Sensor Planning for Large Numbers of Robots

Micah Corah

September 21, 2020

CMU-RI-TR-20-53

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Nathan Michael, *Chair*
Anupam Gupta
Katia Sycara
Mac Schwager, Stanford

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

For my parents, David and Barbara, who supported and encouraged my interest in robotics at every step of this journey.

Abstract

In the wake of a natural disaster, locating and extracting victims quickly is critical because mortality rises rapidly after the first forty-eight hours. In order to assist search and rescue teams and improve response times, teams of aerial robots equipped with sensors and cameras can engage in sensing tasks such as mapping buildings, assessing structural integrity, and locating victims. We seek to enable large numbers of robots to cooperate to complete such sensing tasks more quickly and thereby to improve response times for first responders.

When formalized, such sensing tasks encapsulate numerous computationally difficult (at least NP-Hard) problems related to routing, sensor coverage, and decision processes. One way to simplify planning for these tasks is to focus on maximizing sensing performance over a short time horizon. Specifically, consider the problem of how to select motions for a team of robots to maximize a notion of sensing quality (the sensing objective) over the near future, say by maximizing the amount of unknown space in a map that robots will observe over the next several seconds. By repeating this process regularly (about once a second), the robots can react quickly to new observations as they work to complete the sensing task. In technical terms, this planning and control process forms an example of receding-horizon control. Fortunately, common sensing objectives for these problems benefit from well-known monotonicity properties (e.g. submodularity), and greedy algorithms can exploit these monotonicity properties to solve the receding-horizon optimization problems that we study near-optimally.

However, greedy algorithms typically force robots to make decisions sequentially. Thus, planning time grows with the number of robots and eventually exceeds time constraints to replan in real time. This is particularly important in distributed settings as the accumulation of communication latencies between robots in sequence can be significant on its own. Further, recent works that have begun to investigate sequential greedy planning, have demonstrated that reducing the number of sequential steps while retaining suboptimality guarantees can be hard or impossible.

This thesis demonstrates that halting such growth in planning time is possible for many sensing problems. To do so, we introduce new greedy methods, Randomized Sequential Partitions (RSP) and Range-limited RSP. These methods enable planning with a fixed number of sequential steps that does not grow with the number of robots. Additionally, we prove that our algorithms approach the initial near-optimality guarantees for sequential planning for many sensing problems. In doing so, we develop new methods for quantifying redundancy between potential future observations and highlight the importance of a relatively unknown monotonicity property of some sensing objectives.

We apply our algorithms to autonomous mapping (known as exploration) and target tracking problems which serve as proxies for the variety of tasks and combinations of tasks that may arise in search and rescue scenarios. Simulation results demonstrate that our greedy planners often approach the performance of sequential planning (in terms of target position uncertainty) given only a few planning steps (2-4), even for very large numbers of robots (96). This amounts to a $24\times$ reduction in the number of sequential steps and an equivalent or greater reduction in the duration of distributed planning.

With exploration, we apply our methods in the context of a complex system where robots equipped with depth cameras map unknown, three-dimensional environments such as an office space or a cave. Moreover, the analysis we present when applying our planning methods to mapping objectives also provides valuable insights into the design of such exploration systems. While consistently improving completion times via greedy methods proves challenging, we demonstrate that sequential planning and RSP increase coverage rates early in simulation trials and reliably improve solution quality. Finally, we present a distributed implementation of RSP and simulation results for exploration of an office environment in real time, along with a 5% improvement in task completion time in this setting.

Acknowledgments

I would like to begin by thanking my family for their ongoing support. In particular, my parents have encouraged my interests in robotics for quite a long time and put up with quite a nerdy youngster. They deserve more than just my thanks. I should also thank my partner, Srujana. She kept me alive, happy, and well fed during the final and most harrowing months of my doctorate.

