number of targets increases in the search space. Table 3.5 and Fig. 3.6 show that CAST not only outperforms all others across multi-target and cost scenarios, additionally the total cost incurred is hardly affected by k since CAST enables cost awareness through decentralized decision making independent of team size J and sparsity rate k. In contrast, SPATS being myopic in nature exhibits more randomness in the actions selected, whereas RSI approximates its mutual information objective assuming $k = 1$, thereby requiring more sensing actions to recover all targets as k increases.

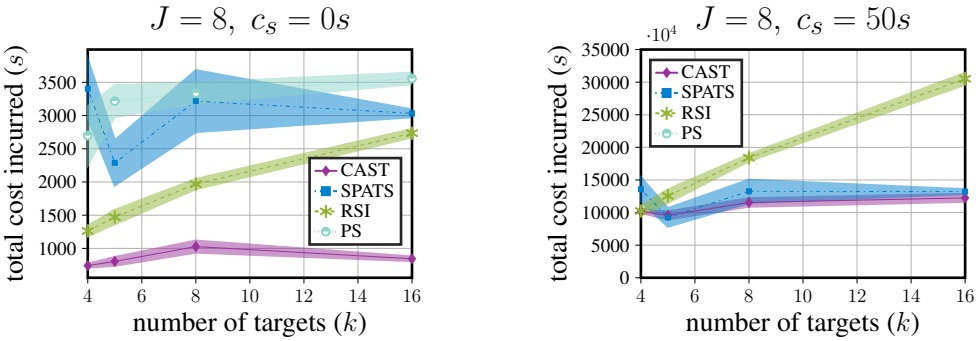


Figure 3.6: Total cost incurred versus number of targets $k \in \{4, 5, 8, 16\}$ in a 16×16 grid with $J = 8$ agents. Baseline PS is excluded on the right for better visualization.

3.5.6 Comparison with myopic cost-aware variations of SPATS

As discussed earlier in Section 3.5.1, to the best of our knowledge, there are no existing cost-aware active search baselines to compare against CAST. We therefore modify the

Table 3.5: Total cost (mean and s.e. over 5 trials) to achieve full recovery in a 16×16 grid with $J = 8$ agents, k targets. For different number of targets, CAST incurs similar total cost for the same grid size and teamsize. CAST also incurs a lower cost compared to baselines.

Algorithm	k	$c_s = 0$s	$c_s = 50$s
CAST	4	**740.7 (26.5)**	**8130.4 (293.1)**
SPATS		3404.4 (432.9)	13574.4 (1792.2)
RSI		1262.8 (73.4)	10242.8 (685.8)
PS		2698.3 (438.6)	70208.3 (11404.6)
CAST	8	**735.3 (57.9)**	**9157.0 (476.7)**
SPATS		3217.2 (461.5)	13267.2 (1737.3)
RSI		1968.4 (72.2)	18398.4 (500.3)
PS		3339.9 (151.7)	86889.9 (3945.2)
CAST	16	**843.9 (29.9)**	**8880.8 (316.2)**
SPATS		3032.7 (54.3)	13212.7 (321.0)
RSI		2734.4 (58.9)	30524.4 (871.3)
PS		3559.1 (83.7)	92589.1 (2176.8)

cost-agnostic myopic active search baseline SPATS (Ghods et al., 2021b) to incorporate cost-awareness in two different ways.

SPATS-scalarize. First, we consider a scalarized cost-aware version of the SPATS decision making objective:

$$\boldsymbol{x}_t^j = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, \lambda_r \mathbb{E}_{y|\boldsymbol{x},\tilde{\boldsymbol{\beta}}}[-\|\tilde{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}(\mathbb{D}_t^j \cup \{\boldsymbol{x}, y\})\|_2^2] - \lambda_d c_d(x_{t-1}^j, x) - \lambda_s c_s \tag{3.12}$$

Since the sensing cost c_s is a constant for all actions, it does not affect the action selected for different c_s. Instead, the agent will optimize only for the reward vs. travel cost, weighted by the reward coefficient λ_r and the travel cost coefficient λ_d. λ_s is the sensing cost coefficient, such that $\lambda_r + \lambda_d + \lambda_s = 1$.

SPATS-pareto. Second, we consider a pareto-optimization approach to augment the myopic decision making step of a SPATS agent. For each feasible action $\boldsymbol{x}$, the agent constructs a myopic reward-cost vector: $\begin{bmatrix} \mathbb{E}_{y|\boldsymbol{x},\tilde{\boldsymbol{\beta}}}[-\|\tilde{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}(\mathbb{D}_t^j \cup \{\boldsymbol{x}, y\})\|_2^2] \\ -(c_d(x_{t-1}^j, x) + c_s) \end{bmatrix}$. The pareto-front over such reward-cost vectors would exclude all actions with a lower reward *and* a higher cost. Then the agent selects the action from the pareto-front having the maximum one-step reward per unit cost:

$$\boldsymbol{x}_t^j = \underset{\boldsymbol{x}}{argmax}\, \frac{\mathbb{E}_{y|\boldsymbol{x},\tilde{\boldsymbol{\beta}}}[-\|\tilde{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}(\mathbb{D}_t^j \cup \{\boldsymbol{x}, y\})\|_2^2]}{c_d(x_{t-1}^j, x) + c_s}. \tag{3.13}$$

Unlike SPATS-scalarize, SPATS-pareto does not depend on the chosen λ_d.

In Table 3.6 and Table 3.7, we observe that CAST still outperforms these modified cost-aware myopic baselines, SPATS-scalarize and SPATS-pareto, across different number of targets, different team sizes and different cost scenarios. For SPATS-scalarize, when

c_s=0s, grid search over λ_d indicates minimum cost incurred for λ_d=0.5, so we set λ_d=0.5, λ_r=0.5 in our experiments. λ_s does not affect the action selected, so we set λ_s=0. When c_s=50s, SPATS-scalarize with λ_d=0.5 selects the same actions as when c_s=0s, thus incurring a noticeably higher cost compared to cost-agnostic SPATS with λ_d=0,λ_s=0. SPATS-pareto selects the action from its pareto-front maximizing the one-step reward per unit cost (Eq. (3.13)), which as we discussed in Section 3.4 would prefer actions with a higher incurred cost for the same reward. As a result, SPATS-scalarize outperforms SPATS-pareto in the same myopic decision making setting. In contrast, lookahead planning in CAST enables cost-aware action selection and incurs a lower cumulative cost than these baselines. CAST shows noticeable performance gain especially across different team sizes (J) with a higher sensing cost (c_s = 50s) or more number of targets (k) in the search space. These observations therefore imply the need for careful consideration of how cost-awareness is incorporated in multi-agent active search and validate our proposed new algorithm CAST in this setting.

Table 3.6: Total cost (mean and s.e. over 10 trials) to achieve full recovery in a 16×16 grid with J agents, $k = 5$ targets. CAST outperforms the myopic cost-aware modifications of SPATS across different team sizes and cost scenarios.

Algorithm	J	$c_s = 0$s	$c_s = 50$s
CAST	4	**655.9 (39.4)**	**6852.3 (314.1)**
SPATS-scalarize		673.5 (61.5)	13853.6 (845.4)
SPATS-pareto		693.7 (33.5)	23470.7 (1341.9)
CAST	8	**827.0 (48.4)**	**9529.7 (350.6)**
SPATS-scalarize		851.2 (53.4)	16296.8 (956.8)
SPATS-pareto		1018.0 (74.8)	26783.4 (1392.1)
CAST	12	**991.6 (39.6)**	**7647.6 (445.4)**
SPATS-scalarize		1001.2 (64.7)	18598.6 (1067.6)
SPATS-pareto		1122.4 (146.6)	32472.2 (1487.6)

3.5.7 Visualizing cost-aware and cost-agnostic agent behavior

For further visualization, we refer to the webpage here[1] demonstrating the multi-agent active search behavior of CAST, SPATS and RSI with J=4 agents and k=3 targets, c_s=$\{0\text{s}, 50\text{s}\}$ in an 8×8 search space. CAST shows a distinct change in the nature of sensing actions due to its cost-awareness. When sensing is expensive (c_s = 50s), the CAST agents initially prefer broader sensing actions to quickly adjust their belief regarding possible target positions in the search space; whereas when traveling is more expensive (c_s = 0s), the CAST agents are more conservative in their movement and tend to gradually make their way through the search space with smaller region sensing actions that cover positions adjacent to their current location.

[1] https://sites.google.com/view/cast-multiagent/home

Table 3.7: Total cost (mean and s.e. over 5 trials) to achieve full recovery in a 16×16 grid with $J = 8$ agents, k targets. CAST outperforms the myopic cost-aware modifications of SPATS for different number of targets in the search space, with increasing cost-efficiency in comparison for higher k.

Algorithm	k	$c_s = 0$s	$c_s = 50$s
CAST	4	**740.7 (26.5)**	**8130.4 (293.1)**
SPATS-scalarize		777.5 (66.7)	15522.7 (1563.5)
SPATS-pareto		909.0 (78.5)	26157.2 (1805.2)
CAST	8	**735.3 (57.9)**	**9157.0 (476.7)**
SPATS-scalarize		884.8 (67.5)	19669.7 (944.2)
SPATS-pareto		1010.4 (55.9)	29126.9 (752.3)
CAST	16	**843.9 (29.9)**	**8880.8 (316.2)**
SPATS-scalarize		976.6 (44.6)	21328.7 (918.3)
SPATS-pareto		1049.6 (49.5)	29651.4 (870.2)

Part II

Uncertainty awareness in decision making

Extracting necessary information from noisy data is a central theme and of crucial importance in almost all of science and engineering. In this part of the thesis, we focus on the challenges due to observation noise in multi-agent active search. Specifically, in Chapter 4 we consider observation noise due to different sensors providing detection and location measurements to the agents, which requires joint modeling of epistemic uncertainty in the agent's posterior belief. Next, in Chapter 5 we consider observation noise when targets are moving in the environment, which requires the agents to appropriately model uncertainty over the target search space for adaptive search and tracking.

4 | Observation noise in target detection and location measurements

4.1 Introduction

Agents that are tasked with searching for targets in an unknown environment observe their surroundings with sensors like RGB-D cameras, Lidar, Sonar, etc. But observations received by agents are not always accurate, rather the sensor measurements are corrupted by noise. A part of this observation noise can be inherent to the sensor being used and accounts for aleatoric uncertainty in posterior inference, whereas a part of it can be i.i.d random. This epistemic uncertainty arising from the agent's noisy observations due to the latter (i.e. random noise) can be reduced with more observations of the same surroundings or regions. This implicitly assumes that the targets and other objects in the surroundings being observed are stationary, so subsequent observations only differ in terms of the i.i.d random added noise. In contrast, if there are targets moving in the environment, the agent must also account for the uncertainty in its posterior belief arising from such non-stationarity in its surroundings. Since the agents in our active search setup have partial observability over the search space, it further exacerbates the challenges in target recovery from observations with different sources of observation noise.

Prior work studying the problem of state estimation from noisy observations can be categorized into two domains focusing respectively on *target location* and *detection uncertainty*, typically independent of each other. Due to noise in the sensing setup, an agent may mistake something that is not a target to be an object of interest (OOI) or vice versa, thereby giving rise to detection uncertainty. On the other hand, errors in the depth sensor measurements may cause an agent to perceive OOIs to be located nearer to or farther from itself than they actually are, leading to uncertainty about the target's true location. In this chapter, we instead propose an approach for observation noise-aware posterior inference and decentralized asynchronous decision making that can jointly account for uncertainty in both target detection and location.

4.2 Problem Formulation

Consider multiple ground robots searching a space to locate some OOIs (Fig. 4.1a). Each robot moves around and observes certain regions of the search space by taking pictures to

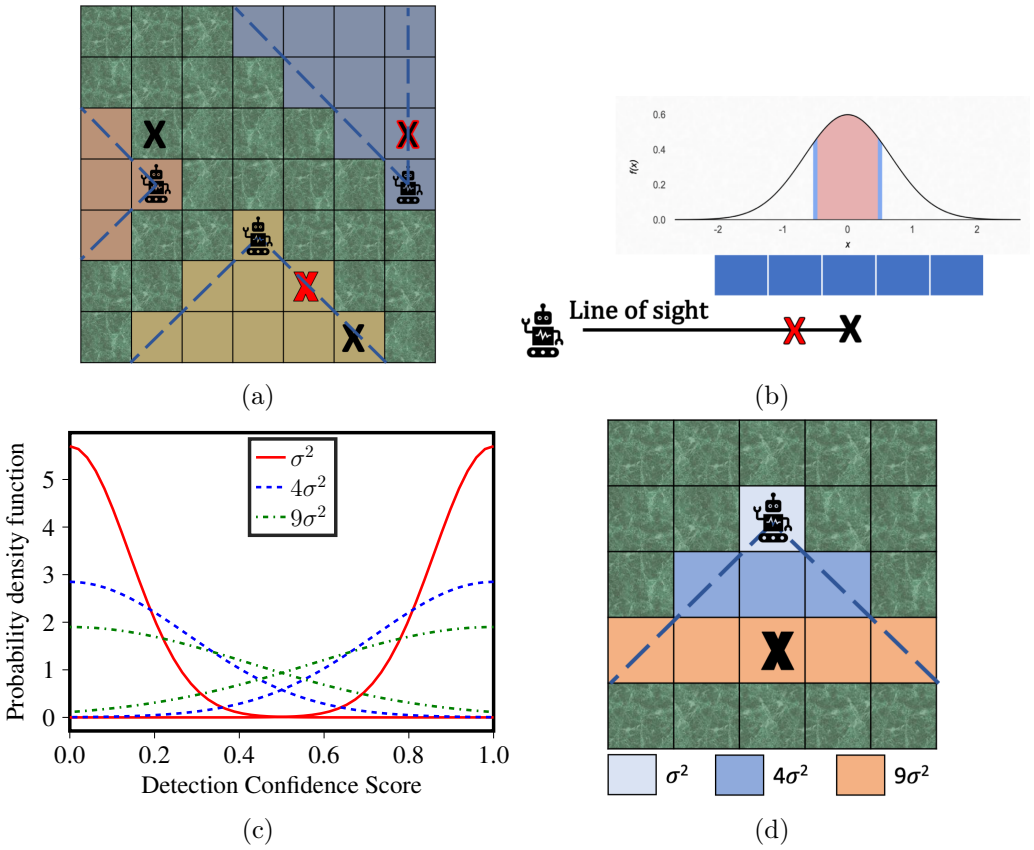


Figure 4.1: **Problem setup.** (a) Multiple agents sense different parts of the environment looking for OOIs. True OOIs are crossed in black. Targets detected by the agent in its field of view are crossed in red. (b) Due to location uncertainty, the observed depth of the true OOI (" × " in black) is shifted towards the agent, to the grid cell marked " × " in red. (c) An object detector becomes less confident about positive (1) as well as negative (0) labels as its distance to the object increases (Eq. (4.1)). (d) Illustration of the depth dependant variance levels for target detection in the FOV of a ground robot in our setup.

detect OOIs and measuring their distances from its current location. We assume the robots can localize themselves accurately. The colored cells in Fig. 4.1a illustrate each robot's sensing action using a camera with a 90° field of view (FOV). Each robot independently decides its next sensing action given its current belief about OOIs in the search space.

Once a robot senses a region, it obtains information about both the presence (detection) and corresponding depth (location) of objects in its FOV. For example, RGB-D sensors coupled with an object detector like YOLOv3 (Redmon and Farhadi, 2018) can be used to extract such information. The object detector identifies OOIs with a confidence score that varies with distance from the camera. Objects farther away from the camera usually have a lower probability of being correctly identified. This gives rise to target detection uncertainty in the robot's observation. Further, the depth sensor is prone to error in the measured distance to the OOI. This leads to target location uncertainty in the robot's observation. Our objective is to account for both the detection and location uncertainty in our observation model and utilize it to make improved active sensing decisions.

4.2.1 Ground truth model for target detection uncertainty

Let $\xi_i \in \{0, 1\}$ denote the output of an object detector with perfect detection ability which labels an object i at a distance l_i away from the camera accurately with either a '0' (not OOI) or a '1' (OOI). Typically, the confidence score of the object detector gradually declines as a function of the OOI's distance from the camera. Prior work in Ghods et al. (2021a) has characterized this behavior using a depth aware detection model where the output y_i of an imperfect object detector is the true object label modified by an additive one-sided Gaussian noise:

$$y_i = \xi_i + n_i^d, \text{ with } n_i^d \sim \mathcal{N}^+(0, \sigma_i^2(l_i)). \tag{4.1}$$

Therefore, $y_i \in [0, 1]$ follows a one-sided normal distribution centered at ξ_i=0 for a true negative and ξ_i=1 for a true positive label. The variance $\sigma_i^2(\cdot)$ is an increasing function of l_i (Fig. 4.1d). So the probability of a false negative or a false positive OOI detection increases at distances farther away from the camera. This is illustrated in Fig. 4.1c where the horizontal axis indicates the detector's y_i and the curves indicate the varying probability densities at different object distances l_i.

4.2.2 Ground truth model for target location uncertainty

Let $\zeta_i \in \mathbb{R}$ denote the measurement from an accurate depth sensor of its true distance to an object i. Real depth measurements have error along the agent's line of sight. Prior work in Belhedi et al. (2012) has characterized the error as a Gaussian distribution. In our setup, we model the target location uncertainty as additive Gaussian noise so that

$$y_i = \zeta_i + n_i^\ell, \text{ with } n_i^\ell \sim \mathcal{N}(0, r_u^2) \tag{4.2}$$

where $y_i \in \mathbb{R}$ is the measurement from an imperfect depth sensor and r_u parameterizes the uncertainty of such measurements along the agent's line of sight. As depicted in Fig. 4.1b, the measured depth follows a normal probability distribution centered on the true target location.

4.2.3 Sensing model

We consider a gridded search environment described by a sparse matrix which we want to recover through multi-agent active search. M is the total number of grid cells. $\boldsymbol{\beta} \in \{0,1\}^{M\times 1}$ denotes the flattened vector representation of the environment having k non-zero entries at the true locations of the k OOIs. $\mathbf{X}_t \in \{0,1\}^{Q_t \times M}$ is the sensing action at time t. Q_t is the number of grid cells covered by the robot's FOV under $\mathbf{X}_t$. Each row of $\mathbf{X}_t$ is a one hot vector $\{0,1\}^{1\times M}$ indicating the position of one of the sensed grid cells in the robot's FOV. $\mathbf{X}_t\boldsymbol{\beta} \in \{0,1\}^{Q_t\times 1}$ is the ground truth observation due to sensing action $\mathbf{X}_t$. $\mathbf{y}_t \in \mathbb{R}^{Q_t\times 1}$ is the agent's observation vector indicating the noisy sensor measurement from executing $\mathbf{X}_t$. The sensing model is

$$\underbrace{\mathbf{y}_t}_{\text{noisy measurement}} = \underbrace{\mathbf{X}_t\boldsymbol{\beta}}_{\text{ground truth observation}} + \underbrace{\mathbf{n}_t}_{\text{measurement noise}} \tag{4.3}$$

where $\mathbf{n}_t \in \mathbb{R}^{Q_t\times 1}$ is composed of the noise from (4.1), (4.2).

Remark 1. Note that the model described above is what our simulator considers to be ground truth. Our algorithm and its agents are neither aware of the number of targets nor their true locations, and only receive the measurement $(\mathbf{X}_t, \mathbf{y}_t)$.

4.2.4 Communication

We assume that communication, although unreliable, will be available sometimes and the agents should communicate when possible. The agents share their measurements asynchronously, and do not wait on inter-agent communications. Further, the set of available past measurements need not remain consistent across agents due to communication unreliability.

Remark 2. Since our goal is uncertainty-aware active sensing and not planning a continuous path, we only require a coarse discretization of our environment. For example, in Section 4.7, we cover a 250m×250m region with square grid cells of size 15m. We also assume that individual OOIs occupy an entire grid cell and the location uncertainty from the depth sensor only affects which cell was determined to have the OOI, not the OOI's placement within the cell.

To recover the search vector $\boldsymbol{\beta}$ by actively identifying all the OOIs, at each time t, an agent j chooses the best sensing action $\mathbf{X}_t^j$ based on its belief about the OOIs given the measurements available thus far in its measurement set $\mathbf{D}_t^j$. Assuming that all the agents collectively obtain T measurements, our objective is to correctly estimate the sparse vector $\boldsymbol{\beta}$ with as few measurements T as possible. For a single agent, our problem reduces to sequential decision making with the measurement set $\mathbf{D}_t^1 = \{(\mathbf{X}_1, \mathbf{y}_1), \ldots, (\mathbf{X}_{t-1}, \mathbf{y}_{t-1})\}$ available to the agent at time t. In the multi-agent setting, following our communication constraints, we use a decentralized and asynchronous parallel approach with agents independently deciding individual sensing actions (Kandasamy et al., 2018; Ghods et al., 2021a,b).

4.3 Related Work

Target location uncertainty. The issue of location uncertainty has been studied by the robotics community, particularly in tasks such as path planning, localization and tracking

(Caglioti et al., 2006; Stroupe et al., 2001). Such problems are typically approached using filtering algorithms that can recursively estimate the robot or target state and the associated location uncertainty, while abstracting away the detection uncertainty by thresholding the detection probability to zero or one. Sensor measurements from RGB cameras and depth sensors used in typical robotic tasks like simultaneous localization and mapping (SLAM) and navigation often introduce data association error due to pose mismatch between the corresponding sensors (Basso et al., 2018). Further, depth images suffer from position dependent geometric distortions and distance dependent measurement bias (Belhedi et al., 2012). This also introduces location uncertainty in the sensor measurements. As a result several sensor calibration algorithms have been proposed which mainly focus on parameterizing the error in the depth measurements and then learning those parameters from carefully collected training data (Zhou and Koltun, 2014; Zuñiga-Noël et al., 2019; Chen et al., 2019a; Miller et al., 2013; Basso et al., 2014). However such approaches typically engineer away the need to account for detection uncertainty by using some predetermined visual patterns in the collected dataset. Moreover, in contrast to our problem setup, they do not focus on learning how to autonomously collect appropriate data for reducing the location uncertainty in the sensor measurements.

Our goal of estimating the number and location of targets through sensing actions has some similarities to the robotics problem of SLAM for mapping an unknown environment while estimating a robot's pose within it. Particularly, the data association problem in SLAM is concerned with matching noisy sensor observations with map landmarks (landmarks being identified can be thought as similar to recovering targets in active search) and this problem setup has been studied extensively in the SLAM literature (Cadena et al., 2016). Recently, Zhang et al. (2023) focused on this matching problem with an unknown number of landmarks and unknown prior data association, and proposed a nested optimization algorithm which iterates over the number of landmarks k and assuming a certain k, optimizes the batch assignment of landmarks from noisy measurements. But their optimization method is only tractable assuming isotropic noise (a simpler noise model than Section 4.2 and Section 4.4) and it was also not observed to be robust to increasing noise levels. Moreover, there has been little focus on actively controlling the agent's trajectory for improved data association, or on decentralized multi-agent approaches for active SLAM (Placed et al., 2023).

Target detection uncertainty. The domain of sparse signal recovery is primarily concerned with the problem of choosing sensing actions in the face of detection uncertainty, assuming that once the signal is detected then its location is accurate. Prior work in this area has proposed algorithms that use principles from compressed sensing together with constrained optimization to estimate the signal (Carmi et al., 2010; Needell and Vershynin, 2010). Ma et al. (2017) formulated this as an active search problem with realistic region sensing actions and Igoe et al. (2022) proposed a reinforcement learning approach incorporating the detection uncertainty in the observations into a POMDP framework. Ghods et al. (2021b,a) build on the former setup and go on to model detection uncertainty as a function of the target's distance from the sensing agent.

The discrete spatial search problem is also studied in search theory by considering detection uncertainty through false positive detections in sensing individual cells (Kress et al., 2008; Chung and Burdick, 2012; Cheng et al., 2019) but they do not relate the

uncertainty with sensor capabilities and do not generalize to realistic region sensing actions.

Passive uncertainty modeling. The task of characterizing the uncertainty of detection and location of objects in an image has also been extensively studied in computer vision (Hall et al., 2020; Kampffmeyer et al., 2016; Gonzalez-Garcia et al., 2015; Caicedo and Lazebnik, 2015; Wang et al., 2020). In contrast, our aim is to adaptively choose which images to capture (i.e. decide where to sense and in which direction) by considering the associated uncertainties so that the agents can physically locate the OOIs in a search space.

4.3.1 Naïve approach and challenges

In the context of the prior related work described above and our discussion of the multi-agent active search setting so far, one might consider a preliminary approach to jointly accounting for detection and location noise would be to simply plug-in the sensing model from Eq. (4.3) to the SPATS algorithm (Chapter 2). Leaving aside the differences in action space for UAVs in SPATS compared to the ground robots (Section 4.2), recall that we assume a discretized search space, which would require carefully addressing how the (continuous) location noise affects the observed target location. Moreover, the expectation-maximization approach to posterior belief update in Ghods et al. (2021b) depends on a tunable hyperparameter specific to the block-sparsity assumption which does not apply to the ground robot's action space. Keeping in mind these differences, we observed the need for a novel approach to model the joint detection and location noise in this setting.