I also have many to thank in the Resilient Intelligent Systems Lab. My officemates, Ellen and Derek were ever-present companions. Ellen, I believe our fish tank set new standards for office decor as well as undergraduate achievement. Derek, aside from being my office compatriot, you were also an early mentor when I was still an intern. Thank you for that time. And, Vibhav, you were my replacement, but I preferred being students together. Kshitij and Cormac, we collaborated on some of the best work that is not in this thesis. Thank you Curt for keeping my robots alive before I learned to avoid them. And, thanks Wennie for doing the hard work to develop systems for exploration and mapping. Vishnu, thank you for demonstrating the first feasible solution to the graduation problem. Your leadership was invaluable. Shaurya, thank you for being someone who I can talk to about research and everything else. Also, John, the pandemic has taken our pizza, and I want it back. Thanks to everyone else in the lab who has made my time here brighter, including Alex, Xuning, Aditya, Arjav, Mosam, Moses, Tim, Matt, Mike, Erik, and anyone else who I may have forgotten. Also, Karen Widmaier, you put in a good deal of hard work for our lab, and you make our hearts warmer.

Many thanks to my friends in the Robotics Institute. Achal and Radhika, Dhruv and Cara, and now I am compelled to mention Xuning again, thank you for the adventures that brought so much joy to this time. Shushman, you are a caring friend. I did not introduce you to my family; I introduced my family to you. Thank you Reuben, you speak softly, but your words carry great weight. Nick, you have done great things for the RI, but you also do great things for frogs.

Thanks to everyone else who contributed to my time here. Rachel Burcin, I am only one of a vast number of people who have been significantly impacted by your tireless enthusiasm administrating RISS. I could not keep up with you. Reid, you welcomed me to CMU twice, and I hope to someday become more like you. Thank you Michael Erdmann, you allowed me to babble about submodular functions in front of 16-811. Thanks Sankalp, your words in front of same class introduced me to same topic. My thanks also to Roie for validating my interest in 3-increasing functions and for our too infrequent exchanges on the topic.

Thank you Nate for guiding my journey in robotics at CMU. You accepted my stubbornness and pushed me to excellence. Thanks also to the rest of my committee, Anupam, Katia, and Mac for direction and advice. I hope these first few pages are to your liking; for the next hundred, well I am not Asimov.

Contents

List of Figures

List of Tables

List of Algorithms

Notation

General mathematical notation

$\mathbb{R}$, $\mathbb{R}_{>0}$	The set of real numbers and real numbers greater than zero respectively
$\mathbb{Z}$	The set of integers
$\mathbb{SO}(3)$	The special orthogonal group in three dimensions (rotations)
$\mathbb{SE}(3)$	The special Euclidean group for three dimensions (rigid-body motions)
$\mathbf{p}$, $\mathbf{v}$	Examples of vectors
$A_{1:n}$, A_X	Examples of indexing "sets." Elements of sets will generally be implicitly associated with time or agent indices which will be accessed using ranges and sets of indices
$X^\star$, X^{g}, X^{d}	Superscripts designating (elements of) an optimal solution to an optimization problem and examples of output from greedy and distributed algorithms as will be evident from context

Information theory and probability

$\mathbb{P}(X\!=\!x\vert Y\!=\!y)$	Probability that a random variable X equals x conditional on $Y\!=\!y$
$\mathbb{E}[X]$	The expected value of X
$\mathbb{H}(X\vert Y)$	Entropy of X conditional on Y
$\mathbb{I}(X;Y\vert Z)$	Mutual information between X and Y conditional on Z

1

Combinatorial optimization

$\mathscr{U}$	The ground set
$\mathscr{I}$	An independence system
$\mathscr{B}$	A block of a partition (matroid)
$g(A, B)$, $g(x)$	Evaluation of a set function g at $A \cup B$ where $A, B \subseteq \mathscr{U}$ and implicit evaluation at $\{x\}$ where $x \in \mathscr{U}$
$g(A\|B)$, $g(A; B\|C)$	The former shows evaluation at A "conditional" on B ($g(A, B) - g(A)$) using similar notation as for mutual information. More generally, these two are first and second discrete derivatives of g where $g(A; B\|C) = g(B; A\|C) = g(A, B, C) - g(A, C) - g(B, C) + g(C)$ (with A, B, C disjoint).

Graphs

$\mathcal{G}$	A (directed or undirected) graph
$\mathcal{E}$	The set of edges of a graph
$\tilde{\mathcal{E}}$	Edges removed from a complete graph, $\mathcal{R} \times \mathcal{R} \setminus \mathcal{E}$ if robots are vertices
$\mathcal{W}$	Edge weights

As hinted above, we will forgo use of a symbol for vertices in favor of the symbol for the set of objects—generally robots or agents—in question.