We now describe our approach to the multi-agent active search problem described in Section 4.2 in two stages. First, in Section 4.4, we outline our inference procedure that agents use to identify OOIs and estimate their locations given the set of available measurements thus far. Next, in Section 4.6, we describe our decentralized and asynchronous multi-agent decision making algorithm that utilizes the estimates from the proposed inference method.

4.4 UnIK: Uncertainty-aware Inference using Kalman filter

4.4.1 Belief representation

Following Section 4.2, we want to recover the ground truth search vector $\boldsymbol{\beta}$ by identifying all the OOIs. Both the number of targets and their locations are unknown to the agents. We therefore initialize each agent with a Gaussian prior over $\boldsymbol{\beta}$, denoted by $p_0(\boldsymbol{\beta}) = \mathcal{N}(\hat{\boldsymbol{\beta}}_0, \mathbf{P}_0)$. Given the measurement set $\mathbb{D}_t^j$ available to the agent j at time t, we use a Kalman filter (Kalman, 1960) to update its posterior belief $p(\boldsymbol{\beta}|\mathbb{D}_t^j) = \mathcal{N}(\hat{\boldsymbol{\beta}}_t^j, \mathbf{P}_t^j)$.

Kalman filter (KF) is a recursive linear state estimator that minimizes the mean squared error between the predicted and true measurements (Kalman, 1960). It operates in two main steps: *prediction*, where the belief state estimate is propagated forward using a dynamics model, and *update*, where this predicted estimate is corrected based on the agent's noisy observations. KF assumes Gaussian observation noise and relies on covariance matrices to balance uncertainty between the model and measurements. It is widely used in applications like navigation, robotics, and signal processing where realtime state estimation is crucial.

In our problem setup, an agent j at time t maintains a belief over the search vector $\boldsymbol{\beta}$ based on its measurements $\mathbb{D}_t^j$. Note that the targets are static in this setting. Therefore, we define the Kalman filter *process model* for our search vector $\boldsymbol{\beta}$ as $\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1}$. The linear

measurement model is $\boldsymbol{z}_t = \boldsymbol{x}_t\boldsymbol{\beta}_t + \boldsymbol{\nu}_t$, where the noise variable $\boldsymbol{\nu}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{z}_t})$. Assuming a uniform Gaussian prior over $\boldsymbol{\beta}$, $p_0(\boldsymbol{\beta}) = \mathcal{N}\Big(\hat{\boldsymbol{\beta}}_0, \hat{\mathbf{P}}_0\Big)$, Kalman filtering recursively updates the belief $p_t^j\Big(\boldsymbol{\beta}|\mathbb{D}_t^j\Big) = \mathcal{N}\Big(\hat{\boldsymbol{\beta}}_t, \hat{\mathbf{P}}_t\Big)$ using the following alternate *prediction* and *update* steps.

$$\textit{Prediction: } \hat{\boldsymbol{\beta}}_t^- = \hat{\boldsymbol{\beta}}_{t-1},\ \hat{\mathbf{P}}_t^- = \hat{\mathbf{P}}_{t-1} \tag{4.4}$$

$$\textit{Kalman gain: } \mathbf{K}_t = \hat{\mathbf{P}}_t^- \boldsymbol{x}_t^{\mathrm{T}}(\boldsymbol{x}_t\hat{\mathbf{P}}_t^-\boldsymbol{x}_t^{\mathrm{T}} + \boldsymbol{\Sigma}_{\boldsymbol{z}_t} + \lambda_t\mathbf{I})^{-1} \tag{4.5}$$

$$\textit{Update: } \hat{\boldsymbol{\beta}}_t = \hat{\boldsymbol{\beta}}_t^- + \mathbf{K}_t(\boldsymbol{y}_t - \boldsymbol{x}_t\hat{\boldsymbol{\beta}}_t^-) \tag{4.6}$$

$$\hat{\mathbf{P}}_t = (\mathbf{I} - \mathbf{K}_t\boldsymbol{x}_t)\hat{\mathbf{P}}_t^-(\mathbf{I} - \mathbf{K}_t\boldsymbol{x}_t)^{\mathrm{T}} + \mathbf{K}_t\boldsymbol{\Sigma}_{\boldsymbol{z}_t}\mathbf{K}_t^{\mathrm{T}} \tag{4.7}$$

Here λ_t is a regularization constant. As before, to recover all the OOIs, the agent's MAP belief mean estimate $\hat{\boldsymbol{\beta}}$ must identify the number and location of all targets in the unknown search vector $\boldsymbol{\beta}$.

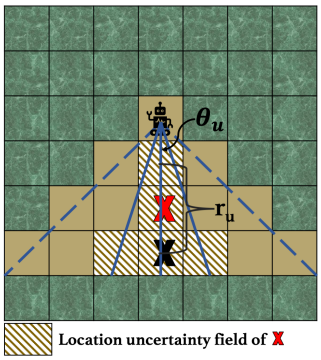
4.4.2 Overview of UnIK

Our approach to jointly accounting for target detection and location uncertainty in noisy sensor observations is based on suitably constructing the measurement noise covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{z}_t}$ to update the posterior Kalman filter estimates (Eqs. (4.4) to (4.7)). First, we initialize $\boldsymbol{\Sigma}_{\boldsymbol{z}_t} = \mathrm{diag}\Big([\sigma_q^2(l_q)]_{q=1}^{Q_t}\Big)$ to account for the OOI detection uncertainty in the robot's FOV using the distance dependent variance from Eq. (4.1). Next, the agent estimates the possible OOI locations in its FOV by thresholding the noisy observation $\boldsymbol{y}_t$: $\boldsymbol{y}_{thr,t} = \mathbf{1}(\boldsymbol{y}_t >= c_{thr})$. For each possible OOI, the agent then determines its location uncertainty field i.e. the grid cells in its FOV where this OOI might be truly located.

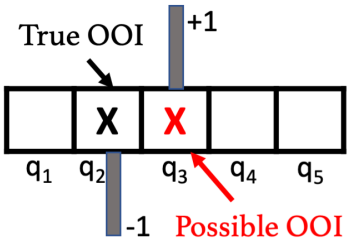
Following Fig. 4.2a, suppose $\boldsymbol{y}_{thr,t}$ indicates an OOI in the position marked " $\times$ " in red, whereas the OOI is actually located in the grid cell marked " $\times$ " in black. Along its line of sight to the observed OOI location (" $\times$ " in red), the agent computes the *location uncertainty field* using the radial uncertainty parameter r_u (same as Eq. (4.2)) and an angular uncertainty parameter θ_u shown in Fig. 4.2a. It will include all the grid cells in the region bounded radially by $[-r_u, r_u]$ around the observed OOI location within an angular spread of $[-\theta_u, \theta_u]$ relative to the line of sight to the agent. After that, the location uncertainty field for each possible OOI index in $\boldsymbol{y}_{thr,t}$ is used to compute the variance and covariance due to location uncertainty in $\boldsymbol{\Sigma}_{\boldsymbol{z}_t}$ and we describe this next.

Table 4.1: Modeling the probability distribution for target location uncertainty in the measurement model using the observed target location at $\tilde{q}_{OOI} = q_3$ and its uncertainty field $\{q_1, q_2, q_3, q_4, q_5\}$.

$\tilde{q}_{OOI}$	q_{OOI}	$\boldsymbol{\epsilon}^{\ell}_{t,q_{OOI}}$
q_3	q_1	$[$ -1 0 1 0 0 $\cdots$ 0$]^{\mathrm{T}}$
	q_2	$[$ 0 -1 1 0 0 $\cdots$ 0$]^{\mathrm{T}}$
	q_3	$[$ 0 0 0 0 0 $\cdots$ 0$]^{\mathrm{T}}$
	q_4	$[$ 0 0 1 -1 0 $\cdots$ 0$]^{\mathrm{T}}$
	q_5	$[$ 0 0 1 0 -1 $\cdots$ 0$]^{\mathrm{T}}$



(a)



(b)

Figure 4.2: (a) Showing the location uncertainty field for an OOI detected in $\boldsymbol{y}_{thr,t}$ at the position marked " × " in red. It includes all grid cells in the region bounded radially by $[-r_u, r_u]$ around the observed OOI location within an angular spread of $[-\theta_u, \theta_u]$ relative to the line of sight to the agent. (b) Showing part of the additive noise vector due to target location uncertainty in an agent's observation (Eq. (4.3)) when an OOI is actually present at q_2 but observed at q_3.

Using the computed location uncertainty field for an OOI at $\tilde{q}_{OOI}$ in its FOV, the agent computes a probability function over the grid cells where the OOI could truly be located. Referring to the observation noise due to location uncertainty $\boldsymbol{\epsilon}_t^{\ell}$ in Eq. (4.2), we note that in a discretized grid environment, it essentially results in swapping the OOI's observed location within a region around the true OOI location (Fig. 4.1b, Fig. 4.2b). This needs to be captured in the measurement model for the agent's KF. For example, following Fig. 4.2b, if $\boldsymbol{y}_{thr,t}$ indicates that an OOI is located at $\tilde{q}_{OOI} = q_3$ and its location uncertainty field is $\{q_1, q_2, q_3, q_4, q_5\}$, Table 4.1 shows the possible location swaps the agent reasons about including the case where the OOI's true location is also q_3. In Table 4.1, $\tilde{q}_{OOI}$ is the observed location, q_{OOI} is the possible true location within the location uncertainty field and $\boldsymbol{\epsilon}^{l}_{t,q_{OOI}}$ is the noise component due to location uncertainty that executes the position swap between the true and observed OOI locations. The index with -1 indicates q_{OOI} and that with 1 indicates $\tilde{q}_{OOI}$. After enumerating all feasible OOI position swaps in the location uncertainty field, the agent updates the measurement noise covariance as $\boldsymbol{\Sigma}_{\boldsymbol{z}_t}^{q_3,q_3}$ += $0^2 \times \frac{1}{5} + (1)^2 \times (1 - \frac{1}{5})$ and $\boldsymbol{\Sigma}_{\boldsymbol{z}_t}^{q_{1/2/4/5},q_3} = \boldsymbol{\Sigma}_{\boldsymbol{z}_t}^{q_3,q_{1/2/4/5}}$ += $1 \times (-1) \times \frac{1}{5} + 1 \times 0 \times \frac{3}{5} + 0 \times 0 \times \frac{1}{5}$. This is repeated for all possible OOIs in the agent's FOV. The constructed measurement noise covariance $\boldsymbol{\Sigma}_{\boldsymbol{z}_t}$ is then used to update the agent's KF with $(\boldsymbol{x}_t, \boldsymbol{y}_t)$.

Algorithm 4 outlines our UnIK algorithm for target recovery from noisy observations due to both detection and location uncertainty.

4.5 Experiments with UnIK

Our goal is to analyze the improvements using the inference method proposed in UnIK over baselines that do not jointly account for both target location and detection uncertainty in active search. We compare UnIK against two other inference methods: (1) NATS (Ghods et al., 2021a), which only accounts for detection uncertainty and (2) LU, which is our designed location uncertainty only baseline.

Algorithm 4 UnIK (inference with random action selection)

Assume: Sensing model (4.3), sparse true state $\boldsymbol{\beta}$
Set: $\mathbb{D}_0^j = \phi$, $p_0^j(\boldsymbol{\beta}) = \mathcal{N}(\hat{\boldsymbol{\beta}}_0^j, \hat{\mathbf{P}}_0^j)$, $\hat{\boldsymbol{\beta}}_0^j = \frac{1}{n}\mathbf{1}_{n\times 1}$, $\hat{\mathbf{P}}_0^j = \sigma_0^2\mathbf{I}$
for t in $\{1, \ldots, T\}$ **do**
    Select $\boldsymbol{x}_t^j$ uniform randomly; observe $\boldsymbol{y}_t^j$.
    Update $\mathbb{D}_t^j = \mathbb{D}_{t-1}^j \cup \{(\boldsymbol{x}_t^j, \boldsymbol{y}_t^j)\}$
    Obtain $\boldsymbol{y}_{thr,t}^j = \mathbf{1}(\boldsymbol{y}_t^j \geq c_{thr})$
    Initialize $\boldsymbol{\Sigma}_{\boldsymbol{z}_t}$ to account for the target detection uncertainty in the current FOV
    for each possible OOI in $\boldsymbol{y}_{thr,t}^j$ **do** ▷ Construct $\boldsymbol{\Sigma}_{\boldsymbol{z}_t}$ using $\boldsymbol{y}_{thr,t}^j$
        Obtain its location uncertainty field
        Update $\boldsymbol{\Sigma}_{\boldsymbol{z}_t}$ to account for its location uncertainty
    Estimate $\hat{\boldsymbol{\beta}}_t^j, \hat{\mathbf{P}}_t^j$ using KF

4.5.1 Baselines

NATS (Ghods et al., 2021a) is a Thompson sampling based active search algorithm that accounts for target detection uncertainty but does not consider location uncertainty. NATS uses sparse Bayesian learning (SBL) (Tipping, 2001) to estimate $\boldsymbol{\beta}$ using the set of available measurements. The detection uncertainty is modeled as an increasing function of the distance between the agent and the detected OOI. In order to ensure fair comparison, we use the inference method of NATS with the same distance dependent detection uncertainty model assumed for UnIK.

LU is our designed baseline inference algorithm that only accounts for the target's *l*ocation *u*ncertainty but not its detection uncertainty. For every observation $\boldsymbol{y}_t$ received by an agent, it is first thresholded to obtain $\boldsymbol{y}_{thr,t} = (\boldsymbol{y}_t \geq c_{LU})$ where c_{LU} is a parameter (constant). Corresponding to each possible OOI location q in $\boldsymbol{y}_{thr,t}$, indicated by its non-zero indices, we either initialize a Gaussian distribution $\mathcal{N}(\boldsymbol{q}, \boldsymbol{\Sigma}_{\boldsymbol{q}})$ centered at grid cell q or check to see if there already exists a previously initialized Gaussian distribution $\mathcal{N}(\boldsymbol{q}', \boldsymbol{\Sigma}_{\boldsymbol{q}'})$ subsuming q with probability at least 95%. In the latter case, we update the already existing Gaussian distribution using

$$\boldsymbol{q}'' = \boldsymbol{q}' + \boldsymbol{\Sigma}_{\boldsymbol{q}'}(\boldsymbol{\Sigma}_{\boldsymbol{q}'} + \boldsymbol{\Sigma}_{\boldsymbol{z}_t}^{\boldsymbol{q}})^{-1}(\boldsymbol{q} - \boldsymbol{q}') \tag{4.8}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{q}''} = \boldsymbol{\Sigma}_{\boldsymbol{q}'} - \boldsymbol{\Sigma}_{\boldsymbol{q}'}(\boldsymbol{\Sigma}_{\boldsymbol{q}'} + \boldsymbol{\Sigma}_{\boldsymbol{z}_t}^{\boldsymbol{q}})^{-1}\boldsymbol{\Sigma}_{\boldsymbol{q}'} \tag{4.9}$$

Note that $\boldsymbol{q}$, $\boldsymbol{q}'$ and $\boldsymbol{q}''$ are 2-dimensional vectors of x,y-coordinates. Correspondingly, $\boldsymbol{\Sigma}_{\boldsymbol{q}}$, $\boldsymbol{\Sigma}_{\boldsymbol{q}'}$ and $\boldsymbol{\Sigma}_{\boldsymbol{q}''}$ are 2×2 covariance matrices. We extract the target location uncertainty at the position q (denoted $\sigma_{q,t,LU}^2$) from the matrix $\boldsymbol{\Sigma}_{\boldsymbol{z}_t}$ constructed as described in Section 4.4.2. Then, $\boldsymbol{\Sigma}_{\boldsymbol{z}_t}^{\boldsymbol{q}} = \sigma_{q,t,LU}^2 \times \mathbf{I}_2$. This helps ensure that LU uses the same target location uncertainty model assumed for UnIK. Finally, LU estimates $\hat{\boldsymbol{\beta}}_t$ as the cumulative probability density estimate over the search space due to all the Gaussian distributions.

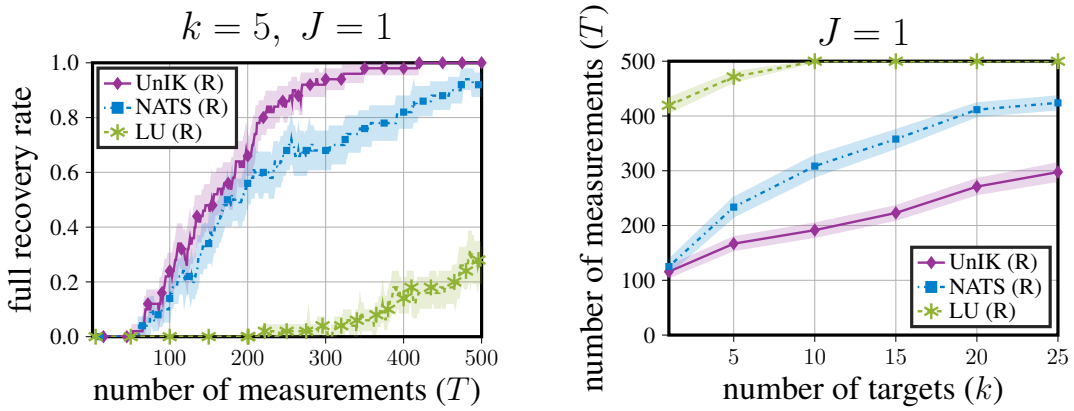
4.5.2 Simulation setup

In the following experiments, we consider a 2-dimensional (2D) discretized search space having J agents tasked with recovering the unknown search vector $\boldsymbol{\beta}$ which is randomly

generated using a uniform sparse prior with k non-zero entries each with value 1. We assume that agents are positioned at the centre of the grid cells they occupy and are free to move in any direction in the search space. When positioned in a cell, an agent can look in one of 4 possible directions: north, south, east or west to a maximum distance of 5 grid cells ahead with a 90° FOV. The agents' performance is measured using the full recovery rate. The plots show mean values with shaded regions indicating standard error over multiple trials, each trial differing only in the instantiation of the true position of the k OOIs in $\boldsymbol{\beta}$.

4.5.3 2D search space discretized into 16×16 grid cells

Fig. 4.3 illustrates the performance of UnIK compared to NATS and LU in a single agent setting ($J = 1$) in a 16×16 search space, over 50 random trials. At each step, an action is chosen uniformly at random and the agent receives the observation. The agent's actions and corresponding observations are the same across the three inference algorithms.



(a) (b)

Figure 4.3: *(R) indicates uniformly random action selection.* (a) Plot showing the full recovery rate versus number of measurements for $J = 1$ agent in a 16×16 grid having $k = 5$ targets. **UnIK estimates the number and position of *all* targets with fewer measurements.** (b) Plot comparing the number of measurements needed by $J = 1$ agent to fully recover all targets with increasing number of targets $k \in \{1, 5, 10, 15, 20, 25\}$ in a 16×16 search space. **UnIK scales better i.e. shows a smaller increase in the required number of measurements to achieve full recovery.**

Fig. 4.3a shows the full recovery rate of an agent when there are $k = 5$ targets distributed in a 16×16 search space. Unlike NATS or LU, UnIK leverages the combined target detection and location uncertainty modeling to detect all the OOIs with fewer measurements. Fig. 4.3b further compares the scaling efficiency of the algorithms in terms of number of measurements needed to achieve full recovery by varying the number of targets $k \in \{1, 5, 10, 15, 20, 25\}$. With increasing k, NATS lacking the target location uncertainty modeling requires an increasingly larger number of measurements compared to UnIK. On the other hand, in the absence of target detection uncertainty modeling, LU's performance is sensitive to the detection threshold and declines faster, failing in all 50 trials to recover all OOIs within $T = 500$ measurements for $k > 5$. In Fig. 4.3, LU achieved the best performance at $c_{LU} = \frac{2}{3}$ for $k = 1$ and $c_{LU} = \frac{3}{4}$ for $k = 5$.

4.6 TS-UnIK : Thompson sampling with UnIK

Following the discussion in Chapter 2, recall that parallelized asynchronous Thompson sampling (TS) is an excellent candidate for a decentralized multi-agent algorithm (Kandasamy et al., 2018; Ghods et al., 2021a,b). By using a posterior sample in the reward function, TS enables multiple agents to independently choose distinct sensing actions that maximize their respective rewards while incurring little additional regret compared to a centralized planner. Therefore we adopt a TS approach for action selection and combine it with the proposed inference method UnIK for a multi-agent active search algorithm that can recover targets using noisy observations with both target detection and location uncertainty.

Next we describe our TS based reward formulation. At time t, an agent j having available measurements $\mathbb{D}^j_{t-1}$ infers its posterior mean estimate $\hat{\boldsymbol{\beta}}^j_{t-1}$ and posterior covariance matrix $\hat{\mathbf{P}}^j_{t-1}$ using UnIK. It then draws a sample $\tilde{\boldsymbol{\beta}}^j_t \sim \mathcal{N}(\hat{\boldsymbol{\beta}}^j_{t-1}, \hat{\mathbf{P}}^j_{t-1})$. Assuming $\tilde{\boldsymbol{\beta}}^j_t$ to be the true search vector, the agent will prefer choosing an action $\boldsymbol{x}^j_t$ that will allow its updated estimate $\hat{\boldsymbol{\beta}}^j_t$ at time t to be as close as possible to $\tilde{\boldsymbol{\beta}}^j_t$.

Objective. In particular, we want to maximize the reward function

$$\mathcal{R}(\tilde{\boldsymbol{\beta}}^j_t, \mathbb{D}^j_{t-1}, \boldsymbol{x}^j_t) = \frac{\mathbb{E}_{\boldsymbol{z}^j_t|\tilde{\boldsymbol{\beta}}^j_t, \boldsymbol{x}^j_t}[-||\tilde{\boldsymbol{\beta}}^j_t - \hat{\boldsymbol{\beta}}^j_t||_2^2]}{\mathbb{E}_{\boldsymbol{z}^j_t|\tilde{\boldsymbol{\beta}}^j_t, \boldsymbol{x}^j_t}[||\hat{\boldsymbol{\beta}}^j_t||_2^2]} \tag{4.10}$$

where $\boldsymbol{z}^j_t$ is the KF measurement variable. Note that $\hat{\boldsymbol{\beta}}^j_t$ is a one-step lookahead estimate assuming that $\boldsymbol{x}^j_t$ would result in an observation $\boldsymbol{z}^j_t$ following the KF measurement model if $\boldsymbol{\beta}^j_t = \tilde{\boldsymbol{\beta}}^j_t$. The numerator favours actions $\boldsymbol{x}^j_t$ such that in expectation, the estimate $\hat{\boldsymbol{\beta}}^j_t$ is close to the assumed true state $\tilde{\boldsymbol{\beta}}^j_t$. The denominator $\mathbb{E}_{\boldsymbol{z}^j_t|\tilde{\boldsymbol{\beta}}^j_t, \boldsymbol{x}^j_t}[||\hat{\boldsymbol{\beta}}^j_t||_2^2]$ is analogous to the estimated strength of the signal, and since the numerator is non-positive, a larger denominator will lead to an overall higher reward. Therefore, our formulated reward prefers sensing actions that would maximally reduce the uncertainty in the agent's current posterior distribution.

Using $\hat{\boldsymbol{\beta}}^j_t = \hat{\boldsymbol{\beta}}^j_{t-1} + \mathbf{K}^j_t(\boldsymbol{z}^j_t - \boldsymbol{x}^j_t\hat{\boldsymbol{\beta}}^j_{t-1})$ and $\boldsymbol{z}^j_t \sim \mathcal{N}(\boldsymbol{x}^j_t\tilde{\boldsymbol{\beta}}^j_t, \boldsymbol{\Sigma}_{\boldsymbol{z}^j_t})$, we can derive the following:

1. $\mathbb{E}_{\boldsymbol{z}^j_t|\tilde{\boldsymbol{\beta}}^j_t, \boldsymbol{x}^j_t}[-||\tilde{\boldsymbol{\beta}}^j_t - \hat{\boldsymbol{\beta}}^j_t||_2^2]$ (4.11)

$= \mathbb{E}_{\boldsymbol{z}^j_t|\tilde{\boldsymbol{\beta}}^j_t, \boldsymbol{x}^j_t}[-||\tilde{\boldsymbol{\beta}}^j_t - \hat{\boldsymbol{\beta}}^j_{t-1} - \mathbf{K}^j_t(\boldsymbol{z}^j_t - \boldsymbol{x}^j_t\hat{\boldsymbol{\beta}}^j_{t-1})||_2^2]$ (4.12)

$= \mathbb{E}_{\boldsymbol{z}^j_t|\tilde{\boldsymbol{\beta}}^j_t, \boldsymbol{x}^j_t}[-||\tilde{\boldsymbol{\beta}}^j_t - \hat{\boldsymbol{\beta}}^j_{t-1} + \mathbf{K}^j_t\boldsymbol{x}^j_t\hat{\boldsymbol{\beta}}^j_{t-1} - \mathbf{K}^j_t\boldsymbol{z}^j_t||_2^2]$ (4.13)

$= \mathbb{E}_{\boldsymbol{z}^j_t|\tilde{\boldsymbol{\beta}}^j_t, \boldsymbol{x}^j_t}[-||\tilde{\boldsymbol{\beta}}^j_t - \hat{\boldsymbol{\beta}}^j_{t-1} + \mathbf{K}^j_t\boldsymbol{x}^j_t\hat{\boldsymbol{\beta}}^j_{t-1}||_2^2 + 2(\tilde{\boldsymbol{\beta}}^j_t - \hat{\boldsymbol{\beta}}^j_{t-1} + \mathbf{K}^j_t\boldsymbol{x}^j_t\hat{\boldsymbol{\beta}}^j_{t-1})^{\mathrm{T}}\mathbf{K}^j_t\boldsymbol{z}^j_t - ||\mathbf{K}^j_t\boldsymbol{z}^j_t||^2]$ (4.14)

$= -||\tilde{\boldsymbol{\beta}}^j_t - \hat{\boldsymbol{\beta}}^j_{t-1} + \mathbf{K}^j_t\boldsymbol{x}^j_t\hat{\boldsymbol{\beta}}^j_{t-1}||_2^2 + 2(\tilde{\boldsymbol{\beta}}^j_t - \hat{\boldsymbol{\beta}}^j_{t-1} + \mathbf{K}^j_t\boldsymbol{x}^j_t\hat{\boldsymbol{\beta}}^j_{t-1})^{\mathrm{T}}\mathbf{K}^j_t\boldsymbol{x}^j_t\tilde{\boldsymbol{\beta}}^j_t$
$\quad - ||\mathbf{K}^j_t||_F^2(\mathrm{tr}\Big(\boldsymbol{\Sigma}_{\boldsymbol{z}^j_t}\Big) + ||\boldsymbol{x}^j_t\tilde{\boldsymbol{\beta}}^j_t||_2^2)$ (4.15)

The last equation uses $\boldsymbol{z}^j_t|\tilde{\boldsymbol{\beta}}^j_t, \boldsymbol{x}^j_t \sim \mathcal{N}(\boldsymbol{x}^j_t\tilde{\boldsymbol{\beta}}^j_t, \boldsymbol{\Sigma}_{\boldsymbol{z}^j_t})$, so we get $\mathbb{E}_{\boldsymbol{z}^j_t|\tilde{\boldsymbol{\beta}}^j_t, \boldsymbol{x}^j_t}[\boldsymbol{z}^j_t] = \boldsymbol{x}^j_t\tilde{\boldsymbol{\beta}}^j_t$ and $\mathbb{E}_{\boldsymbol{z}^j_t|\tilde{\boldsymbol{\beta}}^j_t, \boldsymbol{x}^j_t}[||\boldsymbol{z}^j_t||^2] = \mathrm{tr}\Big(\boldsymbol{\Sigma}_{\boldsymbol{z}^j_t}\Big) + ||\boldsymbol{x}^j_t\tilde{\boldsymbol{\beta}}^j_t||_2^2$.

2. $\mathbb{E}_{\boldsymbol{z}_t^j|\tilde{\boldsymbol{\beta}}_t^j,\boldsymbol{x}_t^j}[||\hat{\boldsymbol{\beta}}_t^j||_2^2]$ (4.16)

$= \mathbb{E}_{\boldsymbol{z}_t^j|\tilde{\boldsymbol{\beta}}_t^j,\boldsymbol{x}_t^j}[||\hat{\boldsymbol{\beta}}_{t-1}^j + \mathbf{K}_t^j(\boldsymbol{z}_t^j - \boldsymbol{x}_t^j\hat{\boldsymbol{\beta}}_{t-1}^j)||_2^2]$ (4.17)

$= ||\hat{\boldsymbol{\beta}}_{t-1}^j - \mathbf{K}_t^j\boldsymbol{x}_t^j\hat{\boldsymbol{\beta}}_{t-1}^j||_2^2 + ||\mathbf{K}_t^j||_F^2(\mathrm{tr}\Big(\boldsymbol{\Sigma}_{\boldsymbol{z}_t^j}\Big) + ||\boldsymbol{x}_t^j\tilde{\boldsymbol{\beta}}_t^j||_2^2) + 2(\hat{\boldsymbol{\beta}}_{t-1}^j - \mathbf{K}_t^j\boldsymbol{x}_t^j\tilde{\boldsymbol{\beta}}_t^j)^{\mathrm{T}}\mathbf{K}_t^j\boldsymbol{x}_t^j\tilde{\boldsymbol{\beta}}_t^j$ (4.18)

Together, Eq. (4.15) and Eq. (4.18) can be substituted back in Eq. (4.10) to compute the objective $\mathcal{R}\Big(\tilde{\boldsymbol{\beta}}_t^j, \mathbb{D}_{t-1}^j, \boldsymbol{x}_t^j\Big)$. Finally, among all actions $\{\boldsymbol{x}_t^j\}$ at time t, agent j chooses $\boldsymbol{x}_t^{j^*}|\tilde{\boldsymbol{\beta}}_t^j = \mathrm{argmax}_{\boldsymbol{x}_t^j}\,\mathcal{R}(\tilde{\boldsymbol{\beta}}_t^j, \mathbb{D}_{t-1}^j, \boldsymbol{x}_t^j)$. Algorithm 5 outlines this action selection process.

Algorithm 5 TS-UnIK

1: **Assume:** Sensing model (4.3), true state $\boldsymbol{\beta}$, J agents
2: **Set:** $\mathbb{D}_0^j = \phi$, $p_0^j(\boldsymbol{\beta}) = \mathcal{N}(\hat{\boldsymbol{\beta}}_0^j, \hat{\mathbf{P}}_0^j)$, $\hat{\boldsymbol{\beta}}_0^j = \frac{1}{n}\mathbf{1}_{n\times 1}$, $\hat{\mathbf{P}}_0^j = \sigma_0^2\mathbf{I}\ \forall j \in \{1,\ldots,J\}$
3: **for** t in $\{1,\ldots,T\}$ **do** ▷ For any available agent j
4: Sample $\tilde{\boldsymbol{\beta}}_t^j \sim p(\boldsymbol{\beta}|\mathbb{D}_{t-1}^j) = \mathcal{N}(\hat{\boldsymbol{\beta}}_{t-1}^j, \hat{\mathbf{P}}_{t-1}^j)$
5: $\boldsymbol{x}_t^{j^*} = \mathrm{argmax}_{\boldsymbol{x}_t^j}\,\mathcal{R}(\tilde{\boldsymbol{\beta}}_t^j, \mathbb{D}_{t-1}^j, \boldsymbol{x}_t^j)$. Observe $\boldsymbol{y}_t^j$.
6: Update $\mathbb{D}_t^j = \mathbb{D}_{t-1}^j \cup \{(\boldsymbol{x}_t^{j^*}, \boldsymbol{y}_t^j)\}$. Share $(\boldsymbol{x}_t^{j^*}, \boldsymbol{y}_t^j)$ asynchronously with teammates.
7: Estimate $\hat{\boldsymbol{\beta}}_t^j, \hat{\mathbf{P}}_t^j$ using UnIK (Line 6-Line 11)

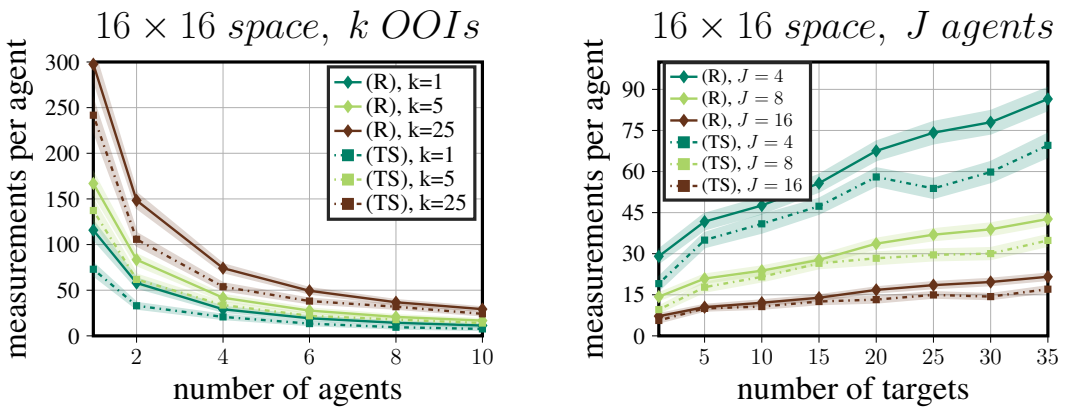
Data association. TS based decision making ensures that each agent can independently choose its next sensing location based on the uncertainty in its current posterior over the search space, and subsequently update its individual posterior estimates using its own measurements as well as those received from other agents. We therefore do not require perfect data association between measurements from different agents on the same OOI, nor a central controller for synchronization of observations across agents.

Computational complexity. Due to the Kalman filter based recursive nature of UnIK, for any agent j, the time complexity of each inference step is the same and bounded by $\mathrm{O}(n^{2.376})$. Additionally, TS-UnIK requires an agent to select the best (maximum reward) among all feasible actions at each time step. Therefore Line 5 in Algorithm 5 results in $\mathrm{O}(|\mathcal{A}|n^2)$ complexity at each time step where $|\mathcal{A}|$ is the size of the agent's feasible action space.

4.7 Experiments with TS-UnIK

We now demonstrate the efficiency of TS based action selection in combination with UnIK for decentralized and asynchronous multi-agent active search.

UnIK (R) vs. TS-UnIK. Following our observations in Section 4.5 where UnIK outperformed other baselines in fully recovering all OOIs, we will now use UnIK with uniformly random action selection (i.e. UnIK (R)) as the baseline against which we compare the performance of TS-UnIK.



(a) (b)

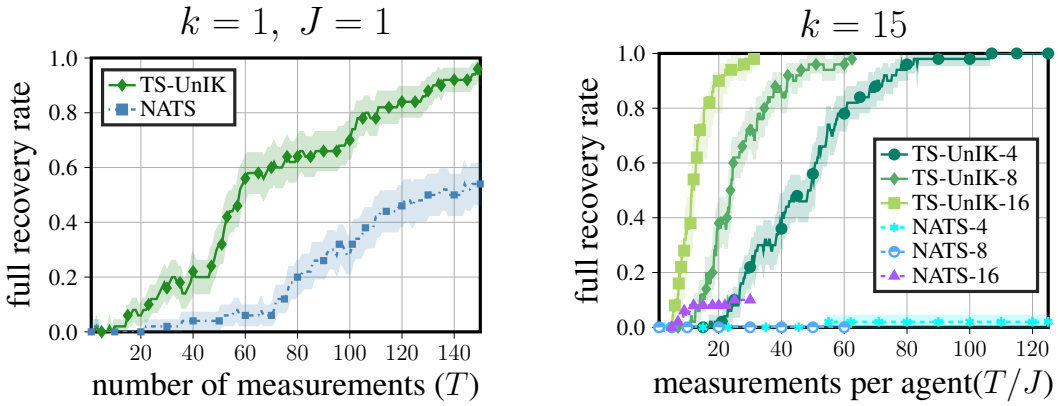
Figure 4.4: *(R) indicates uniformly random action selection.* (a) For the same team size, TS-UnIK enables efficient action selection requiring fewer measurements per agent to fully recover *all* targets. As the team size increases in a 16×16 search space, the performance of UnIK (R) catches up and upper bounds the number of measurements per agent required by TS-UnIK. (b) TS-UnIK consistently outperforms UnIK (R) for different number of targets k to be recovered in the search space by teams of different sizes J.

Fig. 4.4a plots the number of measurements needed per agent (T/J) to fully recover k OOIs as the number of agents J increases. In a perfect sensing setup, we would expect a single agent algorithm using the TS based action selection strategy to require J times as many measurements as that required per agent in a team with J agents (Ghods et al., 2021b). We observe this to hold for TS-UnIK, resulting in a J times improvement in performance when the number of agents multiplies J times. As the team size becomes larger, we observe that the full recovery performance plateaus in the absence of inter-agent coordination or centralized control.

Fig. 4.4b shows the robustness of multi-agent active search with TS-UnIK in terms of the number of measurements per agent required by a team to fully recover all OOIs as the number of OOIs k increases. For different team sizes (J) (indicated as UnIK/TS-UnIK-J), we observe that for both UnIK (R) and TS-UnIK to fully recover all OOIs, the average number of measurements needed per agent in the team increases with more number of OOIs in the search space. However in all the settings, TS-UnIK outperforms UnIK (R) with uniform random action selection and the performance gap widens further with more targets k for different J.

TS-UnIK vs. NATS. We now compare the performance of TS-UnIK against NATS with TS based action selection (Ghods et al., 2021a). Fig. 4.5a shows the full recovery rate within $T = 150$ measurements in a single agent setting ($J = 1$) when there is $k = 1$ target in a 16×16 search space. In this setting, TS-UnIK outperforms NATS and enables the single agent to achieve the desired recovery rate more efficiently with significantly fewer measurements.

Fig. 4.5b further validates the superiority of TS-UnIK over NATS in a multi-agent setting with multiple OOIs. For different team sizes $J \in \{4, 8, 16\}$ with the same total number of measurements $T = 500$, we compare the full recovery rate achieved by the team in terms



(a) (b)

Figure 4.5: (a) $J = 1$ agent is able to recover $k = 1$ targets in a 16×16 search space with fewer measurements using TS-UnIK than NATS. **Although both algorithms use Thompson sampling for action selection, the joint uncertainty modeling in TS-UnIK provides an advantage over only accounting for detection uncertainty in the noisy sensor observations.** (b) With $k = 15$ targets in a 16×16 search space, teams with different number of agents J following TS-UnIK achieve full recovery of all targets with fewer measurements per agent than those following NATS. **The ability to model both location *and* detection uncertainty becomes increasingly more advantageous when there are more targets in the search space.**

of the number of measurements required per agent. As before, we observe that TS-UnIK is able to efficiently recover all the OOIs and demonstrates a J times improvement in performance as the team size grows by a factor of J. On the other hand, with an identical team size and the same number of measurements per agent, NATS fails to achieve full recovery in the majority of trials. The deterioration in performance due to the absence of target location uncertainty modeling in NATS's inference method as observed in Section 4.5, is further exacerbated in the decentralized multi-agent setting with asynchronous communication. In particular, when different agents perform overlapping sensing actions, they may observe the same OOI at different locations due to observation noise arising from target location uncertainty. As a result, the combined detection and location uncertainty modeling in UnIK not only helps better estimate the possible OOI positions but this improved posterior belief also facilitates TS to choose effective sensing actions leading to efficient recovery of all OOIs with TS-UnIK.

TS-UnIK in Unreal Engine 4 (UE4) with AirSim plugin We test TS-UnIK in a pseudo-realistic environment created in UE4 with an AirSim plugin. It is a 250m×250m field with trees and animals, discretized into 16×16 grid cells. There are $k = 5$ humans randomly positioned within the field, who are the OOIs for our ground robot. We use YOLOv3 (Redmon and Farhadi, 2018) (off-the-shelf) as the agent's object detector, which provides a label and confidence score for OOIs in the agent's FOV (Fig. 4.6). Using the depth maps provided by AirSim, we implement our own depth sensor model that provides noisy location measurements to the agent following Eq. (4.2). Using measurements from these sensors, the agent decides its next sensing action following Eq. (4.10), with an additional

Figure 4.6: OOI detections from YOLOv3 in the robot's current FOV

travel distance penalization term. We refer to the video demonstrating the performance of TS-UnIK in this environment at this site .[1] Supporting the results in Section 4.7, the agent successfully recovers all OOIs with sensing actions that decide where and how the surroundings are observed to eliminate false or missed detections over time.

[1] https://sites.google.com/view/unik-icra23

5 | Uncertainty due to dynamic targets when targets outnumber agents

5.1 Introduction

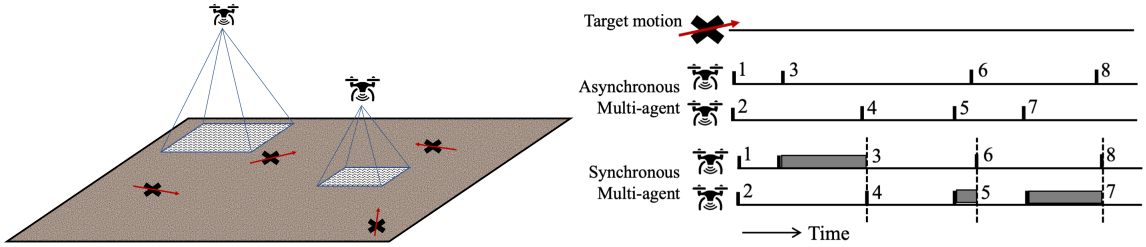
Searching for targets, detecting objects of interest (OOIs), localizing and following them are tasks integral to several robotics applications. In our discussion of multi-agent active search, we have so far focused on settings where the targets or objects of interest (OOIs) are stationary. In such applications like informative path planning (Popović et al., 2020) or simultaneous localization and mapping (SLAM) (Placed et al., 2023), when the OOIs are fixed, an agent (or robot) adaptively selects actions to detect and localize the targets. Here the environment (or, the target distribution) being stationary, an agent tends to explore unseen parts of its surroundings more than exploit already observed viewpoints. In contrast, when targets are dynamic, the environment is non-stationary. Therefore, agents tracking an unknown number of moving targets should trade-off between exploring the possibly unobserved parts of the environment and exploiting their own posterior estimates to localize the previously detected targets at the current timestep. Unfortunately, prior work in multi-target tracking (MTT) has often assumed that the environment is known and exploration is not of primary concern (Robin and Lacroix, 2016). Moreover, with multiple agents, existing MTT methods simplify the explore-exploit dilemma by separating search and tracking into sequential tasks where each agent is assigned to track a target as soon as it is found (Papaioannou et al., 2020). Another approach is to assign sub-teams for executing these tasks separately (Chen and Dames, 2022). Further, majority of these multi-agent multi-target tracking (MAMTT) algorithms require either a central controller to coordinate joint tracking actions, facilitate target hand-offs among agents or they depend on synchronized inter-agent communication for distributed inference and decision making. Unfortunately, such conditions may not be feasible in practice: the environment may be unstructured and unknown, OOIs may need to be simultaneously detected and localized (i.e. without a separation between search and tracking phases), there may not be enough agents to continuously monitor all the targets and unreliable communication channels may hinder inter-agent synchronization at each timestep.

This motivates the need to develop a more practical approach to MAMTT. In particular, we focus on the setting where agents are outnumbered by targets, so that the multi-agent team is unable to continuously cover all targets in their fields of view. The number of targets and their initial locations are unknown. Therefore, agents need to interact with

the environment to collect observations by adaptively making explore-exploit decisions. It is not feasible to continuously track all targets, but our goal is to produce an estimate of the number and positions of all targets with time. In keeping with our discussion so far, we assume that there is no central controller, and agents share their observations asynchronously with teammates, whenever possible. In other words, agents do not wait to receive observations, action selection policy or environment belief information from their teammates and can continue their online decision making when communication is unreliable or even unavailable.

5.2 Problem formulation

Consider a team of J UAVs tasked with search and tracking of an unknown number k of moving targets in a 2-dimensional (2D) region $\mathcal{G}$ of length n_l and width n_w (Fig. 5.1a). The agents can self-localize in a global coordinate system. They are equipped with noisy sensors that provide location 2D coordinates of possible targets in their current field of view (FOV). The targets can move in any direction in the search space at different speeds. We assume that agents typically move faster than targets. Each agent's FOV includes a contiguous rectangular region of the search space, and agents may choose to observe a wider (smaller) area at a greater (lower) vertical height but with more (less) observation noise. We therefore consider a hierarchical region sensing action space for each agent. Agents can communicate asynchronously with their teammates. Over time T, agents observe different parts of the search space to detect and track all targets in the environment.



(a) (b)

Figure 5.1: **Problem setup.** (a) Agents sense different regions of the search space at different vertical heights, receiving noisy 2D location coordinates of the possible targets in their field of view, along with false positive measurements. The targets shown as black crosses move in the search space with different velocities shown by the red arrows. (b) The line at the top indicates the target's continuous motion with time. In our asynchronous multi-agent setup, agents can collect observations without waiting for their teammates whereas in the synchronous setting, the solid boxes indicate the agents' idle wait times.

5.2.1 Target and Measurement Representations

The state of each target is denoted by $\mathbf{x} = \left[l_x, l_y, v_x, v_y\right]^{\mathrm{T}}$, where 2D coordinates $(l_x, l_y) \in [0, n_w] \times [0, n_l]$ and velocity $\mathbf{v} = \left[v_x, v_y\right]^{\mathrm{T}}$, $\|\mathbf{v}\|_2 \leq v_{\max}$. Since both the number of targets and their true locations are unknown, we follow the Random Finite Set (RFS) representation for the multi-target state space $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathcal{X}|}\}$, where $|\mathcal{X}|$ follows a Poisson distribution

and for a given cardinality, the set elements are sampled i.i.d from a uniform distribution (Mahler, 2014). Following prior work, we use the Probability Hypothesis Density (PHD) filter (Mahler, 2003) to maintain a belief over the RFS $\mathcal{X}$.

The PHD $\nu(x)$ is the first statistical moment of a distribution over RFSs. In this case, it is a density over the state space of targets so that for any region $E \subseteq \mathcal{G}$, the expected cardinality of the target RFS in that region is $\int_E \nu(x)dx$. The PHD filter tracks the evolving target density over the search space using models of target motion and measurements gathered by the agents. The measurements $\mathcal{Z}$ are also modeled as a (Poisson) RFS, as are clutter $\kappa(z)$ (false positives) and target births $b(x)$.

5.2.2 Sensing model

An agent with pose $\mathbf{q} = \begin{bmatrix} q_x, q_y \end{bmatrix}^{\mathrm{T}}$ executes a sensing action $\boldsymbol{x}_{\mathbf{q}}$, receiving a measurement set $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$. Any target $\mathbf{x}$ within the agent's FOV may generate a measurement $\mathbf{z}$, with a probability of detection $p_d(\mathbf{x}|\mathbf{q})$. Here we assume a constant $p_d(\cdot)$ when the target $\mathbf{x}$ is within the FOV at $\mathbf{q}$, and 0 otherwise. The agent follows a linear sensing model with additive i.i.d white noise: $\mathbf{z} = h(\mathbf{x}) + \boldsymbol{\omega}$, where $h(\mathbf{x}) = \begin{bmatrix} l_x, l_y \end{bmatrix}^{\mathrm{T}}$ and $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \sigma_h^2 \mathbf{I})$. Additionally, $\mathcal{Z}$ includes i.i.d false positives with clutter rate $\lambda_{\mathbf{q}}$.

The (noisy) target dynamics from state $\boldsymbol{\xi}$ to $\mathbf{x}$ is captured by the target motion model $f(\mathbf{x}|\boldsymbol{\xi})$. The survival probability $p_s(\mathbf{x})$ denotes the target's chances of persisting over successive time steps. The PHD filter formulates the following steps to propagate the posterior density over target states.[1]

$$\text{Prediction: } \bar{\nu}_t(\mathbf{x}) = b(\mathbf{x}) + \int_E f(\mathbf{x}|\boldsymbol{\xi}) p_s(\boldsymbol{\xi}) \nu_{t-1}(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{5.1}$$

$$\text{Update: } \nu_t(\mathbf{x}) = (1 - p_d(\mathbf{x}|\mathbf{q}))\bar{\nu}_t(\mathbf{x}) + \sum_{\mathbf{z} \in \mathcal{Z}_t} \frac{\psi_{\mathbf{z},\mathbf{q}}(\mathbf{x})\bar{\nu}_t(\mathbf{x})}{\eta_{\mathbf{z}}(\bar{\nu}_t)} \tag{5.2}$$

$$\eta_{\mathbf{z}}(\nu) = \kappa(\mathbf{z}|\mathbf{q}) + \int_E \psi_{\mathbf{z},\mathbf{q}}(\mathbf{x})\nu(\mathbf{x})d\mathbf{x} \tag{5.3}$$

$$\psi_{\mathbf{z},\mathbf{q}}(\mathbf{x}) = g(\mathbf{z}|\mathbf{x},\mathbf{q}) p_d(\mathbf{x}|\mathbf{q}) \tag{5.4}$$

Here, $\psi_{\mathbf{z},\mathbf{q}}(\mathbf{x})$ is the probability that the agent at $\mathbf{q}$ receives the measurement $\mathbf{z}$ from a target $\mathbf{x}$ and $g(\mathbf{z}|\mathbf{x},\mathbf{q})$ is the measurement likelihood model. The PHD filter can handle appearing and disappearing targets by defining an appropriate birth density $b(\mathbf{x})$ over the search space, but we assume the number of ground truth targets k is fixed.