Robotics and control

$\mathcal{R}$	A set of (mobile) robots
$\mathcal{A}$	A set of (immobile) agents
$\mathcal{U}$	A set of controls (or sometimes actions)
$f(\mathbf{x}, u)$	System dynamics given state $\mathbf{x}$ and control u (generally for an individual robot)
$h(\mathbf{x})$	Sensing or observation function (also for individual robots)

Algorithm acronyms

SGA	Sequential greedy assignment for submodular maximization on a partition matroid (acronym used in Chapter 4)
DSGA, DSGA$_{n_\mathrm{d}}$	Distributed sequential greedy assignment in general and in n_d rounds, an early version of the methods this thesis proposes
RSP, R-lRSP	Randomized sequential partitions and the range-limited variant, the main contributions of this thesis

Chapter 1

Introduction

Robots are often characterized as being able to *sense* their environment and *act* upon it [25]. While the actions that a robot can perform upon an environment—say folding laundry [130] or preparing a meal—are important [86], sensing is becoming increasingly prominent. This shift has been driven by changes in the markets for both robot platforms and the sensors they carry:

- Depth sensors and cameras have become increasingly prevalent, cheaper, and lighter, and

- Aerial robots have become both popular and pervasive [65, p. 41–51].

Moreover, aerial robots, when equipped with appropriate sensors, are able to navigate complex [61] and unknown [120] environments to accomplish sensing tasks such as videography [22], monitoring crops [17], and mapping [36].

Sensing problems arising from urban disaster and emergency response and in defense settings are time-sensitive and involve operation in unknown and cluttered environments. Sometimes, the time-sensitive nature of a task is immediate. One such case is a fire wherein robots may assist firefighters to outpace the effects of a conflagration. Here, a team of robots might engage in sensing tasks immediately preceding or in parallel with firefighters entering a building.[1] Disaster response has similar features, but the time-sensitivity is driven by scale. In the case of a widespread disaster, teams may have ample time to inspect individual buildings, but when viewed as a whole, the number of locations that must be inspected and the paucity of responders motivate rapid action. In particular, mortality typically increases rapidly forty-eight hours after the start of a disaster while search and rescue teams often operate on a first-come-first-serve basis [138].

Figure 1.1 illustrates an example of how a team of robots may contribute to sensing tasks that arise while searching a building in a disaster response scenario. For the purpose of this thesis, we will focus on applications and challenges related to disaster and emergency response scenarios such as this, although many of the contributions are relevant to other sensing tasks and optimization problems.

---

[1]While human-robot interaction is not a subject of this thesis, humans can be treated as additional members of a team that may provide additional sensor data, dictate sensing objectives, and must not be collided with.

Figure 1.1: This figure illustrates a typical disaster response scenario. Here, first responders wish to inspect a disaster site, such as after an earthquake, in order to ascertain structural integrity and to locate any survivors. To do so, they deploy a team of robots from a mobile base station that they transport along with the rest of their equipment, and the operator uses a tablet interface to instruct the robots to inspect the disaster site. Because communication infrastructure may have been destroyed in the disaster, the team has access to little prior information on the environment, and robots take advantage of distributed, onboard computation to mitigate effects of unreliable communication.

When seeking to improve response times, a designer may work to improve the sensing platform itself along with planning and control algorithms for individual robots [76, 77]. However, platform constraints will ultimately limit the performance of individual robots. Additionally, teams of robots can cooperate in sensing tasks to cover more space at once or to complete tasks more rapidly. We focus on this latter dimension of the problem of improving response times and ask: *how can a designer compose multiple engineered sensing platforms to improve completion time in sensing tasks?* Addressing this question for large teams of robots will involve solving increasingly large planning problems, addressing communication constraints, and accounting for complications such as inter-robot collision constraints.

As sensing tasks progress, the robots' individual and collective understanding of the world (commonly referred to as belief) can also change rapidly. The process of updating control actions in response to new sensor data is commonly referred to as adaptation [6, 79, 91]. Adaptation is particularly important for robots operating in cluttered and urban environments. In these scenarios, as robots obtain new sensor data, they can become aware of both new regions of interest and new unoccupied and traversable space. Often, as for a robot that is moving toward unobserved space [76, 77], this data is immediately pertinent to the selection of control actions. By extension, teams of robots that are able complete sensing tasks rapidly while operating in unknown environments must also adapt frequently.

4

1.1 Challenges for distributed sensing and time-sensitive sensing domains

The challenges that arise in this thesis can be divided into two categories: technical challenges related to the design of sensing systems and domain challenges for urban search and rescue. While the domain challenges will motivate and shape the methods we propose, the following technical challenges apply more generally to aerial sensing systems:

- **Size, Weight, and Power (SWaP) constraints:** Robots, especially aerial robots, have limitations in size and weight based on the application domain. These constraints are pervasive and can lead to further constraints on thrust, flight time, on-board computation, and sensor payloads

- **Safety:** Robots may operate in close proximity, replan frequently, and have non-trivial dynamics. Maintaining safety involves avoiding collisions between robots and other objects despite uncertainties in the states of the robots and the environment. Moreover, systems should be robust to planners that may sometimes fail to provide results, and safety may also include more complex conditions such as the ability to return to an initial state or avoiding hazards

- **Sensing:** The sensors that aerial robots carry are not trivial. Camera views are sensitive to orientation, and observations are complicated by the geometry of occlusions. Robots may also carry a variety of sensors such as thermal cameras, range sensors, or gas detectors, and autonomous sensing systems must account for sensor models, fuse data, and reason about the contributions of sensing actions

- **Communication:** Communication bandwidth between robots and operators may also be limited, and communication links may be unreliable. Communication constraints also affect dissemination of sensor data, operation of distributed algorithms, and interaction with human operators

- **Computation:** Algorithms related to planning and control may also scale poorly (e.g. being NP-Hard) and have to be solved suboptimally or approximately to be tractable. Computational units situated on the robots themselves or elsewhere must also have access to relevant information such as the robots' locations and sensor data

Then, considering the application of methods for time-sensitive sensing for disaster and emergency response in urban environments narrows these challenges and produces more concrete details:

- **Size:** Systems—referring to the entire multi-robot system and related equipment—that are usable by teams of first responders are likely limited in size to what one or two people can carry or what is portable with a small vehicle [138]

- **Hazardous environments:** Environments may contain water, dust, gasses, and other materials that are hazardous to people and robots [138]. Similarly, the structural integrity of a building may be compromised, and rubble may be unstable. Not only should robots be resilient to these hazards, but these hazards may themselves be the focus of sensing tasks, such as to provide situational awareness to responders

who wish to avoid hazards as well

- **Complex geometry:** Rubble creates complex formations and small and irregular voids. Because casualties are often buried deep within the rubble [138], sensing robots should be able navigate such voids or else be able to recognize voids and mark them for further inspection

- **Limited communication:** Disasters frequently destroy communication infrastructure, and wireless communication within a disaster site may be unreliable such as due to rubble [138]. Teams of robots may also employ ad-hoc networks [73], and the communication graphs linking robots may be incomplete or disconnected

- **Lack of information:** Emergency and disaster response activities typically occur within hours of an event [138], and teams often work independently and with limited access to information [137]. By extension and considering other constraints on communication, robot teams will also have limited prior information on the environments that they operate in

Disaster and emergency response scenarios do not always exhibit all of these challenges. However, first responders and search and rescue teams are individually responsible for a wide variety of disasters and emergencies. For this reason, any aerial sensing systems that are useful in practice will likely be those that are readily accessible to responders and which those responders have been trained to use. As such, in order to be useful, robots may have to address many challenges related to different emergencies and disaster scenarios [136, 138].

1.2 Approach and scope

This thesis focuses on the process of jointly planning sensing actions over short time horizons (e.g. seconds) and across large teams of robots, with a focus on sensing tasks arising from urban search and rescue. The proposed approach draws from sequential greedy algorithms for maximization of submodular functions [69, 140] (discussed in detail in Chapter 3) which are becoming increasingly popular for multi-robot sensor planning problems [7, 89, 102, 151, 161, 169, 203].

Lack of reliable communication between robots and damage to infrastructure also motivate our focus on distributed planning. Specifically, robots may not be able to communicate reliably with centralized or remote computational resources (e.g. a mobile base station or remote server) so robots should be able to plan for themselves. However, we note that some of the techniques we develop in this thesis could be adapted to produce efficient centralized and parallel implementations.