Remark 1. As Mahler (2003) explains, using the first order moment to approximate the multi-target belief and deriving recursive PHD update equations to approximate the evolving posterior is justifiable when both sensor covariances and false alarm densities are small, so that (the distribution of) observations from true targets are centered around target states with negligible spread and there is lower noise due to false alarms. Besides, in our SMC-PHD implementation following Ristic et al. (2010), we also ensure we avoid particle impoverishment during update and propagation of the posterior density estimate.

[1] For a detailed understanding of the PHD filter, please refer to Mahler (2014).

Decentralized Asynchronous Multi-Agent Setup.

In our setup, each agent $j \in \{1, \ldots, J\}$ maintains its PHD estimate and decides its next sensing action in a decentralized manner. There is no central controller and inter-agent communication is asynchronous (Fig. 5.1b). Note that agents are not independent, since they share their own observations and incorporate the measurements received from their teammates in subsequent PHD filter update steps. Our setting is also different from the distributed computation setup in prior work (Dames, 2020) where each agent completes a part of the centralized update step and relies on inter-agent synchronized communication to maintain a global PHD estimate across all agents.

Since targets are in continuous motion, our agents must be able to deal with the uncertainty arising from observation noise as well as due to asynchronous communication of time-dependant observations in their posterior PHD updates. In order to enable time-ordered assimilation of received observations by all agents, we assume that any agent j communicates the tuple $(t, \boldsymbol{x}_t^{\mathbf{q}_j}, \mathcal{Z}_t^j)$ where $\boldsymbol{x}_t^{\mathbf{q}_j}$ and $\mathcal{Z}_t^j$ are respectively the agent's sensing action and measurement set at time t.

5.3 Related work

Target detection and tracking are both widely studied problems, typically considered as distinct tasks in various applications like search and rescue (Murphy, 2004a), security surveillance (Doitsidis et al., 2012), computer games (Oskam et al., 2009), etc. Robin and Lacroix (2016) present a detailed survey of the many different approaches and taxonomy used in robotics and related fields for such scenarios. Here, we discuss some of the most relevant prior work in our context.

The single target state is commonly modeled assuming linear dynamics with additive Gaussian noise, using the Kalman filter (Kalman, 1960) or using non-parametric particle filters (Doucet et al., 2001). In multi-target settings, alternative approaches like Multiple Hypothesis Tracker (MHT) (Blackman, 2004), Joint Probabilistic Data Association (JPDA) (Fortmann et al., 1983) and Probability Hypothesis Density (PHD) filter (Mahler, 2003) have been proposed, all of which differ in how they perform data association (Stone et al., 2013). The PHD filter is particularly suited when unique identities for each target are not required, for example, in search and rescue tasks, where agents should detect and localize *all* survivors. Following the setup in Section 5.2, we build on the Sequential Monte Carlo (SMC) formulation of the PHD filter presented in Ristic et al. (2010).

Prior work in MAMTT algorithms primarily considers centralized or distributed settings, the latter still necessitating synchronized communication among subgroups of agents at each time step. Coupled with a PHD filter, some of the common action selection methods previously proposed for tracking include mutual information and expected count based objectives (Dames et al., 2017), Renyi divergence maximization (Papaioannou et al., 2020) and Lloyd's algorithm for Voronoi-cell based control (Dames, 2020; Chen and Dames, 2020). Papaioannou et al. (2020) discuss the benefits of a simultaneous search-and-tracking algorithm but their proposed method requires that agents transition from searching to tracking upon target detection, foregoing further exploration. In similar spirit, Van Nguyen et al. (2022) propose solving an information gain based multi-objective optimization problem over a finite planning horizon to decide the best (greedy) joint action for a centralized search-and-tracking task. In contrast with these deterministic objectives, Xin and Dames

(2022) demonstrate the superior performance of stochastic optimization methods like Particle Swarm Optimization (PSO) and Simulated Annealing (SA) for better coverage and localization in such settings. Unfortunately, none of these prior approaches are applied in the decentralized and asynchronous multi-agent setup when agents are unable to support continuous target coverage.

Recently, some learning based approaches have been proposed for tracking. Jeong et al. (2021) consider the single-agent setting, assuming a known number of (one or two) targets which only start moving after being first observed by the agent. Zhou et al. (2022a); Tzes et al. (2023) both propose GNN-based algorithms, trained by imitation from a centralized expert, and deployed in the distributed inference and decision making with synchronized communication setup. While Zhou et al. (2022a) do not consider dynamic targets, Tzes et al. (2023) simplify the problem to deterministic optimal control over a fixed horizon, differing from our more complex setting.

5.3.1 Naïve approach and challenges

Following the theme of our prior multi-agent active search algorithms, our framework for multi-agent active search and tracking should combine posterior belief update with asynchronous decision making. Our first attempt at this involved using a Kalman filter, similar to Chapter 4, for modeling the uncertainty-aware posterior belief. However this was unsuccessful due to challenges in data association for maintaining a multi-modal joint posterior belief over the entire search space. Rosencrantz et al. (2003) similarly observe the importance of tractable belief updates and use a particle filter for belief tracking in a game of laser tag with robot teams. In contrast with our setting, the number of agents being tracked is known a priori, observation detection noise is not modeled and agents select their actions in a centralized greedy turn-based approach. In the absence of such simplifying assumptions, next we describe our proposed algorithm to address some of the resulting challenges to enable multi-agent active search and tracking with fewer agents than targets.

5.4 DecSTER: *Dec*entralized Multi-Agent Active *S*earch-and-*T*racking without continuous cov*er*age

We now describe our algorithm DecSTER for multi-agent active search and tracking without continuous coverage.

Notation. Agent j at time t has a history of available actions and observations $\mathbb{D}_t^j = \{(t', \mathbf{a}_{t'}^{\mathbf{q}'_j}, \mathcal{Z}_{t'}^{j'})\}_{t'<t, j' \in \{1,\ldots,J\}}$. Using $\mathbb{D}_t^j$, it computes the PHD ν_t^j (Eqs. (5.1) and (5.2)) over the target RFS. In our SMC-PHD implementation, $\nu_t^j = \{(w_{t,1}^j, \mathbf{x}_{t,1}^j), \ldots, (w_{t,\rho}^j, \mathbf{x}_{t,\rho}^j)\}$ where $\mathbf{x}_{t,1}^j, \ldots, \mathbf{x}_{t,\rho}^j$ are the ρ particles with weights $w_{t,1}^j, \ldots, w_{t,\rho}^j$.

The SMC-PHD filter propagation steps follow from Ristic et al. (2010). In our decentralized setup, each agent maintains its own posterior PHD ν_t^j. Next, we will describe the decision making step executed by agent j at time t.

Thompson sampling for decision making. Prior work in multi-agent active search with static targets has demonstrated the effectiveness of Thompson sampling (TS) as a decentralized decision making algorithm (Chapter 2), both in theory (Ghods et al., 2021b) and in practice (Bakshi et al., 2023). TS ensures stochasticity in decision making by sampling different plausible realizations of the ground truth from the posterior belief and selecting

the best action to maximize the desired reward for a particular sample. The uncertainty in the agent's belief over the state space is reflected in the posterior samples, which makes TS suitable for driving exploration and exploitation. Chen and Dames (2022) couples a TS-based active search strategy with the deterministic Lloyd's algorithm for tracking, but in their setup, agents are pre-assigned to only one of search or tracking tasks. Instead, we propose a TS strategy to enable agents to naturally trade-off exploratory sensing actions that might discover undetected targets, with exploitative sensing actions that help localize and track the previously detected dynamic targets in our simultaneous search-and-tracking setting.

To the best of our knowledge, prior work has not studied the problem of TS in a continuous (not discretized) search space with a PHD posterior. This is challenging because the PHD is not a distribution and does not include second order uncertainty information, whereas TS is typically applied in the Bayesian setting with the samples drawn from a posterior distribution for which both first and second order moment estimates are available (Russo et al., 2018). Prior work in Zhou et al. (2022b) has proposed particle Thompson sampling (PTS) and regenerative PTS (RPTS) algorithms for particle filters where particles are sampled proportional to their weights. Therefore, we adopt a similar principle in our first proposed TS strategy for the SMC-PHD posterior, called TS-PHD-I (Algorithm 6).

Algorithm 6 TS-PHD-I

1: **Input:** PHD $\nu = \{(w_1, \mathbf{x}_1), \ldots, (w_\rho, \mathbf{x}_\rho)\}$.
2: Sample $\tilde{\rho}$ particles $\{\mathbf{x}_i\}_{i=1}^{\tilde{\rho}}$ from ν, proportional to the weights $\{w_1, \ldots, w_\rho\}$
3: Cluster the $\tilde{\rho}$ particles using k-means with $\tilde{n} = \sum_{i=1}^{\tilde{\rho}} w_i$ centroids ($\tilde{\mathcal{X}} = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{\tilde{n}}\}$) which form the TS

We note that this method has drawbacks. It tends to sample more particles from the regions in the PHD where the agent already estimates targets might be present. The samples drawn are thus more likely to be biased against regions of the target state space where the agent might be less certain about its observations owing to false positives or missed detections. Furthermore, this method does a poor job of modeling the uncertainty about the number of true targets.

To address the drawbacks of Algorithm 6, we now describe a second proposed approach to Thompson sampling from our SMC-PHD posterior (Algorithm 7). Recall that the expected cardinality of the target RFS $\mathcal{X}$ over a region $E \subseteq \mathcal{G}$ is given by $\hat{n} = \int_E \nu(\mathbf{x}) d\mathbf{x}$. In case of the SMC-PHD representation, $\hat{n} = \sum_i w_i, \forall \mathbf{x}_i \in E$ (Ristic et al., 2010) i.e. the sum of particle weights of the SMC-PHD in the region E is the expected cardinality of $\mathcal{X}$ in that region. Further, Mahler (2003) shows that the PHD is the best Poisson approximation of the multitarget posterior in terms of KL divergence. We therefore draw a sample $\tilde{n}$ of the cardinality of the target RFS from a Poisson distribution with mean $\hat{n} = \sum_i w_i$ (i.e. $\tilde{n} \sim \text{Poisson}(\hat{n})$). Then we sample $\tilde{n}$ locations of the possible targets by drawing from a mixture of already estimated target locations in the PHD and some locations drawn uniformly at random over the search space. These $\tilde{n}$ particles $\tilde{\mathcal{X}} = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{\tilde{n}}\}$ form our TS.

Objective. The Optimal Sub-Pattern Assignment (OSPA) metric is typically used in the MTT literature for evaluating the tracking performance of an algorithm and is defined as the error between two sets. Given sets $\mathcal{X}$ and $\mathcal{Y}$, where $|\mathcal{X}| = m \leq |\mathcal{Y}| = n$ without loss of

Algorithm 7 TS-PHD-II

1: **Input:** PHD $\nu=\{(w_1,\mathbf{x}_1),\ldots,(w_\rho,\mathbf{x}_\rho)\}$. $\hat{n}_{\mathcal{G}}=\sum_{i=1}^{\rho} w_i$. Target location estimates $\hat{\mathcal{X}}=\{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_{\hat{n}_{\mathcal{G}}}\}$ from ν.
2: Sample $\tilde{n}\sim$ Poisson$(\hat{n}_{\mathcal{G}})$
3: Sample uniformly random target locations $\tilde{\mathcal{X}}_R$
4: Sample $\tilde{n}$ target locations $(\tilde{\mathcal{X}})$ from $\hat{\mathcal{X}}\cup\tilde{\mathcal{X}}_R$ as the TS

generality,

$$\text{OSPA}(\mathcal{X},\mathcal{Y})=\big(\frac{1}{n}\min_{\pi\in\Pi_n}\sum_{i=1}^{m} d_c(x_i,y_{\pi(i)})^p+c^p(n-m)\big)^{\frac{1}{p}}$$

where c is the cut-off distance, $d_c(x,y)=min(c,||x-y||)$ and Π_n is the set of all permutations of the set $\{1,\ldots,n\}$. The distance error component of the OSPA computes the minimum cost assignment between $\mathcal{X}$ and $\mathcal{Y}$, such that $x_i\in\mathcal{X}$ is matched to $y_{i'}\in\mathcal{Y}$ only when they are within a distance c of each other.

Given a true target set $\mathcal{X}=\{\mathbf{x}_1,\ldots,\mathbf{x}_{|\mathcal{X}|}\}$ and an estimated set $\mathcal{Y}=\{\mathbf{y}_1,\ldots,\mathbf{y}_{|\mathcal{Y}|}\}$ of possible target locations, our goal is to minimize OSPA$(\mathcal{X},\mathcal{Y})$. Since the ground truth $\mathcal{X}$ is unknown, each agent j instead draws a TS $\tilde{\mathcal{X}}_t^j$ from the predicted PHD $\bar{\nu}_{t+1}^j$. Assuming observations are generated by $\tilde{\mathcal{X}}_t^j$ for any action $\boldsymbol{x}$ and $\mathcal{Y}_t^j$ is the estimated target set following the PHD filter update, agent j then selects:

$$\mathbf{a}_t^j=\operatorname*{argmin}_{\mathbf{a}}\mathbb{E}_{\mathcal{Y}_t^j|\tilde{\mathcal{X}}_t^j,\mathbf{a}}[\text{OSPA}(\tilde{\mathcal{X}}_t^j,\mathcal{Y}_t^j)] \tag{5.5}$$

Algorithm 8 outlines our proposed algorithm, called DecSTER. In our decentralized and asynchronous multi-agent setting, each agent individually runs DecSTER with its own sampled $\tilde{\mathcal{X}}_t^j$. Hence the stochasticity in the sampling procedure enables agents to make decentralized explore-exploit decisions for simultaneous search-and-tracking in their action space.

Algorithm 8 DecSTER for agent j at time t

1: **Input:** PHD $\nu_t^j=\{(w_1^j,\mathbf{x}_1^j),\ldots,(w_\rho^j,\mathbf{x}_\rho^j)\}$
2: Compute predicted PHD $\bar{\nu}_{t+1}^j$ (Eq. (5.1)).
3: Draw TS $\tilde{\mathcal{X}}_t^j\sim\bar{\nu}_{t+1}^j$.
4: Assuming pseudo-measurements at $\tilde{\mathcal{X}}_t^j$, estimate expected target set $\mathcal{Y}_t^j$ and select action $\boldsymbol{x}_t^j$ (Eq. (5.5).
5: Observe $\mathcal{Z}_t^j$. Update PHD ν_{t+1}^j (Eq. (5.2)).
6: Estimate target set $\hat{\mathcal{X}}_{t+1}^j$ from ν_{t+1}^j (Ristic et al., 2010).
7: Asynchronously communicate $(t,\boldsymbol{x}_t^j,\mathcal{Z}_t^j)$ with team.

Remark 2. Prior work in search-and-tracking (Papaioannou et al., 2020; Chen and Dames, 2022) tends to separate the search and tracking phases of the task, and maintains either a visit count or dynamic occupancy grid to compute the action selection objective during the exploration phase. Such methods scale poorly with the size of the environment since agents need to maintain a discretization over the search space (Van Nguyen et al., 2022). In contrast, our SMC inference for multi-target belief is parallelizable over particles in

the posterior PHD, while our TS-based decision making is scalable with increasing teamsize J.

5.5 Experiments

We now describe our experimental setup. Consider a 2D search space with dimensions $n_l \times n_w = 16 \times 16$. There are k targets moving in this region, whose starting locations and velocities are chosen uniformly at random, such that $v_{\max} = 0.1$. A team of J agents are tasked with search-and-tracking of all the targets over $T = 150$ steps. The agents' action space $\mathcal{A}$ consists of hierarchical region sensing actions of width 1×1, 2×2, 4×4 and 8×8, $|\mathcal{A}| = 340$. Since actions with larger FOV receive noisier observations, we vary the false positive (clutter) rate as $\lambda \in \{0.005, 0.04, 1, 5\}$ for action widths 1, 2, 4 and 8 respectively. The survival probability in the PHD filter is set at $p_s = 1$ and the detection probability $p_d = 0.9$ for targets in the agent's FOV. In our SMC-PHD implementation following Ristic et al. (2010), we initialize 100 new (birth) particles per observation and re-sample 1000 particles per estimated target, following the low variance sampling method in Thrun (2002). We choose $\tilde{\rho} = 100$ (Algorithm 6).

The agents assume the target motion model $\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \boldsymbol{\epsilon}$, where $\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $\Delta T = 1$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$, $\mathbf{Q} = \begin{bmatrix} 0.03 & 0 & 0.05 & 0 \\ 0 & 0.03 & 0 & 0.05 \\ 0.05 & 0 & 0.1 & 0 \\ 0 & 0.05 & 0 & 0.1 \end{bmatrix}$. The sensing model is $\mathbf{z} = \mathbf{H}\mathbf{x} + \boldsymbol{\omega}$, where $\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ and $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, $\sigma = 0.1$. $\mathbf{I}$ is the identity matrix. For the OSPA metric, we set $c = 2$ and $p = 1$.

Remark 3. Our experimental setup is intended to illustrate the abilities of DecSTER for decentralized and asynchronous multi-agent multi-target search and tracking. The action space is chosen so that there is a non-trivial explore-exploit decision to be made by the agents. The maximum target speed is such that targets may cover a considerable distance in the search space over T steps.

In the following experiments, we measure performance in terms of the average OSPA for the entire team of agents. The plots show mean across 10 random trials with the shaded regions indicating standard error. Each trial differs in the initialization of the target locations and their velocities.

5.5.1 Comparing TS-PHD-I with TS-PHD-II

First, we compare the performance of DecSTER using the two proposed approaches for Thompson sampling from a PHD posterior. Fig. 5.2 compares OSPA with number of measurements per agent, for DecSTER-I and DecSTER-II using TS-PHD-I and TS-PHD-II respectively in Line 3 of Algorithm 8. We observe that decision making with TS-PHD-II consistently outperforms that with TS-PHD-I. Since TS-PHD-II samples both the cardinality and locations of the target RFS from the PHD, the samples for different agents

are sufficiently diverse to capture the uncertainty regarding the true multi-target ground truth. In contrast, the samples from TS-PHD-I are generally clustered around the agent's current estimate of target locations. We also empirically observed an improvement in the OSPA performance with TS-PHD-II when, for a particular sampled cardinality $\tilde{n}$, an agent considers multiple samples of $\tilde{n}$ target locations and averages the reward in Eq. (5.5) over them. Our results using DecSTER-II consider 10 such samples per action selection step for any agent j. We also allow DecSTER-I to similarly consider averaging over multiple samples, but this does not improve the sample diversity and does not lead to performance improvement.

Fig. 5.2 further demonstrates the scalability of TS in the decentralized multi-agent active search-and-tracking setting. When teamsize increases n times, agents achieve similar OSPA with n times fewer measurements per agent (Ghods et al., 2021b).

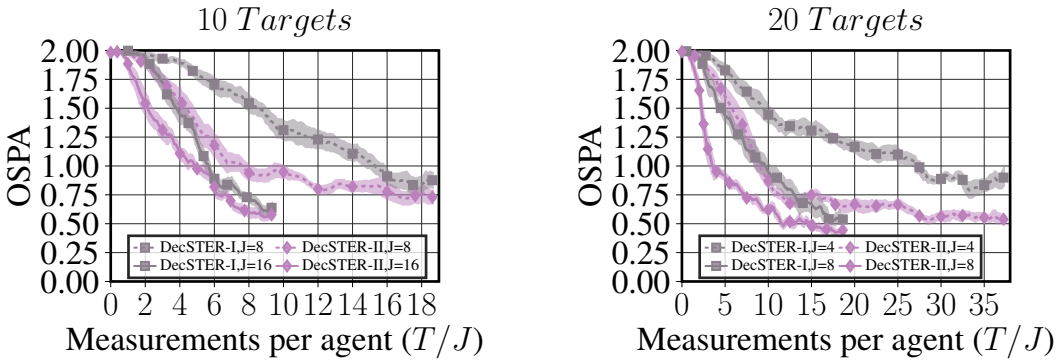


Figure 5.2: **Comparing the proposed TS methods**. DecSTER-II using TS-PHD-II consistently outperforms DecSTER-I using TS-PHD-I. Increasing team size J reduces the number of measurements per agent required to achieve similar OSPA.

5.5.2 Baseline comparisons

We compare DecSTER-I and DecSTER-II with the following baselines. Note that all of them use the same PHD filter inference method, but differ in the action selection policy. **1) RANDOM.** Each agent j selects its next sensing action uniformly at random. **2) RENYI.** At t, agent j computes the predicted PHD $\bar{\nu}^j_{t+1}$ and generates a (pseudo) measurement set $\bar{\mathcal{Z}}^j_t$ for any action $\boldsymbol{x} \in \mathcal{A}$ assuming the estimated target set $\hat{\mathcal{X}}^j_t$ from ν^j_t as ground truth. It then selects the action $\boldsymbol{x}^j_t$ that maximizes the Renyi divergence (with $\alpha = 0.5$) between $\bar{\nu}^j_{t+1}$ and its expected updated PHD ν'^j_{t+1} (Eq. (5.2)). With the SMC-PHD formulation, the Renyi divergence is (Ristic et al., 2011):

$$\sum_{i=1}^{\rho} \bar{w}_i + \frac{\alpha}{1-\alpha} \sum_{i=1}^{\rho} w'_i - \frac{1}{1-\alpha} \sum_{i=1}^{\rho} w'^{\alpha}_i \bar{w}^{1-\alpha}_i \tag{5.6}$$

where $\bar{w}_i$ and w'_i are the weights of the particle i in $\bar{\nu}^j_{t+1}$ and ν'^j_{t+1} respectively. **3) TS-RENYI.** We modify RENYI to use $\tilde{\mathcal{X}}^j_t \sim \bar{\nu}^j_{t+1}$ (with TS-PHD-II) for computing the (pseudo) measurement set $\bar{\mathcal{Z}}^j_t$ and the updated weights w'_i.

Fig. 5.3 shows that our proposed method outperforms all the baselines for different number of targets k and team sizes J. We observe that RENYI agents are information-greedy, therefore the lack of stochasticity in their decision making objective leads different agents to select the same action in the decentralized asynchronous multi-agent setting. Moreover, the computation in Eq. (5.6) depends only on the particles in $\bar{\nu}^j_{t+1}$ and does not account for previously undetected targets. This highlights the drawback of using Renyi divergence as an optimization objective for explore-exploit decisions in the search-and-tracking setting, in contrast with its success in the tracking setting where exploration is not a concern (Papaioannou et al., 2020). This motivated us to propose the TS-RENYI baseline in order to encourage exploration with samples drawn from TS-PHD-II. We observe that TS-RENYI still does not perform noticeably better than RENYI. This is because the weights of the particles in the SMC-PHD filter relate to the expected cardinality of the target set, therefore Eq. (5.6) does not account for any measure of the distance error between $\hat{\mathcal{X}}^j_t$ (or $\tilde{\mathcal{X}}^j_t$) and the estimate $\hat{\mathcal{X}}'^j_{t+1}$ from ν'^j_{t+1}. In contrast, the OSPA objective accounts for both localization error as well as cardinality error in the estimated target set. Thus we observe that our algorithm DecSTER-I is competitive with or outperforms random sensing and information-greedy baselines, and DecSTER-II consistently achieves the lowest OSPA among all with the same number of measurements per agent. Based on these results, we consider DecSTER-II as our best approach in this setting, labeled DecSTER in the following experiments.