While some works that apply submodular maximization to related sensor planning problems allude to distributed implementation [7, 151], planning time for greedy techniques still scales at least linearly with the number of robots. Challenges related to distributed variations of these algorithms have also only begun to be addressed [75, 82, 83], and yet existing approaches still scale poorly in planning time or suboptimality as the number of robots increases. Conversely, this thesis proposes techniques for planning via submodular
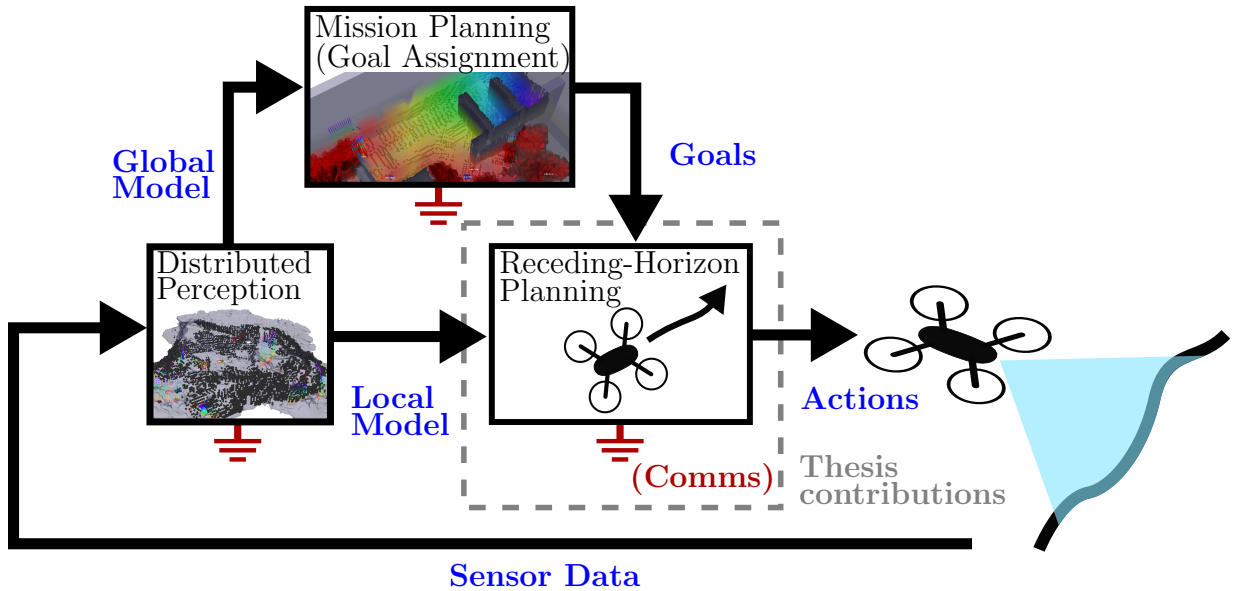
Figure 1.2: Diagram of a typical sensing system. This thesis focuses on planning sensing actions for teams of robots over a receding-horizon (e.g. several seconds). The inputs to the receding-horizon planner are long-term goals for the sensing process and models of the environment that are locally consistent in space and time. The outputs of the receding-horizon planner are control actions, typically a smooth trajectory, which the robot will execute to collect sensor data.

maximization that scale to any number of robots, and we prove that the techniques we propose maintain constant-factor suboptimality for a variety of sensing tasks.

Specifically, we develop a receding-horizon approach to sensor planning—illustrated in Fig. 1.2—whereby robots collectively optimize sensing actions over a short time horizon, typically several seconds. In general, such planners take high-level goals and locally consistent models of the environment as inputs and output control actions in the form of smooth trajectories. This thesis focuses primarily on the distributed aspect of these receding-horizon optimization problems. While we address some challenges related to receding-horizon planning for *individual robots*, that topic can also be addressed separately [76, 77]. We will also provide little detail on methods for high-level mission planning (or goal assignment) and distributed perception which are each necessary components of complete sensing systems and, themselves, important topics of study. Maintaining distributed representations of the environment can also incur significant communication costs. While there are ways to mitigate such communication costs either by intelligently communicating or compressing sensor data [57, 72] or by distributed robots in ways that reduce the need for communication [123], the methods in this thesis will not address this challenge. Likewise, problems related to estimating robot position and states, particularly simultaneous localization and mapping (SLAM), are out of scope both for passive estimation and as an active control problem [93]. Additionally, since our analysis will apply primarily to receding-horizon sub-problems, translating improvements in solution quality on these sub-problems into improvements in task performance—such as for robots to map an environment more quickly—will itself pose an important challenge in this thesis.

7

Further, this thesis will apply the methods for receding-horizon planning which we develop to two application areas: *exploration* (autonomous mapping) and *target tracking*. Here, exploration serves as a simple proxy for more complex tasks related to search and rescue. In particular, planning for exploration addresses the challenge of selecting sensing actions while operating in an environment with unknown geometry. On the other hand, the target tracking problems which we study are not immediately relevant to the search and rescue domain. Instead, these target tracking problems provide an opportunity to study a special class of factored objectives; for target tracking this arises in an objective which can be written as a sum over terms associated with each of the targets. More generally, this class of objectives can capture the multi-faceted sensing tasks that arise in search and rescue. In this sense, our analysis for such factored objectives could enable designers to extend the systems we develop for exploration to simultaneously address a variety of nuanced sensing tasks such as for locating survivors, assessing structural integrity, or identifying hazards.

1.3   Assumptions

The formulations of the exploration and target tracking tasks will also make a few simplifying assumptions. These assumptions are not necessarily realistic in practice but can often be relaxed with minor modifications to the approach. The following assumptions are important to various parts of this thesis:

- **Homogeneity:** We assume teams of robots are *homogeneous* to simplify exposition, but our methods could be applied to heterogeneous teams with only minor modifications

- **Spatial locality:** A stricter assumption is that robots' *incremental motions and sensor ranges are bounded*. However, we make this assumption primarily for the purpose of analysis of large teams of robots, and this assumption need only apply loosely to actual teams of robots. Additionally, spatial locality will arise in different forms for different tasks and will be treated on a case-by-base basis

- **Static environments:** For exploration, we assume that robots map *static environments*. This assumption that environments are largely static while being searched is likely realistic—searchers would generally avoid disturbing objects and rubble to avoid causing collapses and endangering themselves or any survivors. However, robots may also have to avoid collisions with responders who are moving about the environment or may have to recognize minor changes in the map such as after responders clear some rubble or open a door. Thus, while the environment may remain predominantly static for the purpose of the sensing task, addressing extraneous dynamics in and around the environment will be left to future work

1.4   Contributions and outline

This thesis provides two kinds of contributions: development of scalable, distributed algorithms with performance guarantees for multi-robot sensing and development of sensing

applications for exploration (autonomous mapping) and target tracking. Moreover, the central contribution of this thesis is a planning algorithm that guarantees constant factor sensing performance on average for a variety of sensing problems and requires only a fixed number of sequential planning steps for any number of robots. The rest of the thesis will involve implementing such algorithms, analysis of relevant sensing objectives, and developing methods for operating in and exploring unknown environments. These contributions are outlined below:

- **Planning for single- and multi- robot exploration:** Chapter 4 proposes a first attempt toward a distributed sensing algorithm for sensor planning via submodular maximization—by modifying to existing greedy methods [69]. The approach provides post-hoc guarantees and benefits from parallel computation when single-robot planning is expensive. We develop this approach, along with methods for single-robot planning based on Monte-Carlo tree search, to develop a receding-horizon planner for exploration with teams of aerial robots. And, *the rest of this thesis applies similar approaches to single-robot planning, throughout.* Additionally, we guarantee that robots do not collide with each other or the surrounding environment and describe some weak conditions that are sufficient to guarantee liveness. Notably, this is also the only chapter to include experiments with physical robots

- **Scalable multi-agent coverage:** Chapter 5 presents a greedy algorithm (Randomized Sequential Partitions or RSP) for coverage and any other submodular maximization problems which exhibit a certain higher-order monotonicity condition. Our algorithm guarantees constant factor suboptimality in the sensing objective (e.g. coverage) in expectation with fixed numbers of planning and communication rounds (versus one per robot for sequential greedy methods). This guarantee extends to any number of robots given certain conditions on spatial locality. *To our knowledge ours is the first algorithm to provide such guarantees for a non-trivial class of submodular functions.* As such, the remainder of the contributions continue to develop applications and implementations of RSP planning