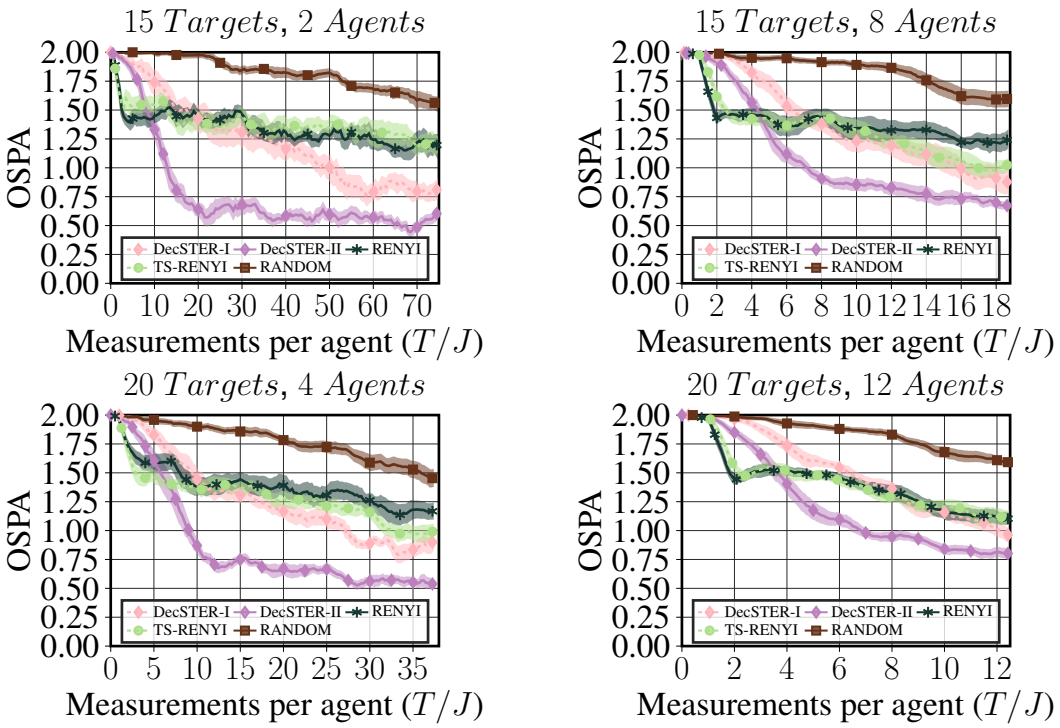


Figure 5.3: **Baseline comparisons**. For different numbers of targets and with fewer agents than targets, DecSTER with TS-PHD-II (denoted DecSTER-II) outperforms random sensing (RANDOM) and information greedy baselines (RENYI, TS-RENYI) by achieving a lower OSPA for the same number of measurements per agent.

DecSTER vs. DecSTER-C. Motivated by prior work (Dames et al., 2017) that showed the effectiveness of maximizing the expected number of target detections for action selection in multi-agent tracking, we introduce the DecSTER-C baseline where agents select actions minimizing only the cardinality error term of the OSPA. Unlike DecSTER, each agent with DecSTER-C draws multiple samples of cardinality $\tilde{n} \sim \text{Poisson}(\hat{n})$ to consider the uncertainty about the real number of targets in its objective. Fig. 5.4 shows that DecSTER still outperforms DecSTER-C, indicating that jointly considering detection and localization error in the decision making objective is more advantageous for TS-guided action selection in our search-and-tracking setting. This further supports our earlier observations regarding the drawbacks of using the particle weight based Renyi divergence optimization objective in this setting.

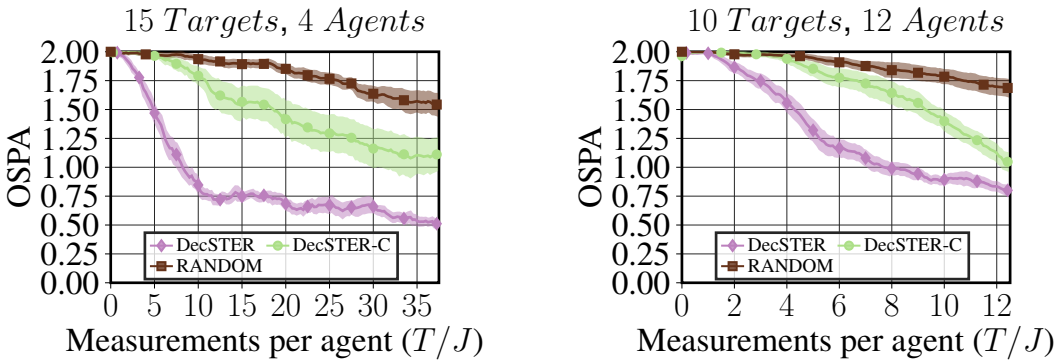


Figure 5.4: **DecSTER vs. DecSTER-C**. DecSTER-C optimizes only for the cardinality error in the OSPA objective. DecSTER outperforms DecSTER-C indicating the advantage of jointly optimizing detection and localization errors with TS-guided explore-exploit decisions.

5.5.3 Robustness to communication delays

Multi-agent systems benefit from leveraging observations shared by their teammates. Agents in our decentralized and asynchronous multi-agent setting benefit from any information shared by their teammates but they can continue searching for and tracking targets without waiting for such communication. Therefore, we now analyze the robustness of DecSTER under unreliable inter-agent communication. In simulation, we consider each agent chooses to communicate its own observation at time t, along with any prior observations it had not shared with its teammates, with a probability $p \in \{0.05, 0.25, 0.50, 0.75, 1\}$. The $p = 1$ setting corresponds to our description and analysis of DecSTER in Fig. 5.3. We observe a graceful decay in the OSPA performance with decreasing rates of inter-agent communication in Fig. 5.5, both when targets outnumber agents and vice versa. Compared to prior work in the centralized or distributed multi-agent tracking setting (Robin and Lacroix, 2016), DecSTER does not depend on synchronized communication within the team, thus agents can adapt and continue their search-and-tracking tasks even when communication is unreliable or unavailable.

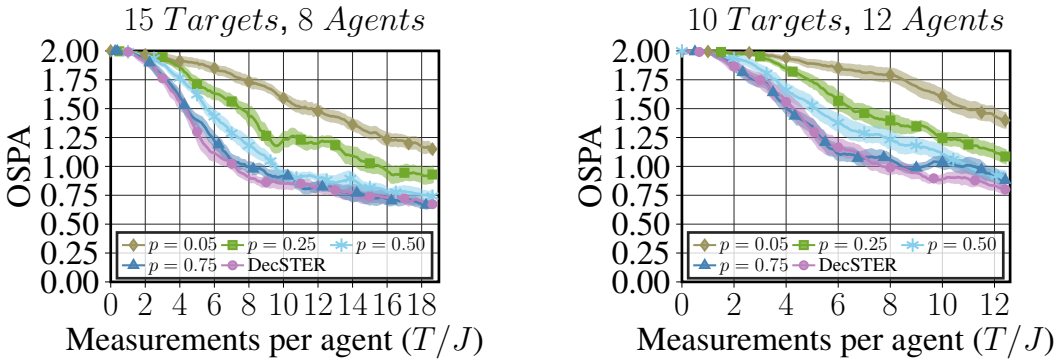


Figure 5.5: **Robustness to unreliable communication**. When agents communicate their actions and observations with decreasing probability p, DecSTER experiences a graceful deterioration in OSPA performance and agents require increasingly more measurements to estimate the number and locations of true targets in the search space.

Part III

Generative models for decision making in MAAS

6 | Diffusion for multi-objective multi-agent active search

6.1 Introduction

Agents interacting in an environment to accomplish tasks like target recovery, localization or mapping do so over multiple time steps, adapting their actions to their observations in the environment. In Chapter 3, we proposed CAST, a sequential decision making algorithm over a finite lookahead horizon which outperformed myopic greedy approaches to multi-agent active search. While effective, CAST encounters scalability challenges as the dimension of the action space or search space increases. This is also a common drawback in most tree search based approaches to long horizon planning, motivating our exploration for methods that could be leveraged for sequential decision making while amortizing the cost of rollouts in the search space. Therefore, in this chapter, we shift our focus to an alternative paradigm: generative models for decision-making in multi-agent active search. While many applications using generative models have focused on creative arts and media entertainment, in robotics, there is an immense scope for using and improving upon such approaches to effectively leverage information about the physical world from data (Yang et al., 2024). Recently, diffusion models (Song et al., 2021b) have emerged as a popular approach to generative modeling, particularly for image and video data. Diffusion has shown great success in video generation as sequences of frames over consecutive time steps, and prior work (Janner et al., 2022) has also proposed diffusion as a sequence modeling technique in offline reinforcement learning. However extrapolating the success of diffusion modeling to active search is non-trivial and in this chapter, we focus on some of these challenges and possible approaches to mitigating them for lookahead decision making in multi-agent active search.

6.2 Problem Formulation

We follow a similar setup as described in Section 4.2. Consider a team of J ground robots searching a space to locate some objects of interest (OOIs) or targets (Fig. 6.1). As before, we assume a gridded search environment described by a sparse matrix $B \in \{0,1\}^{n_\ell \times n_w}$ which the agents recover through active search. $\boldsymbol{\beta} \in \{0,1\}^n$ is the flattened vector representation of B and it is our search vector with k non-zero entries at the k target locations unknown to the agents. $n = n_\ell \times n_w$ is the total number of grid cells where n_ℓ and n_w are respectively the length and width dimensions of B.

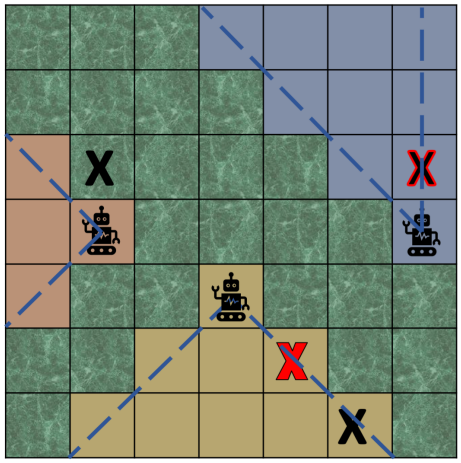


Figure 6.1: **Problem setup.** Agents sense different parts of the environment looking for OOIs. True OOIs are crossed in black. Targets detected by the agent in its field of view are crossed in red.

6.2.1 Sensing model

$\mathbf{x}_t \in \{0,1\}^{Q_t \times n}$ is a region sensing action at time t where Q_t is the number of grid cells in its field of view (FOV). Each row of this matrix is a one-hot vector indicating the grid cell being sensed. As Fig. 6.1 shows, the colored cells illustrate each robot's sensing action with a 90° FOV in their respective viewing directions. We assume a linear sensing model with i.i.d white noise at each grid cell:

$$\boldsymbol{y}_t = \mathbf{x}_t \boldsymbol{\beta} + \boldsymbol{\omega}_t \tag{6.1}$$

where $\boldsymbol{\omega}_t \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{1}_{Q\times 1})$, σ^2 is the (constant) variance of observation noise. Unlike in Chapter 4, we do not consider a distance dependent detection noise and we also do not consider correlations among observations in different grid cells due to location noise. This simplification is made to reduce the complexity of learning to model the observation noise in the agent's belief using the iterative noising and denoising procedure of the diffusion model.

6.2.2 Cost model

Following Chapter 3, we consider associated travel and sensing costs for the agents' actions in active search. An agent travelling from sensing location x_t (for sensing action $\mathbf{x}_t$) to x_{t+1} (for sensing action $\mathbf{x}_{t+1}$) incurs a travel time cost $c_d(x_t, x_{t+1})$ when the Euclidean distance $d(x_t, x_{t+1})$ is traveled at a constant speed v. Separately, executing the sensing action $\mathbf{x}_{t+1}$ at location x_{t+1} incurs a time cost $c_s(x_{t+1})$.

6.2.3 Communication

We follow the same communication model as before (Section 2.2), assuming that communication, although unreliable, will be available sometimes and agents should communicate asynchronously when possible to leverage information sharing in a multi-agent team.

To recover the search vector $\boldsymbol{\beta}$ by actively identifying all the targets, an agent j at time t selects a sensing action $\mathbf{x}_t^j$ based on its set of available measurements $\mathbf{D}_t^j$. In this work, our goal is to learn a non-myopic policy for decentralized and asynchronous (multi-agent) active search so that agents recover $\boldsymbol{\beta}$ with as few measurements T as possible while also optimizing for the total incurred cost.

6.3 Related Work

6.3.1 Diffusion Probabilistic Models

Diffusion models provide a framework to generate samples from an unknown distribution p^* by iteratively transforming samples from a simpler (i.e. easy to sample from) distribution, for example, a standard Gaussian distribution. Following Song et al. (2021a,b); Nakkiran et al. (2024), we provide a brief overview of the main ideas used in this chapter.

Given samples $x \sim p^*$[1], the diffusion process constructs a Markov chain to gradually add Gaussian noise to the data over a series of timesteps:

$$q(x^{\tilde{t}}|x^{\tilde{t}-1}) = \mathcal{N}(x^{\tilde{t}}; \sqrt{\frac{\alpha_{\tilde{t}}}{\alpha_{\tilde{t}-1}}}x^{\tilde{t}-1}, (1 - \frac{\alpha_{\tilde{t}}}{\alpha_{\tilde{t}-1}})\mathbf{I})$$

where $\alpha_{\tilde{t}}$ parameterizes the variance of the noise added over the $\tilde{t} = 1, \ldots, T_{\text{diff}}$ diffusion timesteps. $x^0 = x$. $q(x^0) = p^*$. This assumes a discrete time process, the superscript $\tilde{t}$ denoting the diffusion time step. The noise schedule (typically time dependant, fixed) is designed so that the data is converted to pure noise at $x^{T_{\text{diff}}}$. The reverse process is also a Markov chain which attempts to recover the original data by gradually removing this added noise. The generative model learns the distribution $p_\theta(x)$ by approximating the reverse denoising process:

$$p_\theta(x^0) = \int p_\theta(x^{0:T_{\text{diff}}})dx^{1:T_{\text{diff}}} \quad \text{where } p_\theta(x^{0:T_{\text{diff}}}) = p_\theta(x^{T_{\text{diff}}}) \prod p_\theta(x^{\tilde{t}-1}|x^{\tilde{t}}).$$

The parameters θ are trained to fit p^* by maximizing the variational lower bound (Song et al., 2021a):

$$\max_\theta \mathbb{E}_{q(x^0)}[\log p_\theta(x_0)] \leq \max_\theta \mathbb{E}_{q(x^{0:T_{\text{diff}}})}[\log p_\theta(x^{0:T_{\text{diff}}}) - \log q(x^{1:T_{\text{diff}}}|x^0)].$$

If all the conditionals $p_\theta(x^{\tilde{t}-1}|x^{\tilde{t}})$ are modeled as Gaussians with trainable mean functions and fixed variances, this objective can be simplified to:

$$\mathcal{L} = \sum_{\tilde{t}=1}^{T_{\text{diff}}} \mathbb{E}_{x\sim q(x^0), \epsilon_{\tilde{t}}\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \left[\|\epsilon_\theta^{\tilde{t}}(\sqrt{\alpha_{\tilde{t}}}x^0 + \sqrt{1-\alpha_{\tilde{t}}}\epsilon^{\tilde{t}}) - \epsilon^{\tilde{t}}\|_2^2\right]$$

where $x^{\tilde{t}} = \sqrt{\alpha_{\tilde{t}}}x^0 + \sqrt{1-\alpha_{\tilde{t}}}\epsilon^{\tilde{t}}$, $\epsilon^{\tilde{t}} \sim \mathcal{N}(0, \mathbf{I})$. We can then sample from such a trained model via Langevin dynamics (Ho et al., 2020).

[1]Notation x here is overloaded and does not refer to sensing location as described in Section 6.2.2.

Controllable generation with diffusion models

During the reverse denoising process, the generated samples can be biased or guided towards the support of the distribution with desirable properties. Dhariwal and Nichol (2021) and Ho and Salimans (2022) propose two main approaches for diffusion guidance. In classifier-free guidance (Dhariwal and Nichol, 2021), a conditional diffusion model is trained to generate samples given a particular value of the conditioning variable. In contrast, classifier guidance (Ho and Salimans, 2022) uses the gradient from a separate model $\nabla \log \nu(c|x)$ to bias the sample generation process after training. Diffusion guidance has been widely used for different applications like text generation (Li et al., 2022), video generation conditioned on previously generated frames (Ho et al., 2022a,b) and in reinforcement learning, to sample high reward trajectories or to condition on a certain goal (Janner et al., 2022).

6.3.2 Diffusion models in RL and robotics

Motivated by the performance of data-driven large scale training with sequence modeling architectures in language and vision domains, recent works (Janner et al., 2022; Ajay et al., 2023) have explored the use of diffusion models in offline RL tasks by framing RL as a sequence modeling problem, where the goal is to determine a sequence of actions which when executed in a sequence of observed states would lead to a desired outcome.

Janner et al. (2022) introduced diffusion models as planners in offline RL. Their model is trained to generate a sequence of states and actions that achieve a certain goal using reward-gradient guidance at sampling time. In practice, this approach promises to leverage the multimodal distribution representation capacity of diffusion models to learn from real-world interaction data and only uses the generated states (not the generated actions), alongwith a separate low level controller to determine the actions for the generated state transitions.

Alonso et al. (2024) train a diffusion based world model that can be used for training RL policies. In this work, the diffusion is only modeling the state for the RL agent conditioned on its previous actions and observations. In contrast, Zhou et al. (2024) model diffusion over both states and actions and demonstrate the superior performance of their reward-guided MPC-style lookahead diffusion algorithm over standard RL baselines. But their evaluation is limited to environments with deterministic state transitions where they have access to low dimensional state information. As we discuss later, stochasticity in state transition due to observation noise and unknown ground truth pose a significant challenge in the application of diffusion models for decision making in active search.

Although discrete diffusion models have been explored for applications in language modeling, their application in RL or robotics tasks is limited.

Diffusion models in multi-agent decision making

Some recent works have focused on leveraging diffusion models in multi-agent reinforcement learning, following the centralized training with decentralized execution (CTDE) framework in cooperative multi-agent settings. Zhu et al. (2023) introduce cross-attention in the diffusion network to model the coordination among behaviors of different agents. In this work, diffusion models the state dynamics whereas actions are derived from a separately trained inverse dynamics model. During training, this approach relies on joint action and observation data for centralized updates, whereas during decentralized execution, the per-agent (state) diffusion model also doubles as a behavior model for its teammates. The size

of this cross-attention layer has a quadratic dependency on the teamsize and would also require expensive multi-agent behavior data during training. In Shaoul et al. (2024), a multi-agent path finding approach is proposed by combining single-agent diffusion models with constraint based local planning. Therefore it relies on a central controller to coordinate the trajectories of the different agents. In contrast, there has been little focus on the asynchronous decentralized multi-agent setting which is of interest in this thesis.

6.3.3 Inductive biases in reinforcement learning for active search

A key factor underlying the performance gains recently enjoyed by foundation models in different applications like language modeling or video generation is that they often provide the appropriate inductive bias for the machine learning task. In reinforcement learning, prior work has explored equivariant representation learning in model-free RL (Mondal et al., 2022) and training Q-learning or actor-critic algorithms with equivariant network architectures (Nguyen et al., 2023; Wang et al., 2022) to improve the sample efficiency and generalization performance in different tasks. Recently Igoe et al. (2024) showed that graph neural networks improve the sample efficiency of RL algorithms for active search and Bayesian optimization by modeling appropriate equivariances and invariances in the belief and action space. But their observations were primarily limited to BO or active search settings with a single target and single agent with a disjoint (non-overlapping) action space. In this chapter, we expand on their observations and demonstrate the advantages of using a graph attention architecture with diffusion, particularly for improving training sample efficiency in 2D active search.

6.3.4 Naïve approach and challenges

Following the prior work discussed above, a feasible preliminary approach would be to extend the diffuser framework from Janner et al. (2022) for finite-horizon lookahead in active search. However there are a few assumptions made in most of the existing literature in diffusion for RL which do not hold in our setup and which would affect the performance of diffuser-like implementations for active target recovery. First, unlike standard offline RL tasks which assume deterministic state-dependent observations, our active search setting involves observation noise and an unknown ground truth search vector. This means that across different runs (or episodes), the same sequence of actions could lead to different observations and receive a high or low recovery reward depending on the search vector for that particular run. Moreover, the posterior belief update is non-deterministic given a sequence of actions, instead it also depends on the agent's observations which are influenced by observation noise and the unknown $\boldsymbol{\beta}$. Therefore training a generative model to learn a distribution over state-action sequences in active search is quite challenging and it involves dealing with non-negligible stochasticity and multi-modality in the training data. Finally, in active search, agents do not have fixed start and goal locations, instead agents must trade-off exploration and exploitation over the search space and adapt their subsequent behavior to prior observations. This leads to the optimism bias problem, addressed in Villaflor et al. (2022) using transformers for modeling state-action sequences in similar stochastic MDP formulations. In this chapter, we will focus on these challenges in the context of generative modeling with diffusion for lookahead decision making in active search.

6.4 Diffusion models for policy learning in active search

6.4.1 Belief representation

Following Section 6.2, agents recover the ground truth search vector $\boldsymbol{\beta}$ by actively identifying all the targets. Both the number of targets and their locations are unknown to the agents. Therefore we initialize each agent with a Gaussian prior over the search vector $\boldsymbol{\beta}$:

$$p_0(\boldsymbol{\beta}) = \mathcal{N}(\hat{\boldsymbol{\beta}}_0, \hat{\boldsymbol{\Sigma}}_0) \tag{6.2}$$

where the prior mean vector $\hat{\boldsymbol{\beta}}_0 = \frac{1}{n}\mathbf{1}_{n\times 1}$ and the prior covariance matrix $\hat{\boldsymbol{\Sigma}}_0 = \sigma^2\mathbf{I}_{n\times n}$. $n = n_\ell \times n_w$ is the size of the discretized search space. Since agents follow a linear sensing model (Eq. (6.1)), we can use a Kalman filter (Kalman, 1960) (Section 4.4) to update the posterior belief over the search space. Recall from Section 4.4.1, we can recursively update the posterior belief $p_t(\boldsymbol{\beta}) = \mathcal{N}(\hat{\boldsymbol{\beta}}_t, \hat{\boldsymbol{\Sigma}}_t)$ using the following alternate prediction and update steps.

$$\textit{Prediction: } \hat{\boldsymbol{\beta}}_t^- = \hat{\boldsymbol{\beta}}_{t-1},\ \hat{\boldsymbol{\Sigma}}_t^- = \hat{\boldsymbol{\Sigma}}_{t-1} \tag{6.3}$$

$$\textit{Kalman gain: } \mathbf{K}_t = \hat{\boldsymbol{\Sigma}}_t^-\mathbf{x}_t^{\mathrm{T}}(\mathbf{x}_t\hat{\boldsymbol{\Sigma}}_t^-\mathbf{x}_t^{\mathrm{T}} + \boldsymbol{\Sigma}_{\mathbf{y}_t})^{-1} \tag{6.4}$$

$$\textit{Update: } \hat{\boldsymbol{\beta}}_t = \hat{\boldsymbol{\beta}}_t^- + \mathbf{K}_t(\boldsymbol{y}_t - \mathbf{x}_t\hat{\boldsymbol{\beta}}_t^-) \tag{6.5}$$

$$\hat{\boldsymbol{\Sigma}}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{x}_t)\hat{\boldsymbol{\Sigma}}_t^-(\mathbf{I} - \mathbf{K}_t\mathbf{x}_t)^{\mathrm{T}} + \mathbf{K}_t\boldsymbol{\Sigma}_{\mathbf{y}_t}\mathbf{K}_t^{\mathrm{T}} \tag{6.6}$$

Here, $\mathbf{K}_t$ is the Kalman gain. $\boldsymbol{\Sigma}_{\mathbf{y}_t} = \sigma^2\mathbf{I}_Q$ where Q is the number of grid cells in the field of view of action $\mathbf{x}_t$ and σ follows from the sensing model Eq. (6.1). t denotes the decision making time step.[2]

In the remainder of this chapter, we will refer to the state of an agent as $\mathbf{s}_t = (\hat{\boldsymbol{\beta}}_t, \hat{\boldsymbol{\Sigma}}_t)$. Since our covariance matrix is diagonal, we will only account for the diagonal of $\hat{\boldsymbol{\Sigma}}_t$ in $\mathbf{s}_t$.

6.4.2 Lookahead plan generation with diffusion models

We define the lookahead plan for an agent in state $\mathbf{s}_t$ as the sequence of actions $\boldsymbol{\tau}_t = \mathbf{x}_{t:t+H}$ to maximize an objective $J(\mathbf{s}_t, \boldsymbol{\tau}_t)$. Therefore a diffusion model following Janner et al. (2022) can be trained to learn to generate samples $g = \{\mathbf{s}_{t'}, \mathbf{x}_{t'}\}_{t'=t\ldots t+H}$ from a joint distribution over state-action sequences. Next we will describe the data collection, network architecture and training method for this approach.

Data generation:

We first collect a dataset of trajectories comprising state-action-reward triples over episodes of length T. To simulate episodes for active search, we select actions that maximize the information gain in the agent's posterior belief following Eq. (6.10) and simulate M trials of single-agent single-target active search for T steps each. The ground truth target location differs across trials in the dataset. Each trial leads to a sequence of length T, each time step

[2]Note that throughout this chapter, and for diffusion models in RL in general, we have to carefully distinguish the diffusion timestep which we denote using superscript from the decision making timestep which we denote using subscript.

being labeled with the expected full recovery rate. In other words, for each action $\mathbf{x} \in \tau$ at state $\mathbf{s}$, the reward is given by

$$r = \mathbb{E}_{\tilde{\boldsymbol{\beta}} \sim \mathcal{N}(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\Sigma}}), \mathbf{y} | \boldsymbol{x}, \tilde{\boldsymbol{\beta}}} [\mathbf{1}\{\tilde{\boldsymbol{\beta}}_{\text{thr}} = \hat{\boldsymbol{\beta}}'_{\text{thr}} | (\boldsymbol{x}, \mathbf{y})\} - \mathbf{1}\{\tilde{\boldsymbol{\beta}}_{\text{thr}} \neq \hat{\boldsymbol{\beta}}'_{\text{thr}} | (\boldsymbol{x}, \mathbf{y})\}] \tag{6.7}$$

where $\tilde{\boldsymbol{\beta}}$ is a Thompson sample simulating a possible search vector consistent with the present state estimate. $\hat{\boldsymbol{\beta}}'$ is the one-step lookahead posterior mean estimate assuming the action $\boldsymbol{x}$ is executed with observation $\mathbf{y}$, assuming $\tilde{\boldsymbol{\beta}}$ as the ground truth. Note that $\tilde{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\beta}}'$ are continuous valued vectors in $(0,1)^n$, therefore we consider a threshold $c_{\text{thr}} \in (0,1)$ to get quantized vectors $\tilde{\boldsymbol{\beta}}_{\text{thr}}$ and $\hat{\boldsymbol{\beta}}'_{\text{thr}}$ in $\{0,1\}^n$. In other words, $[\tilde{\boldsymbol{\beta}}_{\text{thr}}]_i = \mathbf{1}\{[\tilde{\boldsymbol{\beta}}]_i >= c_{\text{thr}}\},\ \forall i \in \{1, \ldots, n\}$. The per-step reward r computes the expected value of all targets being recovered by action $\mathbf{x}$ in state $\mathbf{s}$.

We slice each trial into sequences of length H given by $(\mathbf{s}_{t:t+H}, \boldsymbol{x}_{t:t+H}, r_{t:t+H})$ at each $t \in \{0, \ldots, T-H\}$.

Trajectory representation:

Prior work in diffusion for RL attempts to fold as much of the planning stage as possible into the generative model so that sequential decision making becomes equivalent to drawing samples from a trained diffusion model (Janner et al., 2022; Chi et al., 2024). Considering the matrix representation for $\mathbf{s}_t \in [0,1]^{2 \times n_\ell \times n_w}$ and $\mathbf{x}_t \in \{0,1\}^{n_\ell \times n_w}$, we can represent a state-action sequence of length H as a sequence of 3-channel images, where each image in the sequence is a concatenation of $\mathbf{s}_t$ and $\mathbf{x}_t$ along the channel dimension. Following Janner et al. (2022), we can then train a diffusion model to generate samples $\{\mathbf{s}_{t'}, \mathbf{x}_{t'}\}_{t'=t}^{t+H}$ over lookahead horizon H and consider $\mathbf{x}_{t:t+H}$ as the generated action plan for the subsequent steps. We observed that this resulted in the generation of overly optimistic state-action sequences which did not select actions to balance exploration-exploitation in the search region. We refer to this as the *optimism bias* for diffusion based planning in active search and explain it further in Section 6.4.3.

To mitigate this challenge of optimism bias, we will instead generate a sequence of actions $\mathbf{x}_{t:t+H}$ conditioned on the agent's current state $\mathbf{s}_t$. In other words, we model lookahead decision making as denoising diffusion sampling over a sequence of single-channel images, each image being a $\{0,1\}^{n_\ell \times n_w}$ matrix representation of a region sensing action.

Training loss:

Recall the diffusion training and inference framework briefly described in Section 6.3.1. Since active search agents are tasked with recovering targets with as few measurements as possible, we should sample action sequences from the diffusion model that would have a higher expected full recovery reward (Eq. (6.7)). This conditional sampling framework is captured by gradient guidance during denoising diffusion sampling (Janner et al., 2022) and implies that we train both a trajectory generation model ρ_θ and a (discounted cumulative) return estimation model ν_ψ as described next.

Similar to Janner et al. (2022), we observe more consistent action sequences in the generated samples by using the least squares objective in Eq. (6.8) as the training loss for optimizing the trajectory diffusion model parameters θ.

$$\text{(Trajectory net)} \quad \mathcal{L}(\theta) = \mathbf{E}_{\substack{\tilde{t}\sim U(0,T_{\text{diffusion}}) \\ \mathbf{s},\boldsymbol{\tau}\sim\mathbb{D}_M}}[\|\boldsymbol{\tau} - \rho_\theta(\mathbf{s}, \boldsymbol{\tau}^{\tilde{t}}, \tilde{t})\|_2^2] \tag{6.8}$$

We use the regression loss in Eq. (6.9) for training the return estimation model parameters ψ. Note that we tried two approaches for this training framework: first, using diffusion with the same noise schedule as in ρ_θ and second, without diffusion i.e. with no forward noising. The latter was more stable in terms of training loss curves especially for larger 2D search spaces.

$$\text{(Return net)} \quad \mathcal{L}(\psi) = \mathbf{E}_{\substack{\tilde{t}\sim U(0,T_{\text{diffusion}}) \\ \mathbf{s},\boldsymbol{\tau},\mathbf{r}_{\boldsymbol{\tau}}\sim\mathbb{D}_M}}[\| \sum_{t'=0}^{H-1} \gamma^{t'} \mathbf{r}_{\boldsymbol{\tau}}[t+t'] - \nu_\psi(\mathbf{s}, \boldsymbol{\tau}^{\tilde{t}}, \tilde{t})\|_2^2] \tag{6.9}$$

Note that in Eq. (6.8) and Eq. (6.9), $\mathbf{s}$ indicates the agent's belief state at a decision timestep t, $\boldsymbol{\tau}$ is ta sequence of actions $\mathbf{x}_{t:t+H}$, $\mathbf{r}_{\boldsymbol{\tau}}$ is the reward sequence over $\boldsymbol{\tau}$ (per step reward follows Eq. (6.7)) and $\tilde{t}$ is the diffusion timestep.

Network architecture:

We explore two different neural network architectures for ρ_θ and ν_ψ. This is motivated by our observation that certain inductive biases ensure improved sample efficiency for training in 2D active search, compared to the 1D setup.

- *U-Net*: The U-Net architecture, first proposed for image segmentation (Ronneberger et al., 2015), has been widely used in image based diffusion. It consists of a contracting part and an almost symmetric expansive part, giving the architecture a U-shape. The contracting part is composed of blocks of convolution, activation, normalization and pooling layers which successively reduce the spatial resolution of the input image while increasing the number of features or channels. The expansive part mirrors the block arrangement, but instead upsamples the spatial dimension while combining the feature information from the contracting path through residual connections. In our setup, we can think of the temporal dimension H of the sequence $\tau \in \{0,1\}^{H\times n_\ell \times n_w}$ similar to the channel dimension in images, whereas the search space dimension $n_\ell \times n_w$ is the spatial dimension for images. For ρ_θ, we define the U-Net blocks so that the input and output dimensions are the same, corresponding to the sequence τ. Since we will train a conditional diffusion model, we additionally use FiLM conditioning (Perez et al., 2018) of the concatenated state $\mathbf{s}$ and diffusion timestep $\tilde{t}$ embedding. Our implementation of this architecture follows Janner et al. (2022).
- *GNN with attention*: Graph Neural Networks (GNN) are particularly useful for tasks where the input can be represented as a graph or which can benefit from geometric properties of a graph representation. Igoe et al. (2024) showed that GNNs are particularly well-suited for learning based approaches in Bayesian experiment design due to their domain permutation equivariance properties, resulting in multiple orders of magnitude improvement to training sample efficiency compared to naive parameterizations like MLPs or CNNs. Here we extend their observations to the more general active search setting and propose combining Graph Attention with diffusion for state-conditioned reward-gradient guided lookahead action generation.

In our graph representation, each node corresponds to a grid cell in the discretized search space. Each node i is represented by a feature vector

$$\mathbf{f}_i = [\boldsymbol{\tau}^{\tilde{t}}_{t:t+H,i}\ \hat{\boldsymbol{\beta}}_{t,i}\ \hat{\boldsymbol{\Sigma}}_{t,i}\ \tilde{t}_{\text{emb}}]_{1\times(H+3)}.$$

Here $\hat{\boldsymbol{\beta}}_{t,i}$ and $\hat{\boldsymbol{\Sigma}}_{t,i}$ indicate the i^{th} index element of the conditioning posterior mean and variance. $\boldsymbol{\tau}^0_{t:t+H,i}$ is the vector of 0s and 1s indicating whether the i^{th} grid cell is sensed (1) or not (0) for each of the H decision timesteps in $\boldsymbol{\tau}$. The superscript 0 indicates zero added diffusion noise and more generally, $\tilde{t}$ indicates the diffusion (de)noising timestep. $\tilde{t}_{\text{emb}}$ is a (learnable) embedding of the diffusion timestep. We further consider our graph to be fully connected with edge weights e_{ij} proportional to the Manhattan distance between grid cell i and j in the search space. Therefore, in this representation, the temporal dimension of the sequence modeling problem is captured by the node features and the spatial dimension by the node adjacency matrix.

In the trajectory generation network ρ_θ, the GNN output would model the sequence of actions over H steps. On the other hand, for ν_ψ, we further add a linear projection layer to output a scalar value for the estimated return. In both cases, GNNs rely on the pairwise message passing mechanism between graph nodes to learn the function mapping between the input and output during training. Prior work (Veličković et al., 2018) has proposed Graph Attention Networks (GAT) which introduce multi-head attention for message-passing among neighborhood nodes. We use graph attention layers in our network, where our edge weights are used to learn a soft attention mask. Overall, this network architecture introduces permutation equivariance in ρ_θ and permutation invariance in ν_ψ over the belief state-action space.

We separately train ρ_θ and ν_ψ with the training dataset $\mathbb{D}_M$ described earlier. In order to generate action sequences, we follow the gradient guided sampling framework similar to Janner et al. (2022). Algorithm 9 outlines the inference algorithm for active search with diffusion.

Cost awareness. In contrast with the classifier-free guidance framework, gradient guided diffusion provides flexibility in composing different reward or constraint functions to bias the generated samples at inference time. We leverage this to incorporate cost-awareness in the active search algorithm (Line 6 and Line 7). Here we additionally train a distance estimator d_φ by minimizing the mean square error with respect to the euclidean distance between two sensing locations in the search region.

Sampling batch size. We observe that the batchsize of the generated samples (Line 2) affects the quality of action sequences that the active search agent chooses from. Additionally, the coefficients α, λ in the gradient guidance term (Line 5) also influence the sampled action sequences and thereby the performance of the active search agent.

6.4.3 Optimism bias with diffusion in active search

Section 6.3.4 briefly touches upon some of the reasons behind our observation of optimism bias in this setting and why it has been overlooked in prior work which mostly evaluate planning in deterministic MDPs. Recall that we use gradient guidance from the trained

Algorithm 9 Cost-Aware Diffusion active search (CD-AS)

Input: ρ_θ, ν_ψ, d_φ, $\mathbf{s}_t$, current location x_{t-1}
for each of N_{diff} samples **do**
    Initialize $\boldsymbol{\tau}_t^{T_{\text{diff}}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    for $\tilde{t} = T_{\text{diff}} - 1, \ldots, 0$ **do**
        $\boldsymbol{\tau}_t^{\tilde{t}-1} \sim \mathcal{N}(\rho_\theta(\mathbf{s}_t, \boldsymbol{\tau}^{\tilde{t}}, \tilde{t}) + \alpha \boldsymbol{\Sigma}_{\text{diff}}^{\tilde{t}}(\nabla \nu_\psi(\mathbf{s}_t, \boldsymbol{\tau}_t^{\tilde{t}}, \tilde{t}) - \lambda \nabla d_\varphi(\mathbf{s}_t, \boldsymbol{\tau}_t^{\tilde{t}}, \tilde{t})), \boldsymbol{\Sigma}_{\text{diff}}^{\tilde{t}})$
    Compute $d_{\boldsymbol{\tau}_t^0}$: total distance traveled if $\boldsymbol{\tau}_t^0$ were executed starting from x_{t-1}
Select $\boldsymbol{\tau}_t^0$ that maximizes $\nu_\psi(\mathbf{s}_t, \boldsymbol{\tau}_t^0, 0) - \lambda \times d_{\boldsymbol{\tau}_t^0}$
Execute the first action $\mathbf{x}_t$ in $\boldsymbol{\tau}_t^0$. Observe $\mathbf{y}_t$.
Update $\mathbf{s}_{t+1}$

return network to generate samples over state-action sequences that would lead to higher expected full recovery reward. Since joint generation of states and actions does not disentangle the effects of active decision making from the dynamics of state transition, therefore gradient guided diffusion draws samples where the state-action sequence assumes the target will be recovered at the very first decision timestep thereby achieving a high full recovery reward. This happens because the generative process does not consider the ground truth observation which would determine the posterior belief update and subsequent adaptive action selection. Moreover the discounted cumulative return (used for gradient guidance) estimates the expectation over possible sequences of posterior belief updates ($\mathbf{s}_{t:t+H}$) given the action sequence ($\mathbf{x}_{t:t+H}$) for different search vectors $\boldsymbol{\beta}$, and therefore at inference, fails to account for the stochasticity in the active search belief MDP.

6.4.4 Diffusion for decentralized and asynchronous multi-agent active search

Algorithm 10 Multi-agent active search with diffusion

Input: ρ_θ, ν_ψ, $\mathbf{s}_t^j = (\hat{\boldsymbol{\beta}}_t^j, \hat{\boldsymbol{\Sigma}}_t^j)$, current location x_{t-1}^j
Sample $\boldsymbol{\tau}_{t,j}^0$ following Algorithm 9
Execute the first action $\mathbf{x}_{t,j}$ in $\boldsymbol{\tau}_{t,j}^0$. Observe $\mathbf{y}_{t,j}$.
Share $(\mathbf{x}_{t,j}, \mathbf{y}_{t,j})$ with team.
Update $\mathbf{s}_{t+1,j}$ using $(\mathbf{x}_{t,j}, \mathbf{y}_{t,j})$ and measurements received from other agents.

Algorithm 10 outlines our approach to decentralized and multi-agent cost-aware active search using the trained models. Note that the training data was collected in a single-agent setting, therefore we design each agent to draw its own samples independently over the lookahead horizon conditioned on its current state. This is only meant as a preliminary algorithm for diffusion in multi-agent active search and we leave this open as a potential direction for future research.

6.5 Experiments

6.5.1 Baselines

In this section, we will use the following algorithms as baselines to compare both cost-awareness and measurement efficiency.

- *Myopic information-greedy agent* (IG) (Ma et al., 2017) selects an action to maximize the expected information gain, which is equivalent to maximizing the reduction in entropy of the posterior belief distribution following the action.

$$\boldsymbol{x}_t = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, \mathbb{E}_{\mathbf{y}}[H(p(\hat{\boldsymbol{\beta}})) - H(p(\hat{\boldsymbol{\beta}}|\boldsymbol{x}, \mathbf{y}))] \tag{6.10}$$

- *Myopic Thompson Sampling agent* (TS) (Chapter 2) (Ghods et al., 2021b) selects an action to maximize the expected one-step lookahead reward defined as:

$$\boldsymbol{x}_t = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, \mathbb{E}_{\mathbf{y}|\boldsymbol{x},\tilde{\boldsymbol{\beta}}_t}[-\|\tilde{\boldsymbol{\beta}}_t - (\hat{\boldsymbol{\beta}}_{t+1}|\boldsymbol{x}, \mathbf{y})\|_2^2] \tag{6.11}$$

  where $\tilde{\boldsymbol{\beta}}_t$ is a Thompson sample from the posterior belief $p(\boldsymbol{\beta}) = \mathcal{N}(\hat{\boldsymbol{\beta}}_t, \hat{\boldsymbol{\Sigma}}_t)$ and $\hat{\boldsymbol{\beta}}_{t+1}|\boldsymbol{x}, \mathbf{y}$ is the one-step lookahead posterior mean if action $\boldsymbol{x}$ were executed with observation $\mathbf{y}$.

- *Implicit Q-Learning* (IQL) (Kostrikov et al., 2022) is an offline RL algorithm that decouples policy improvement from value function estimation and uses expectile regression to train a Q-function without querying unseen actions in the dataset, thereby avoiding bootstrapping errors in the temporal difference loss. IQL outperforms other approaches in offline RL particularly for tasks that require multi-step dyynamic programming updates to extract optimal behavior by stitching together sub-optimal training data. For our lookahead decision making task in active search, we have access to simulated trials with a myopic policy. As the search space becomes larger, our training dataset includes fewer number of optimal sequences of actions. Therefore, we use IQL as a baseline algorithm which implicitly learns a policy from a Q-function trained for multi-step dynamic programming updates. Note that unlike diffusion-based approaches, IQL is not primarily focused on multi-modality in the dataset or stochastic policies.
- *CAST* (Chapter 3) (Banerjee et al., 2023a), our previously proposed Thompson sampling and pareto-optimization guided Monte Carlo Tree Search based algorithm is adapted to the sensin model for ground robots. We consider CAST as an online, search based lookahead decision making algorithm for active search and compare CAST to our proposed diffusion modeling based generative approach to amortized lookahead decision making. Additionally, we will also compare the performance of cost-aware active search in CAST, with the inference-time cost-awareness of gradient-guided MPC style lookahead decision making with our proposed approach.
- *Diffusion policy* (Chi et al., 2024) proposes representing a robot's visuomotor policy as a conditional denoising diffusion process and demonstrates advantages of using diffusion in learning multimodal policies by behavior cloning. In contrast to our approach, this framework does not involve a reward model or gradient guidance to generate

constraint-aware plans. We will consider the diffusion policy baseline, using open-sourced code implementation (Cadene et al., 2024), to compare against our diffusion based approach to active search.

6.5.2 1D grid discretized into 16 cells

We first consider a one-dimensional (1D) search space discretized into 16 grid cells. For a single agent ($J = 1$) and a single target ($k = 1$), we simulate multiple trials where each trial differs in the (unkown) ground truth target location. We consider two settings for observation noise: $\sigma = \frac{1}{16}$ and $\sigma = 0.2$.

During training, our diffusion based approach uses $H = 8$ length sequences and the U-Net architecture for both ρ_θ and ν_ψ, with 64 denoising diffusion steps. At evaluation, we use a gradient guidance coefficient of $\alpha = 10$. For each decision step, we sample $N_{\text{diff}} = 10000$ action sequences of length $H = 8$ conditioned on the current belief state, execute the first sensing action from the sequence and replan with the updated posterior belief at the next timestep.

Fig. 6.2 shows that our diffusion based lookahead decision making outperforms myopic and shallow lookahead baselines. EIG is a myopic active search baseline which only considers a one-step lookahead reward. CAST is simulated with a search tree of depth 2 and 5000 simulation episodes. In both low and high observation noise settings, diffusion is able to recover the target with minimum (optimal) number of measurements. Note that for diffusion, it is important to sample a batch of action sequences at test time and select the one with the highest estimated return.

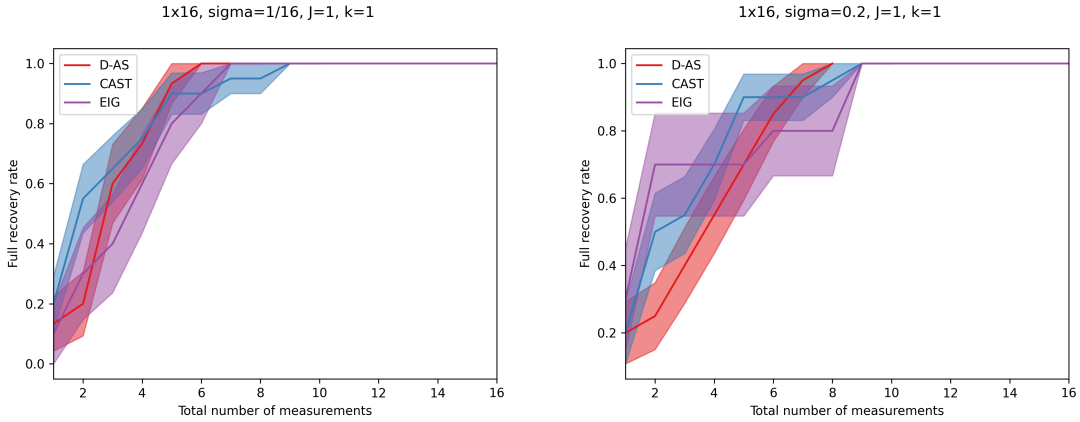


Figure 6.2: **Lookahead vs myopic decision making in 1D search space of size $n = 16$. $J = 1$ agent. $k = 1$ target. Low ($\sigma = \frac{1}{16}$) and high ($\sigma = 0.2$) observation noise.** Our diffusion based approach recovers the optimal sequence of actions and achieves full recovery faster than the myopic and shallow search tree based baselines. Plots show mean and standard error over 20 trials.

In Fig. 6.3, we compare the performance of online search (CAST) versus generative sampling (CD-AS) for lookahead cost-aware decision making in active search. CAST combines Monte Carlo Tree Search with pareto-optimization to trade-off travel cost and sensing cost. When sensing cost is low (0 secs), CAST focuses on optimizing the total incurred

travel cost and selects successive sensing locations close to one another. In contrast, when sensing cost is much higher than travel cost (sensing cost 50 secs), CAST optimizes the total number of measurements or sensing actions to minimize the incurred total cost. In Fig. 6.3, when the observation noise is higher ($\sigma = 0.2$ instead of $\sigma = \frac{1}{16}$), the total cost incurred for full recovery increases for both CAST and CD-AS. However when sensing cost is 0s, CAST outperforms CD-AS. This is because the pareto-front constructed by CAST considers the ground truth total cost for a sequence of actions, whereas the cost-awareness in CD-AS is determined by the parameters α, λ and the distance network d_φ. We observed that the network d_φ (when trained on the same dataset samples as ν_ψ) has poor generalization over the combinatorial space of length 8 lookahead action sequences, which explains the weaker cost-awareness of CD-AS. On the other hand, when sensing cost is 50s, CD-AS outperforms CAST across both levels of observation noise, further supporting our observation in Fig. 6.2 that diffusion based lookahead with gradient guidance recovers the optimal sequence of adaptive sensing actions.

6.5.3 2D grid discretized into 8×8 grid cells

Next, we focus on a two-dimensional (2D) search space discretized into 8×8 grid cells. We will consider scenarios with two different teamsizes ($J = 1, 3$) and number of targets ($k = 1, 4$). For each such scenario, we simulate multiple trials where every trial differs in the ground truth target location(s). As before, we consider two levels of observation noise: low noise with $\sigma = \frac{1}{16}$ and high noise with $\sigma = 0.2$.

We train the diffusion model on sequences of length $H = 10$. We use the previously described U-Net architecture for the trajectory network ρ_θ with $T_{\text{diff}} = 64$ diffusion denoising steps. For the return network ν_ψ, we use the GNN architecture and optimize Eq. (6.9) with $T_{\text{diff}} = 0$ i.e. without diffusion noising. This was observed to lead to better training sample efficiency and lower prediction error. At evaluation, we use a gradient guidance coefficient of $\alpha = 10$. For each decision step, we sample $N_{\text{diff}} = 100$ action sequences of length $H = 10$ conditioned on the current belief state, execute the first action and replan conditioned on the updated posterior belief at the next timestep (Algorithm 9).