- **Target tracking and quantifying inter-robot redundancy with factored objectives:** Chapter 6 introduces target tracking problems and applies similar methods for planning as the preceding chapters. Although we found in Chapter 5 that some sensing objectives satisfy a higher-order monotonicity condition that enables development of scalable planners with suboptimality guarantees, these results do not apply in general to some popular objectives (e.g. mutual information). However, we prove that some special cases of mutual information (those that can be factored as sums) satisfy similar suboptimality guarantees. We also provide detailed analysis for applying this result to target tracking problems. Further, this sum decomposition is relevant to the kinds of multi-objective problems which may arise when robots engage in more complex sensing tasks. This chapter also provides new analysis for approximate receding-horizon planning in real time that is relevant to sensing problems throughout this thesis. Last, simulation studies establish performance improvements for RSP planning in terms of target uncertainty and demonstrate that the suboptimality guarantees remain well-behaved for large numbers of robots (with results for

up to 96 robots, providing a 24× reduction in the number of planning steps compared to planning sequentially)

- **Time-sensitive sensing in unknown environments:** Chapter 7 revisits the problem of multi-robot exploration. We provide more in-depth analysis of objective functions and present one case of a mutual information objective that satisfies the monotonicity conditions we study in this thesis. This chapter also incorporates some changes in planner design that significantly improve on what Chapter 4 presents. Regarding simulation results, we find that improving completion times via sequential or RSP planning (versus having no explicit coordination in planning) can be challenging. Still, we demonstrate that planning with RSP improves coverage rates early in simulation trials and reliably improves solution quality (suboptimality)

- **Design and implementation of a distributed sensor planner:** While the previous chapters *described* distributed algorithms, Chapter 8 is the first to provide a distributed implementation of an RSP planner. Toward this end, we provide detailed discussion of design decisions and results that investigate communication costs. Simulation results demonstrate this distributed, synchronous, anytime planner in exploration of an office-like environment, verify that the system behaves as expected, and identify a 5% improvement in task completion time for coordination via RSP, albeit for a small set of experiments

1.4.1 Code release

Some of the source code for these contributions is available via the BSD license. The numerical experiments for coverage in Chapter 5, the target tracking simulations in Chapter 6, and the communication study in Chapter 8 rely on a common code-base,[2] written in Julia. We have also released the distributed implementation of RSP[3] that Chapter 8 describes which has been implemented in C++ via ROS. This release encompasses the distributed scheduling and communication aspects of RSP and excludes methods for exploration. We hope that the design of this package can enable incorporation of RSP methods alongside various single-robot receding-horizon sensor planners with minimal effort.

[2]https://github.com/mcorah/MultiAgentSensing
[3]https://github.com/mcorah/distributed_randomized_sequential_partitions

Chapter 2

Sensing Problems

Before developing the contributions of this thesis, let us pause to discuss the nature of the problems that we study. So far, the introduction (Chapter 1) has discussed sensing problems at a high level and from the perspective of applications, particularly search and rescue. This chapter will focus more closely on problem form and evaluation for time-sensitive sensing and for target tracking. Later, Chapter 3 will provide technical background methods and detailed discussion of related works.

We provide one caveat: the discussion in this chapter is primarily for the purpose of empirical evaluation of the systems we propose and to broadly characterize how features of sensing problems relate to the requirements of potential solution methods. Where some sensing processes admit algorithms with formal guarantees [79], features such as robot motion confound existing methods. Likewise, the receding-horizon techniques and other methods that we develop involve approximations and heuristics, and we would not expect the resulting systems to satisfy rigorous suboptimality guarantees.

2.1 Time-sensitive sensing

The *time-sensitive sensing* moniker intends to provide some formalism and structure to the problems we study in this thesis. Chapter 1 provided motivation for why completing sensing tasks quickly can be important. However, designers might consider other criteria. For example, even early methods for exploration based on frontiers [198] could be described as *complete* in the sense that the system will eventually explore the entire environment under reasonable assumptions.

In defining the time-sensitive sensing problem, we take a cue from Golovin and Krause [79] and describe the controller using notation for policies. However, in line with the rest of this thesis and to provide a direct description of the system, we also draw on notation from control theory.