Fig. 6.4 shows that our diffusion based lookahead decision making (D-AS) outperforms myopic greedy as well as shallow online lookahead baselines. We do not focus on cost-awareness in this comparison, therefore $\lambda = 0$ in Line 5. CAST is simulated with a budget of 25000 state-action rollouts to build a search tree of depth 2 at each online decision making step. In Section 6.5.5 we further discuss the trade-off between decision quality and time per decision making step for CAST. Recall from Section 6.4.2 that EIG is the baseline (behavior) policy used to collect the training data for our approach. It is a myopic greedy algorithm, therefore EIG does not always recover the optimal action sequence. In spite of sub-optimality in the training data, our gradient-guided diffusion model is able to generate action sequences that recover the target with fewer measurements than the EIG policy. We also consider IQL as a baseline due to its reported competitive performance in offline RL benchmark environments that benefit from a multi-step dynamic programming type algorithm to stitch together sub-optimal sequences into optimal plans or trajectories (Kostrikov et al., 2022). Unfortunately, IQL does not consider stochastic policies or multi-modality in action sequences for a task, which explains why IQL is unable to recover the target(s) in active search. Diffusion policy is another baseline for decision making with diffusion, but it follows the behavior cloning framework and as a result its performance

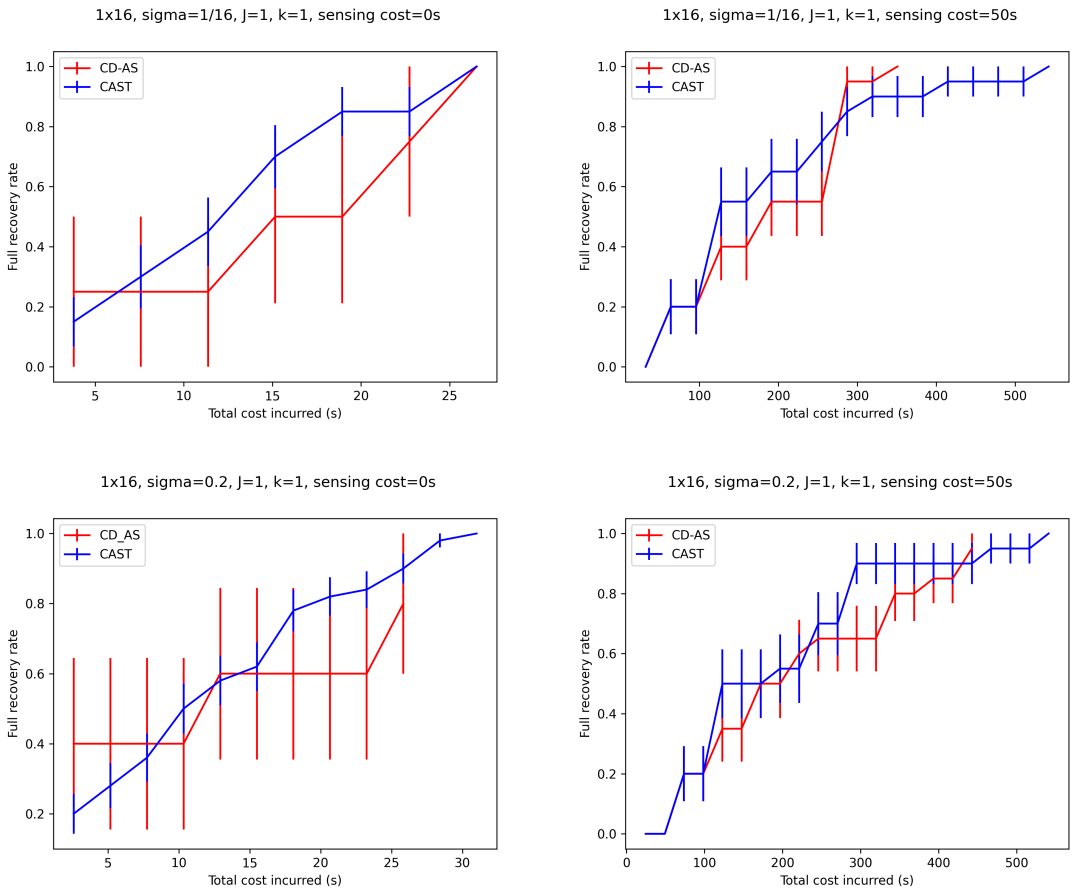


Figure 6.3: **Cost-aware decision making in 1D search space of size** $n = 16$**.** $J = 1$ **agent.** $k = 1$ **target. Low** ($\sigma = \frac{1}{16}$) **and high** ($\sigma = 0.2$) **observation noise.** Our diffusion based approach called CD-AS incurs smaller or competitive total cost compared to CAST when sensing cost is higher than traveling, but when sensing cost is low then CAST selects sensing actions which incur a smaller total cost. Plots show mean and standard error over 20 trials.

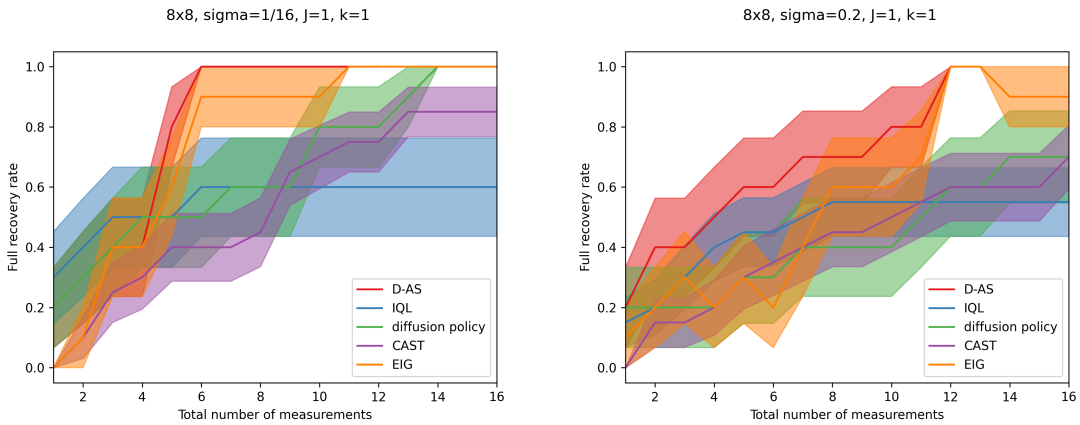


Figure 6.4: **Lookahead vs. myopic decision making in 2D search space of size 8×8. $J = 1$ agent. $k = 1$ target. Low ($\sigma = \frac{1}{16}$) and high ($\sigma = 0.2$) observation noise.** Our diffusion based approach (D-AS) samples the optimal sequence of actions and achieves full recovery with fewer measurements compared to myopic active search (EIG), offline RL (IQL), behavior cloning based diffusion (diffusion policy) and shallow online tree search (CAST) baselines. Plots show mean and standard error over 10 trials.

is limited by the quality of the training dataset. We observe that it is competitive with some of the active search baselines in the low noise setting but its performance deteriorates with higher observation noise. This is because the training dataset is collected with the EIG behavior policy, which is myopic in nature and does not support behavior cloning for lookahead decision making. In fact the lack of an optimal dataset is a practical and important constraint for training learning based approaches for long horizon planning or lookahead decision making in autonomous agents. The ability to utilize and learn from sub-optimal action sequences is an advantage of our gradient-guided diffusion framework as a generalizable and data-efficient algorithm for active search.

In Fig. 6.5, we compare the cost-aware performance of CAST versus our approach CD-AS. Recall that CAST computes the ground truth cost of sensing actions during tree search for online decision making, whereas cost-awareness in CD-AS is incorporated by gradient guidance from the trained distance network d_φ during denoising diffusion sampling. We observe that CAST outperforms CD-AS under both cost scenarios. Spatial distance estimation with neural networks is a difficult learning problem (Ramakrishnan et al., 2024), therefore we observe that gradient guidance with a scalarized combination of ν_ψ and d_φ does not sample cost-aware action sequences as effectively as constructing a pareto-front with ground truth cost estimates.

6.5.4 Decentralized and asynchronous multi-agent active search

We assume there are $J = 3$ agents and $k = 4$ targets distributed in an 8×8 search space. Following Algorithm 10, each agent samples its own set of action sequences conditioned on its belief state. In contrast, multi-agent CAST relies on Thompson sampling to build each agent's search tree. We observe that in Fig. 6.6, our diffusion based approach D-AS (with $\lambda = 0$) outperforms CAST in terms of the number of measurements required to completely

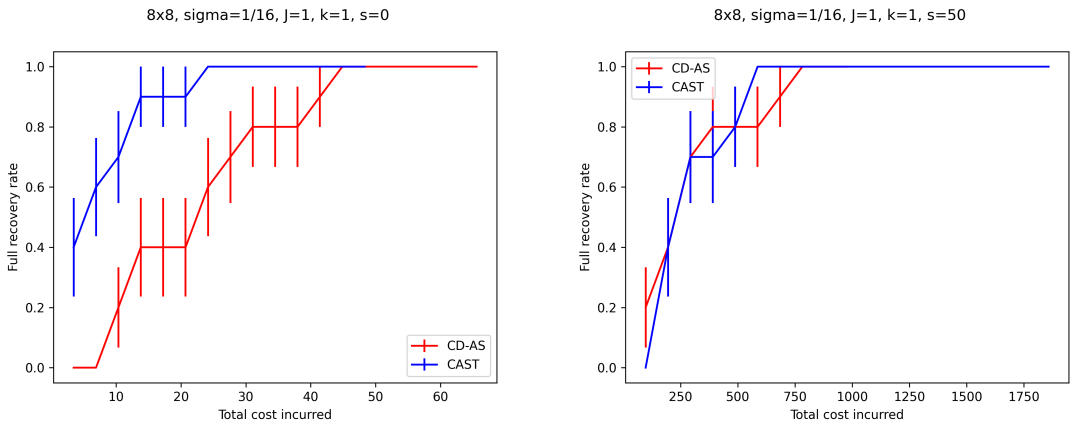


Figure 6.5: **Cost-aware decision making in a 2D search space of size** 8×8**.** $J = 1$ **agent.** $k = 1$ **target. Low** ($\sigma = \frac{1}{16}$) **and high** ($\sigma = 0.2$) **observation noise.** Our diffusion based approach requires fewer number of measurements (Fig. 6.4) but incurs a higher travel cost compared to CAST which combines cost-awareness with tree search based planning. Plots show mean and standard error over 10 trials.

recover all targets. Fig. 6.7 further compares their cost-awareness. When sensing is more expensive than traveling (sensing cost 50s), CD-AS is competitive with CAST in terms of the total cost incurred for full recovery. But CAST outperforms CD-AS when sensing cost is low (0s). In the latter case, CAST optimizes for the ground truth travel cost incurred in active search and selects action sequences that incur a low travel cost but do not optimally explore the search space. In contrast, CD-AS does not sample such low cost sequences since the training dataset was collected with the EIG policy which is not cost-aware and does not trade-off cost of sensing actions with information gain from exploring the search space.

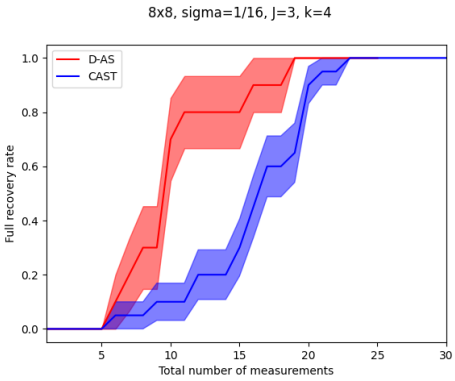


Figure 6.6: **Lookahead vs myopic decision making in 2D** 8×8 **search space.** $J = 3$ **agents.** $k = 4$ **targets.** Our diffusion based approach recovers the optimal sequence of actions for full recovery faster compared to shallow lookahead multi-agent active search with CAST. Plots show mean and standard error over 10 trials.

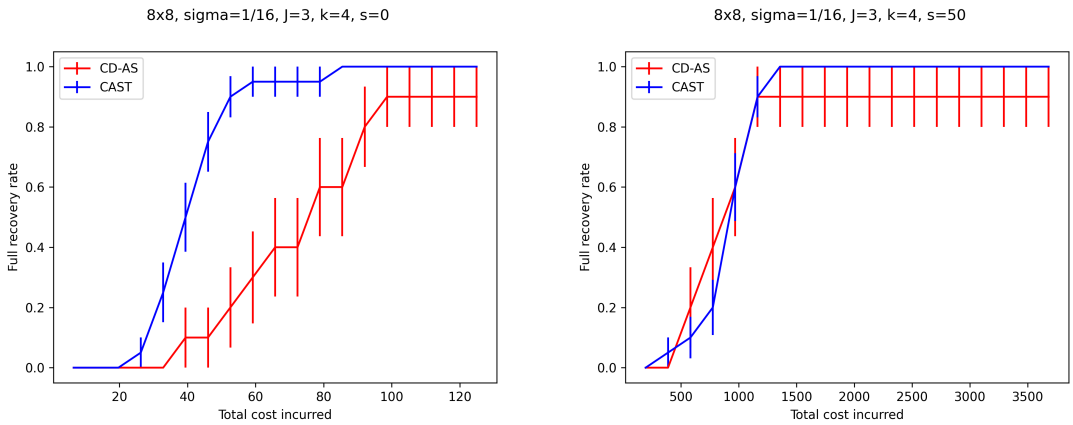


Figure 6.7: **Cost-aware decision making in 2D search space of size** 8×8 **.** $J = 3$ **agents.** $k = 4$ **targets.** Our diffusion based approach CD-AS incurs a higher total cost compared to CAST which combines cost-awareness with search tree based planning. Plots show mean and standard error over 10 trials.

6.5.5 Amortizing the cost of lookahead decision making

Recall from Section 6.1 that our exploration of generative sampling approaches in this setting was motivated by the goal of improving the time complexity of decision making while preserving the performance advantages of lookahead in active search. In Table 6.1 we compare the CPU wall clock time per decision making step in CAST with the GPU time for sampling lookahead action sequences and selecting the next cost-aware sensing action in CD-AS. For a 1D search space of size 1×16, at every decision step, CAST constructs a search tree of depth 2 with 5000 rollouts for a measured wall clock time of 120.40 secs. In contrast, CD-AS takes 78.68 secs to sample a batch of 10000 action sequences of lookahead depth 8 using 32 denoising steps. In the 2D search space of size 8×8, while CAST requires 405.47 secs to build a search tree of depth 2 with 25000 rollouts, it only takes 132.75 secs for CD-AS to sample a batch of 100 action sequences with lookahead depth 10 using 32 denoising steps. In both settings, we observe that denoising diffusion sampling in CD-AS allows for longer horizon rollouts at a lower time complexity. In future work, with higher quality datasets and advanced training techniques, we believe that the inference time performance of CD-AS can be further improved thereby providing an efficient alternative approach to tree search for planning and lookahead decision making in active search.

$n_\ell \times n_w$	CAST	CD-AS
1×16	120.40	78.68
8×8	405.47	132.75

Table 6.1: **Comparing the average wall-clock time in seconds per decision making step with CAST and our proposed CD-AS in both 1D and 2D search spaces.** CD-AS amortizes the time complexity of rolling out state-action sequences in tree search based approaches for lookahead decision making and leads to more than 30% improvement per decision step.

7 | Conclusion

7.1 Thesis Summary

Interactive decision making with multi-agent teams is a ubiquitous component of applications with software agents as well as physical embodied robots. With the increasing adoption of autonomous agents in our daily lives, ranging from companion chatbots, AI tutors to even AI scientists and assistive robots, it becomes imperative to develop robust solutions to many of the challenges faced by existing planning and decision making algorithms. In this thesis, we proposed the multi-agent active search framework to characterize a practical setting encompassing several applications like robotic search and rescue, wildlife patrolling, environment monitoring and others. Our assumptions of decentralized decision making and asynchronous inter-agent communication reflect real world constraints for deploying such multi-robot teams. Specifically, we focused on the following four distinct challenges faced in practice. In Chapter 3, we discussed cost-awareness in decision making with lookahead tree search in CAST. Chapter 4 demonstrated the importance of uncertainty modeling due to sensor noise in the agent's belief representation and Chapter 5 discussed methods for modeling non-stationarity in the posterior belief. Finally, in Chapter 6 we discussed generative modeling approaches for adaptive lookahead decision making in active search.

7.2 Current Limitations and Future Directions

The algorithms proposed in this thesis demonstrate improvements in addressing the challenges we discussed in the respective problem settings. However we would be remiss not to draw attention to the challenges that still remain to be solved for practical real world deployment of multi-agent systems capable of active decison making.

The algorithms proposed in Part I and Part II involved online decision making approaches that can generalize to different search regions but might not be scalable to larger action spaces. Part III, in contrast, showed that a learning based approach would be scalable but its ability to generalize across environments is limited by our ability to collect high quality decision making datasets. This trade-off between decision quality versus decision efficiency is an important practical consideration for developing real world systems. With the current focus in the research community on developing foundation models for various tasks, we observe similar ideas in approaches like Chain-of-Thought (Wei et al., 2022) proposed for harder reasoning tasks. While these methods are being primarily developed with software AI agents in mind, perhaps future work would also benefit from exploring how such reasoning algorithms integrate with embodied agents that have diverse physical capabilities for different tasks in manipulation, navigation, etc. Preliminary research in this

direction has developed methods like SayCan (Ahn et al., 2022) but they do not solve many of the open problems in long horizon reasoning or ambiguity in task specification and are primarily limited to single agent settings.

Throughout this thesis, we have leveraged the stochasticity in Thompson sampling (TS) as a decentralized decision making algorithm in multi-agent systems with asynchronous inter-agent communication. While TS provides a flexible framework that is scalable with teamsize, Chapter 2 discussed the challenges involved with TS policies in information-gathering tasks. Moreover, in Chapter 3, the batch size for TS at the root node of the search tree is a parameter which needs to be carefully tuned so that agents do not end up constructing the same search tree. Given these limitations, future work might benefit from learning strategic decision making policies in multi-agent active search that can adaptively enable inter-agent collaboration conditioned on the task or objectives. In scenarios with environment non-stationarity where the agent's policy can affect the behavior of targets, future work can leverage the literature on principal-agent mechanism design for adaptive decision making between strategic interacting agents (Ivanov et al., 2024; Banerjee et al., 2023c).

Finally, in Chapter 6 we discussed some of the challenges involved in training trajectory models, return and spatial distance estimators for active search. Although sequence models like transformers, state space models and diffusion models are recently being explored for different RL settings, Section 6.4.3 demonstrates the need for benchmarks that consider environments with stochastic state dynamics, observation noise and partial observability. Network architectures that consider symmetries in the state space and action space can improve generalization to unseen environments as well as improve the sample efficiency during model training. Moreover, with the increasing adoption of generative models in different applications, it is crucial to evaluate the robustness of such approaches to different sources of uncertainty in decision making tasks, for example, different levels of observation noise during training and evaluation, variable number of interacting agents, etc. Consequently, future work in learning based approaches to active decision making could involve exploring alternate network architectures and training strategies that can address these challenges with the goal of developing scalable and robust systems for real world applications of multi-agent active search.

Bibliography

Yasin Abbasi-Yadkori, David Pal, and Csaba Szepesvari. Online-to-confidence-set conversions and application to sparse stochastic bandits. In *Artificial Intelligence and Statistics*, pages 1–9, 2012. 2.3

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 7.2

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B Tenenbaum, Tommi S Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023. 6.3.2

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *arXiv preprint arXiv:2405.12399*, 2024. 6.3.2

J-Y Audibert, R Munos, and C Szepesvari. Use of variance estimation in the multi-armed bandit problem. In *NIPS Workshop on On-line Trading of Exploration and Exploitation*, 2006. 3.4.3

Javad Azimi, Alan Fern, and Xiaoli Z Fern. Batch Bayesian optimization via simulation matching. In *Advances in Neural Information Processing Systems*, pages 109–117, 2010. 2.3

Javad Azimi, Alan Fern, Xiaoli Zhang-Fern, Glencora Borradaile, and Brent Heeringa. Batch active learning via coordinated matching. *arXiv preprint arXiv:1206.6458*, 2012. 2.1, 2.3

Nikhil Angad Bakshi, Tejus Gupta, Ramina Ghods, and Jeff Schneider. Guts: Generalized uncertainty-aware thompson sampling for multi-agent active search. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7735–7741. IEEE, 2023. 1.1.3, 5.4

Tucker Balch and Ronald C. Arkin. Communication in reactive multiagent robotic systems. *Auton. Robots*, 1(1):27–52, feb 1995. ISSN 0929-5593. doi: 10.1007/BF00735341. URL `https://doi.org/10.1007/BF00735341`. 1.1.1

Arundhati Banerjee and Jeff Schneider. Decentralized multi-agent active search and tracking when targets outnumber agents. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7229–7235, 2024. doi: 10.1109/ICRA57147.2024.10609977. 1.2

Arundhati Banerjee, Ramina Ghods, and Jeff Schneider. Cost-awareness in multi-agent active search. In *ECAI 2023*, pages 182–189. IOS Press, 2023a. 1.2, 1.2, 6.5.1

Arundhati Banerjee, Ramina Ghods, and Jeff Schneider. Multi-agent active search using detection and location uncertainty. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7720–7727. IEEE, 2023b. 1.2

Arundhati Banerjee, Soham Phade, Stefano Ermon, and Stephan Zheng. Mermaide: Learning to align learners using model-based meta-learning. *arXiv preprint arXiv:2304.04668*, 2023c. 7.2

Filippo Basso, Alberto Pretto, and Emanuele Menegatti. Unsupervised intrinsic and extrinsic calibration of a camera-depth sensor couple. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6244–6249, 2014. doi: 10.1109/ICRA.2014.6907780. 4.3

Filippo Basso, Emanuele Menegatti, and Alberto Pretto. Robust intrinsic and extrinsic calibration of RGB-D cameras. *IEEE Transactions on Robotics*, 34(5):1315–1332, 2018. 4.3

Amira Belhedi, Adrien Bartoli, Steve Bourgeois, Kamel Hamrouni, Patrick Sayd, and Vincent Gay-Bellile. Noise modelling and uncertainty propagation for tof sensors. In *European Conference on Computer Vision*, pages 476–485. Springer, 2012. 4.2.2, 4.3

D S Bernstein, R Givan, N Immerman, and S Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4), 2002. 3.3

G Best, O M Cliff, T Patten, R R Mettu, and R Fitch. Decentralised monte carlo tree search for active perception. In *Algorithmic Foundations of Robotics XII*. Springer, 2020. 1.2, 3.3

Graeme Best, Oliver M Cliff, Timothy Patten, Ramgopal R Mettu, and Robert Fitch. Dec-MCTS: Decentralized planning for multi-robot active perception. *The International Journal of Robotics Research*, 38(2-3):316–337, 2019. 2.1, 2.3

Samuel S Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, 2004. 5.3

Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009. 2.4.2

Gabor Braun, Sebastian Pokutta, and Yao Xie. Info-greedy sequential adaptive compressed sensing. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 858–865, 2014. doi: 10.1109/ALLERTON.2014.7028544. 1.1.2

Gábor Braun, Sebastian Pokutta, and Yao Xie. Info-greedy sequential adaptive compressed sensing. *IEEE Journal of selected topics in signal processing*, 9(4):601–611, 2015. 2.1, 2.3

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*, 4(1), 2012. 3.3

Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization

and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6): 1309–1332, 2016. doi: 10.1109/TRO.2016.2624754. 4.3

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. `https://github.com/huggingface/lerobot`, 2024. 6.5.1

V. Caglioti, A. Citterio, and A. Fossati. Cooperative, distributed localization in multi-robot systems: a minimum-entropy approach. In *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*, pages 25–30, 2006. doi: 10.1109/DIS.2006.20. 4.3

Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2488–2496, 2015. 4.3

Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006. 2.3, 2.4.2

J. Carlson and R.R. Murphy. How ugvs physically fail in the field. *IEEE Transactions on Robotics*, 21(3):423–437, 2005. doi: 10.1109/TRO.2004.838027. 1

Avishy Carmi, Pini Gurfil, and Dimitri Kanevsky. Methods for Sparse Signal Recovery Using Kalman Filtering With Embedded Pseudo-Measurement Norms and Quasi-Norms. *IEEE Transactions on Signal Processing*, 58(4):2405–2409, 2010. doi: 10.1109/TSP.2009.2038959. 4.3

Alexandra Carpentier and Remi Munos. Bandit theory meets compressed sensing for high dimensional stochastic linear bandit. In *Artificial Intelligence and Statistics*, pages 190–198, 2012. 2.3

Mattia Carpin, Stefano Rosati, Mohammad Emtiyaz Khan, and Bixio Rimoldi. Uavs using bayesian optimization to locate wifi devices. *arXiv preprint arXiv:1510.03592*, 2015. 1.1.2, 2.3

Guangda Chen, Guowei Cui, Zhongxiao Jin, Feng Wu, and Xiaoping Chen. Accurate Intrinsic and Extrinsic Calibration of RGB-D Cameras With GP-Based Depth Correction. *IEEE Sensors Journal*, 19(7):2685–2694, 2019a. doi: 10.1109/JSEN.2018.2889805. 1.2, 4.3

J. Chen, T. Shu, T. Li, and C.W. de Silva. Deep reinforced learning tree for spatiotemporal monitoring with mobile robotic wireless sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, PP(99):1–15, 2019b. ISSN 21682216. 2.3

Jun Chen and Philip Dames. Distributed multi-target search and tracking using a coordinated team of ground and aerial robots. In *Robotics science and systems*, 2020. 5.3

Jun Chen and Philip Dames. Active multi-target search using distributed thompson sampling. 2022. 5.1, 5.4, 5.4

Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001. 2.4.2

W Chen and L Liu. Pareto monte carlo tree search for multi-objective informative planning. In *Robotics: Science and Systems*, 2019. 1.2, 3.3

TCE Cheng, Boris Kriheli, Eugene Levner, and CT Ng. Scheduling an autonomous robot searching for hidden targets. *Annals of Operations Research*, pages 1–15, 2019. 4.3

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024. 6.4.2, 6.5.1

Shih-Yi Chien, Huadong Wang, and Michael Lewis. Human vs. algorithmic path planning for search and rescue by robot teams. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 54, 09 2010. doi: 10.1177/154193121005400423. 2.4.2

Timothy H. Chung and Joel W. Burdick. Analysis of Search Decision Making Using Probabilistic Search Strategies. *IEEE Transactions on Robotics*, 28(1):132–144, 2012. doi: 10.1109/TRO.2011.2170333. 4.3

R Coulom. Computing "elo ratings" of move patterns in the game of go. *ICGA journal*, 30 (4), 2007. 2

Philip Dames, Pratap Tokekar, and Vijay Kumar. Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots. *The International Journal of Robotics Research*, 36(13-14):1540–1553, 2017. 1.2, 2.3, 5.3, 5.5.2

Philip M Dames. Distributed multi-target search and tracking using the phd filter. *Autonomous robots*, 44(3-4):673–689, 2020. 5.2.2, 5.3

Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004. 2.4.2

Mark A Davenport and Ery Arias-Castro. Compressive binary search. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 1827–1831. IEEE, 2012. 2.3

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 6.3.1

Lefteris Doitsidis, Stephan Weiss, Alessandro Renzaglia, Markus W Achtelik, Elias Kosmatopoulos, Roland Siegwart, and Davide Scaramuzza. Optimal surveillance coverage for teams of micro aerial vehicles in gps-denied environments using onboard vision. *Autonomous Robots*, 33:173–188, 2012. 5.3

David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4): 1289–1306, 2006a. 2.3, 2.4.2

D.L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4): 1289–1306, 2006b. doi: 10.1109/TIT.2006.871582. 1.1.2

Arnaud Doucet, Nando De Freitas, Neil James Gordon, et al. *Sequential Monte Carlo methods in practice*, volume 1. Springer, 2001. 5.3

John T Feddema, Chris Lewis, and David Schoenwald. Decentralized control of cooperative robotic vehicles: theory and applications. IEEE, 2002. 2.1

Johannes Fischer and Ömer Sahin Tas. Information particle filter tree: An online algorithm for pomdps with belief-based rewards on continuous domains. In *International Conference on Machine Learning*, pages 3177–3187. PMLR, 2020. 3.3

Genevieve Flaspohler, Victoria Preston, Anna PM Michel, Yogesh Girdhar, and Nicholas Roy. Information-guided robotic maximum seek-and-sample in partially observable continuous environments. *IEEE Robotics and Automation Letters*, 4(4):3782–3789, 2019. 2.1, 3.3

Thomas Fortmann, Yaakov Bar-Shalom, and Molly Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE journal of Oceanic Engineering*, 8(3):173–184, 1983. 5.3

R Ghods, W J Durkin, and J Schneider. Multi-agent active search using realistic depth-aware noise model. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021a. 1.2, 4.2.1, 4.2.4, 4.3, 4.5, 4.5.1, 4.6, 4.7

Ramina Ghods, Arundhati Banerjee, and Jeff Schneider. Decentralized multi-agent active search for sparse signals. In *Uncertainty in Artificial Intelligence*, pages 696–706. PMLR, 2021b. 1.1, 1, 2.5.3, 2.6.1, 3.2, 3.3.1, 3.5, 3.5.1, 3.5.1, 3.5.6, 4.2.4, 4.3, 4.3.1, 4.6, 4.7, 5.4, 5.5.1, 6.5.1

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011. 3.3

Abel Gonzalez-Garcia, Alexander Vezhnevets, and Vittorio Ferrari. An active search strategy for efficient object class detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3022–3031, 2015. 4.3

Quanquan Gu, Tong Zhang, and Jiawei Han. Batch-mode active learning via error bound minimization. In *UAI*, pages 300–309, 2014. 2.1, 2.3

David Hall, Feras Dayoub, John Skinner, Haoyang Zhang, Dimity Miller, Peter Corke, Gustavo Carneiro, Anelia Angelova, and Niko Suenderhauf. Probabilistic Object Detection: Definition and Evaluation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020. 4.3

Jarvis Haupt, Robert Nowak, and Rui Castro. Adaptive sensing for sparse signal recovery. In *2009 IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop*, pages 702–707. IEEE, 2009a. 2.3

Jarvis D Haupt, Richard G Baraniuk, Rui M Castro, and Robert D Nowak. Compressive distilled sensing: Sparse recovery using adaptivity in compressive measurements. In *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*, pages 1551–1555. IEEE, 2009b. 2.3

Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):1–59, 2022. 3.3

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 6.3.1

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 6.3.1

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video:

High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022a. 6.3.1

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022b. 6.3.1

Baichuan Huang, Jun Zhao, and Jingbin Liu. A survey of simultaneous localization and mapping. *arXiv preprint arXiv:1909.05214*, 2019. 1.2, 2.3

Conor Igoe, Ramina Ghods, and Jeff Schneider. Multi-Agent Active Search: A Reinforcement Learning Approach. *IEEE Robotics and Automation Letters*, 7(2):754–761, 2022. doi: 10.1109/LRA.2021.3131697. 4.3

Conor Igoe, Tejus Gupta, and Jeff Schneider. Efficient bayesian experiment design with graph neural networks. *In submission*, 2024. 6.3.3, 6.4.2

Rain Industries. Rain industries announces autonomous firefighting helicopter, 2022. URL `https://verticalmag.com/press-releases/rain-industries-announces-autonomous-firefighting-helicopter`. 1.1

Dima Ivanov, Paul Dütting, Inbal Talgam-Cohen, Tonghan Wang, and David C Parkes. Principal-agent reinforcement learning: Orchestrating ai agents with contracts. *arXiv preprint arXiv:2407.18074*, 2024. 7.2

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022. 1.2, 6.1, 6.3.1, 6.3.2, 6.3.4, 6.4.2, 6.4.2, 6.4.2, 6.4.2

Bruno Jedynak, Peter I Frazier, and Raphael Sznitman. Twenty questions with noise: Bayes optimal policies for entropy loss. *Journal of Applied Probability*, 49(1):114–136, 2012. 2.3

Heejin Jeong, Hamed Hassani, Manfred Morari, Daniel D Lee, and George J Pappas. Deep reinforcement learning for active target tracking. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1825–1831. IEEE, 2021. 5.3

S Jiang, R Garnett, and B Moseley. Cost effective active search. In *Advances in Neural Information Processing Systems*, 2019. 3.3.1

L P Kaelbling, M L Littman, and A R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2), 1998. 3.3

R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. 4.4.1, 5.3, 6.4.1

Michael Kampffmeyer, Arnt-Børre Salberg, and Robert Jenssen. Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 680–688, 2016. doi: 10.1109/CVPRW.2016.90. 4.3

Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Asynchronous parallel Bayesian optimisation via Thompson sampling. *arXiv preprint arXiv:1705.09236*, 2017. 1

Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised bayesian optimisation via thompson sampling. In *International Conference*

on Artificial Intelligence and Statistics, pages 133–142, 2018. 2.4, 2.5.3, 4.2.4, 4.6

Kirthevasan Kandasamy, Willie Neiswanger, Reed Zhang, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Myopic posterior sampling for adaptive goal oriented design of experiments. In *International Conference on Machine Learning*, pages 3222–3232, 2019. 2.4, 2.4.1, 3.4.3

D Kent and S Chernova. Human-centric active perception for autonomous observation. In *2020 IEEE ICRA*, 2020. 3.3

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006. 3.3, 3.4.3

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. 6.5.1, 6.5.3

Moshe Kress, Kyle Y Lin, and Roberto Szechtman. Optimal discrete search with imperfect specificity. *Mathematical methods of operations research*, 68(3):539–549, 2008. 4.3

Hung M La, Weihua Sheng, and Jiming Chen. Cooperative and active sensing in mobile sensor networks for scalar field mapping. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):1–12, 2014. 2.3

M Lauri and F Oliehoek. Multi-agent active perception with prediction rewards. *Advances in Neural Information Processing Systems*, 33, 2020. 3.3

M Lauri, J Pajarinen, and J Peters. Multi-agent active information gathering in discrete and continuous-state decentralized pomdps by policy graph improvement. *Autonomous Agents and Multi-Agent Systems*, 34(42), 2020. doi: 10.1007/S10458-020-09467-6. 2.1, 2.3, 3.3

S Lazebnik, C Schmid, and J Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, 2006. 3.2

Jongmin Lee, Geon-Hyeong Kim, Pascal Poupart, and Kee-Eung Kim. Monte-carlo tree search for constrained pomdps. *Advances in Neural Information Processing Systems*, 31, 2018. 3.3

John J Leonard and Hugh F Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IROS*, volume 3, pages 1442–1447, 1991. 2.3

Pei Li and Haibin Duan. A potential game approach to multiple UAV cooperative search and surveillance. *Aerospace Science and Technology*, 68:403–415, 2017. 2.3

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022. 6.3.1

Zhan Wei Lim, David Hsu, and Wee Sun Lee. Adaptive informative path planning in metric spaces. *The International Journal of Robotics Research*, 35(5):585–598, 2016. 3.1

Lanny Lin and Michael A. Goodrich. Uav intelligent path planning for wilderness search and rescue. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 709–714, 2009. doi: 10.1109/IROS.2009.5354455. 2.4.2

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017. 2.3

Yifei Ma, Roman Garnett, and Jeff Schneider. Active search for sparse signals with region sensing. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 1.1.2, 2.1, 2.2, 2.3, 2.6.2, 3.5, 3.5.1, 3.5.1, 4.3, 6.5.1

Ronald PS Mahler. Multitarget bayes filtering via first-order multitarget moments. *IEEE Transactions on Aerospace and Electronic systems*, 39(4):1152–1178, 2003. 5.2.1, 5.2.2, 5.3, 5.4

Ronald PS Mahler. *Advances in statistical multisource-multitarget information fusion*. Artech House, 2014. 5.2.1, 1

Arian Maleki. Approximate message passing algorithms for compressed sensing. *a degree of Doctor of Philosophy, Stanford University*, 2011. 2.4.2

Matthew L. Malloy and Robert D. Nowak. Near-Optimal Adaptive Compressed Sensing. *IEEE Transactions on Information Theory*, 60(7):4001–4012, 2014a. doi: 10.1109/TIT.2014.2321552. 1.2

Matthew L Malloy and Robert D Nowak. Near-optimal adaptive compressed sensing. *IEEE Transactions on Information Theory*, 60(7):4001–4012, 2014b. 2.3

Roman Marchant and Fabio Ramos. Bayesian optimisation for Intelligent Environmental Monitoring. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2242–2249, 2012a. doi: 10.1109/IROS.2012.6385653. 1.2

Roman Marchant and Fabio Ramos. Bayesian optimisation for intelligent environmental monitoring. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 2242–2249. IEEE, 2012b. 2.3

Stephen Miller, Alex Teichman, and Sebastian Thrun. Unsupervised extrinsic calibration of depth sensors in dynamic scenes. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2695–2702, 2013. doi: 10.1109/IROS.2013.6696737. 1.2, 4.3

Arnab Kumar Mondal, Vineet Jain, Kaleem Siddiqi, and Siamak Ravanbakhsh. Eqr: Equivariant representations for data-efficient reinforcement learning. In *International Conference on Machine Learning*, pages 15908–15926. PMLR, 2022. 6.3.3

Robin R Murphy. Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(2):138–153, 2004a. 2.1, 5.3

Robin R Murphy. Trial by fire [rescue robots]. *IEEE Robotics & Automation Magazine*, 11 (3):50–61, 2004b. 1

Robin R Murphy. It is hard to be a robot in the wild. *Science Robotics*, 10(99):eadw1608, 2025. 1

Robin R. Murphy, Satoshi Tadokoro, Daniele Nardi, Adam Jacoff, Paolo Fiorini, Howie Choset, and Aydan M. Erkmen. *Search and Rescue Robotics*, pages 1151–1173. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-30301-5. doi: 10.1007/978-3-540-30301-5_51. URL `https://doi.org/10.1007/978-3-540-30301-5_51`. 1.1

Robin R. Murphy, Jeffery Kravitz, Samuel L. Stover, and Rahmat Shoureshi. Mobile robots in mine rescue and recovery. *IEEE Robotics Automation Magazine*, 16(2):91–103, 2009. doi: 10.1109/MRA.2009.932521. 1

Robin R Murphy, Karen L Dreger, Sean Newsome, Jesse Rodocker, Eric Steimle, Tetsuya Kimura, Kenichi Makabe, Fumitoshi Matsuno, Satoshi Tadokoro, and Kazuyuki Kon. Use of remotely operated marine vehicles at minamisanriku and rikuzentakata japan for disaster recovery. In *2011 IEEE International symposium on safety, security, and rescue robotics*, pages 19–25. IEEE, 2011. 1

Preetum Nakkiran, Arwen Bradley, Hattie Zhou, and Madhu Advani. Step-by-step diffusion: An elementary tutorial. *arXiv preprint arXiv:2406.08929*, 2024. 6.3.1

Deanna Needell and Joel A Tropp. CoSaMP: iterative signal recovery from incomplete and inaccurate samples. *Communications of the ACM*, 53(12):93–100, 2010. 2.4.2

Deanna Needell and Roman Vershynin. Signal Recovery From Incomplete and Inaccurate Measurements Via Regularized Orthogonal Matching Pursuit. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):310–316, 2010. doi: 10.1109/JSTSP.2010.2042412. 4.3

Hai Huu Nguyen, Andrea Baisero, David Klee, Dian Wang, Robert Platt, and Christopher Amato. Equivariant reinforcement learning under partial observability. In *Conference on Robot Learning*, pages 3309–3320. PMLR, 2023. 6.3.3

Minh Tuan Nguyen. Distributed compressive and collaborative sensing data collection in mobile sensor networks. *Internet of Things*, page 100156, 2019. 2.3

F A Oliehoek, C Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016. 3.3

Thomas Oskam, Robert W Sumner, Nils Thuerey, and Markus Gross. Visibility transition planning for dynamic camera control. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 55–65, 2009. 5.3

Savvas Papaioannou, Panayiotis Kolios, Theocharis Theocharides, Christos G Panayiotou, and Marios M Polycarpou. A cooperative multiagent probabilistic framework for search and track missions. *IEEE Transactions on Control of Network Systems*, 8(2):847–858, 2020. 1.2, 5.1, 5.3, 5.4, 5.5.2

Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE, 1993. 2.4.2

R Pěnička, J Faigl, M Saska, and P Váňa. Data collection planning with non-zero sensing distance for a budget and curvature constrained unmanned aerial vehicle. *Autonomous Robots*, 43(8), 2019. 3.3

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 6.4.2

Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *Ijcai*, volume 3, pages 1025–1032, 2003. 3.3

Julio A Placed, Jared Strader, Henry Carrillo, Nikolay Atanasov, Vadim Indelman, Luca Carlone, and José A Castellanos. A survey on active simultaneous localization and mapping: State of the art and new frontiers. *IEEE Transactions on Robotics*, 39(3): 1686–1705, 2023. 4.3, 5.1

Marija Popović, Teresa Vidal-Calleja, Gregory Hitz, Jen Jen Chung, Inkyu Sa, Roland Siegwart, and Juan Nieto. An informative path planning framework for uav-based terrain monitoring. *Autonomous Robots*, 44:889–911, 2020. 5.1

Jorge Peña Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, Moncef Gabbouj, Jenni Raitoharju, and Tomi Westerlund. Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *IEEE Access*, 8:191617–191643, 2020. doi: 10.1109/ACCESS.2020.3030190. 2.1, 2.3

Purnima Rajan, Weidong Han, Raphael Sznitman, Peter Frazier, and Bruno Jedynak. Bayesian multiple target localization. In *International conference on machine learning*, pages 1945–1953. PMLR, 2015. 1.2, 2.3

Santhosh Kumar Ramakrishnan, Erik Wijmans, Philipp Kraehenbuehl, and Vladlen Koltun. Does spatial cognition emerge in frontier models? *arXiv preprint arXiv:2410.06468*, 2024. 6.5.3

Ram Ramanathan and Jason Redi. A brief overview of ad hoc networks: challenges and directions. *IEEE communications Magazine*, 40(5):20–22, 2002. 1.1.1

Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement, 2018. 4.2, 4.7

Branko Ristic, Daniel Clark, and Ba-Ngu Vo. Improved smc implementation of the phd filter. In *2010 13th International Conference on Information Fusion*, pages 1–8. IEEE, 2010. 5.2.2, 5.3, 5.4, 5.4, 6, 5.5

Branko Ristic, Ba-Ngu Vo, and Daniel Clark. A note on the reward function for phd filters with sensor control. *IEEE Transactions on Aerospace and Electronic Systems*, 47(2): 1521–1529, 2011. 5.5.2

Cyril Robin and Simon Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, 40(4):729–760, 2016. 2.1, 5.1, 5.3, 5.5.3

Esther Rolf, David Fridovich-Keil, Max Simchowitz, Benjamin Recht, and Claire Tomlin. A successive-elimination approach to adaptive robotic sensing. *ArXiv e-prints*, 2018. 2.3

Esther Rolf, David Fridovich-Keil, Max Simchowitz, Benjamin Recht, and Claire Tomlin. A successive-elimination approach to adaptive robotic source seeking. *IEEE Transactions on Robotics*, 37(1):34–47, 2020. 1.1.2, 2.1

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 6.4.2

Matthew Rosencrantz, Geoffrey Gordon, and Sebastian Thrun. Locating moving entities in indoor environments with teams of mobile robots. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 233–240, 2003. 5.3.1

Roxana Rădulescu, Patrick Mannion, Diederik M. Roijers, and Ann Nowé. Multi-objective multi-agent decision making: a utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems*, 34(1), 2019. ISSN 1387-2532. doi: 10.1007/s10458-019-09433-x. URL `https://doi.org/10.1007/s10458-019-09433-x`. 3.1

D Russo, B Van Roy, A Kazerouni, I Osband, and Z Wen. A tutorial on thompson sampling. *arXiv:1707.02038*, 2017. 3.4.3

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on Thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1): 1–96, 2018. 2.4, 2.4.2, 5.4

A. Ryan and J.K. Hedrick. A mode-switching path planner for uav-assisted search and rescue. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 1471–1476, 2005. doi: 10.1109/CDC.2005.1582366. 2.4.2

Lorenzo Sabattini, Nikhil Chopra, and Cristian Secchi. Decentralized connectivity maintenance for cooperative control of mobile robotic systems. *International Journal of Robotics Research*, 32(12):1411–1423, 2013. 2.1

D Shah, Q Xie, and Z Xu. Non-asymptotic analysis of monte carlo tree search. In *Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems*, 2020. 3.4.3

Yorai Shaoul, Itamar Mishani, Shivam Vats, Jiaoyang Li, and Maxim Likhachev. Multi-robot motion planning with diffusion models. *arXiv preprint arXiv:2410.03072*, 2024. 6.3.2

David Silver and Joel Veness. Monte-carlo planning in large pomdps. *Advances in neural information processing systems*, 23, 2010. 3.3

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016. 3

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017. 3

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. 6.3.1

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. 1.2, 6.1, 6.3.1

Lawrence D Stone, Roy L Streit, Thomas L Corwin, and Kristine L Bell. *Bayesian multiple target tracking*. Artech House, 2013. 5.3

Malcolm Strens. A Bayesian framework for reinforcement learning. In *ICML*, volume 2000, pages 943–950, 2000. 2.4

A.W. Stroupe, M.C. Martin, and T. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 2, pages 1092–1098 vol.2, 2001. doi: 10.1109/ROBOT.2001.932739. 4.3

F Sukkar, G Best, C Yoo, and R Fitch. Multi-robot region-of-interest reconstruction with dec-mcts. In *2019 International Conference on Robotics and Automation (ICRA)*, 2019. 3.3

Wennie Tabib, John Stecklein, Caleb McDowell, Kshitij Goel, Felix Jonathan, Abhishek Rathod, Meghan Kokoski, Edsel Burkholder, Brian Wallace, Luis Ernesto Navarro-Serment, et al. Decentralized uncertainty-aware active search with a team of aerial robots. *arXiv preprint arXiv:2410.08507*, 2024. 1.1.3

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933. 2.4

Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002. 5.5

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. 2.4.2

Michael E Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1(Jun):211–244, 2001. 2.5.1, 4.5.1

Mariliza Tzes, Nikolaos Bousias, Evangelos Chatzipantazis, and George J Pappas. Graph neural networks for multi-robot active information acquisition. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3497–3503. IEEE, 2023. 5.3

Hoa Van Nguyen, Ba-Ngu Vo, Ba-Tuong Vo, Hamid Rezatofighi, and Damith C Ranasinghe. Multi-objective multi-agent planning for discovering and tracking multiple mobile objects. *arXiv preprint arXiv:2203.04551*, 2022. 5.3, 5.4

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 6.4.2

Adam R Villaflor, Zhe Huang, Swapnil Pande, John M Dolan, and Jeff Schneider. Addressing optimism bias in sequence modeling for reinforcement learning. In *international conference on machine learning*, pages 22270–22283. PMLR, 2022. 6.3.4

Alberto Viseras, Juan Marchal, Marius Schaab, Jordi Pages, and Laia Estivill. Wildfire monitoring and hotspots detection with aerial robots: Measurement campaign and first results. In *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 102–103, 2019. doi: 10.1109/SSRR.2019.8848961. 1.1

Dian Wang, Robin Walters, and Robert Platt. So(2)-equivariant reinforcement learning. In *International Conference on Learning Representations*, 2022. 6.3.3

W Wang and M Sebag. Multi-objective monte-carlo tree search. In *Asian conference on machine learning*, 2012. 3.3

Zining Wang, Di Feng, Yiyang Zhou, Lars Rosenbaum, Fabian Timm, Klaus Dietmayer, Masayoshi Tomizuka, and Wei Zhan. Inferring Spatial Uncertainty in Object Detection. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5792–5799, 2020. doi: 10.1109/IROS45743.2020.9340798. 4.3

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 7.2

Wiki. Drones in wildfire management. *Wikipedia*. URL `https://en.wikipedia.org/wiki/Drones_in_wildfire_management`. 1

Peter M Williams. Bayesian regularization and pruning using a Laplace prior. *Neural computation*, 7(1):117–143, 1995. 2.4.2

David P Wipf and Bhaskar D Rao. Sparse Bayesian learning for basis selection. *IEEE Transactions on Signal processing*, 52(8):2153–2164, 2004. 2.5.1

Jeremy Wyatt. Exploration and inference in learning from reinforcement. 1998. 2.4

Pujie Xin and Philip Dames. Comparing stochastic optimization methods for multi-robot, multi-target tracking. In *International Symposium on Distributed and Autonomous Systems*, 2022. 5.3

Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12):399, 2013a. doi: 10.5772/57313. URL `https://doi.org/10.5772/57313`. 1.1.1

Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12):399, 2013b. 2.1

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*, 2023. 1

Sherry Yang, Jacob Walker, Jack Parker-Holder, Yilun Du, Jake Bruce, Andre Barreto, Pieter Abbeel, and Dale Schuurmans. Video as the new language for real-world decision making. *arXiv preprint arXiv:2402.17139*, 2024. 6.1

Michael M. Zavlanos, Alejandro Ribeiro, and George J. Pappas. Network integrity in mobile robotic networks. *IEEE Transactions on Automatic Control*, 58(1):3–18, 2013. doi: 10.1109/TAC.2012.2203215. 1.1.1

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021. 2.3

Yihao Zhang, Odin A Severinsen, John J Leonard, Luca Carlone, and Kasra Khosoussi. Data-association-free landmark-based slam. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8349–8355. IEEE, 2023. 4.3

Zhilin Zhang and Bhaskar D Rao. Sparse signal recovery with temporally correlated source vectors using sparse Bayesian learning. *IEEE Journal of Selected Topics in Signal Processing*, 5(5):912–926, 2011. 2.5.1, 2.5.1, 3.4.2

Zhilin Zhang and Bhaskar D Rao. Extension of SBL algorithms for the recovery of block sparse signals with intra-block correlation. *IEEE Transactions on Signal Processing*, 61 (8):2009–2015, 2013. 2.5.1

Guangyao Zhou, Sivaramakrishnan Swaminathan, Rajkumar Vasudeva Raju, J Swaroop Guntupalli, Wolfgang Lehrach, Joseph Ortiz, Antoine Dedieu, Miguel Lázaro-Gredilla, and Kevin Murphy. Diffusion model predictive control. *arXiv preprint arXiv:2410.05364*, 2024. 6.3.2

Lifeng Zhou, Vishnu D Sharma, Qingbiao Li, Amanda Prorok, Alejandro Ribeiro, Pratap Tokekar, and Vijay Kumar. Graph neural networks for decentralized multi-robot target tracking. In *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 195–202. IEEE, 2022a. 5.3

Qian-Yi Zhou and Vladlen Koltun. Simultaneous localization and calibration: Self-calibration of consumer depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 454–460, 2014. 1.2, 4.3

Zeyu Zhou, Bruce Hajek, Nakjung Choi, and Anwar Walid. Regenerative particle thompson sampling. *arXiv preprint arXiv:2203.08082*, 2022b. 5.4

Zhengbang Zhu, Minghuan Liu, Liyuan Mao, Bingyi Kang, Minkai Xu, Yong Yu, Stefano Ermon, and Weinan Zhang. Madiff: Offline multi-agent learning with diffusion models. *arXiv preprint arXiv:2305.17330*, 2023. 6.3.2

David Zuñiga-Noël, Jose-Raul Ruiz-Sarmiento, and Javier Gonzalez-Jimenez. Intrinsic calibration of depth cameras for mobile robots using a radial laser scanner. In *International Conference on Computer Analysis of Images and Patterns*, pages 659–671. Springer, 2019. 4.3