Consider an *unknown* static environment $E \sim \mathcal{E}$ drawn from some *known* distribution over possible environments $\mathcal{E}$. One or more robots (this problem definition can be interpreted as describing arbitrary systems, consisting of any number of robots) navigate the environment and obtain observations according to the known dynamics and observation

model

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, u_{t-1}) \qquad\qquad \mathbf{y}_t = h(\mathbf{x}_t, E) \qquad (2.1)$$

where $\mathbf{x}_t$, $\mathbf{y}_t$, and u_t are the state, observation, and control input at the (discrete) time t. Like the environment, the initial state $\mathbf{x}_0 \sim \mathcal{X}_0$ is random and drawn from a known distribution. Given that the robot(s) learn about the environment through the sequence of observations, the controller is a policy:[1]

$$u_t = \pi(\mathbf{X}_{0:t-1}, \mathbf{Y}_{0:t-1}) \qquad (2.2)$$

where the capital letters represent vectors of states and observations. This policy states that the robot makes decisions at each time t given the available information, the states, and observations. Further, as the robot navigates the environment, it must also avoid collisions or otherwise unsafe states. In general, the robot must remain in some set of safe states $\mathcal{X}_{\text{safe}}(E)$ at all times. We seek to minimize the expected completion time[2] (given the distribution over environments and starting positions) for a given sensing task—say mapping a building. Progress in the task is measured by the sensing quality function $J(\mathbf{X}_{0:t}, \mathbf{Y}_{0:t})$ which depends on states and observations while completion is modeled by a quota $B(E)$ which depends on the environment. For example, this sensing quality may represent how much of a building that a robot has observed or mapped by a given time, and the quota may require the robot to map a certain fraction of the building. Putting this all together, the time-sensitive sensing problem is as follows:

$$
\begin{aligned}
\min_{\pi} \quad & \mathbb{E}_{E,\mathbf{x}_0}[T] \\
\text{s.t.} \quad & J(\mathbf{X}_{0:T}, \mathbf{Y}_{0:T}) \geq B(E) \\
& \mathbf{x}_t \in \mathcal{X}_{\text{safe}}(E) \\
& u_t = \pi(\mathbf{X}_{0:t-1}, \mathbf{Y}_{0:t-1}) \\
& \mathbf{x}_t = f(\mathbf{x}_{t-1}, u_{t-1}) \\
& \mathbf{y}_t = h(\mathbf{x}_t, E) \\
& \text{and the above for all } t \text{ in } \{1 \dots T\}
\end{aligned}
\qquad (2.3)
$$

Solving this problem, even approximately, is challenging. This problem is imbued with challenges from several different fields such as control theory, artificial intelligence, and combinatorial optimization as we hint by mixing notation from these fields.

2.1.1 Characteristics of time-sensitive sensing

The following paragraphs discuss some of the challenges and properties of time-sensitive sensing problems.

---

[1]For now, we write the policy and system model as if deterministic. However, the reasoning we present also applies to stochastic systems and policies, as we study elsewhere in this thesis.

[2]Similar theoretic works [41, 79, 169] typically use a more abstract monotonic or modular cost which could represent quantities like time or energy. Although planning with energy costs is an important topic of study [173, 179], such general cost models do not arise in this thesis.

2.1.1a Inference and uncertainty

The robot gains information about the environment by collecting observations from a variety of states. In particular, the set of possible environments that are consistent with a collection of states $\mathbf{X}_{0:t}$ and $\mathbf{Y}_{0:t}$ is[3]

$$\{E : \mathbf{Y}_{0:t} = h(\mathbf{X}_{0:t}, E) \text{ for all possible environments } E\}. \qquad (2.4)$$

Referring back to (2.3), several features of the problem formulation depend on the environment: the observations $h(\mathbf{x}, E)$, the safe set $\mathcal{X}_{\text{safe}}(E)$, and the quota $B(E)$. Effects of uncertainty in observations will naturally arise frequently as this thesis focuses on sensing problems. On the other hand, uncertainty in the quota could be interpreted as complicating certain solution strategies, but the quota will only appear again in experimental evaluation where its value will be known.

Although, we will discuss the safe set less frequently, the uncertainty in which actions are safe has strong implications on system performance such as by bounding maximum speeds [76]. Likewise, this uncertainty implies that we cannot plan complete paths (such as to map a building) a priori because those paths may not be safe. Thus, methods for path planning cannot solve (2.3) on their own. We will continue this discussion later in this section in terms of *safety* and *feasibility*.

2.1.1b Form of the sensing quality

The sensing quality $J(\mathbf{X}_{0:t}, \mathbf{Y}_{0:t})$ is immediately in terms of known quantities: states and observations. However, the sensing quality is also a function of the states and environment

$$J(\mathbf{X}_{0:t}, \mathbf{Y}_{0:t}) = J(\mathbf{X}_{0:t}, h(\mathbf{X}_{0:t}, E)), \qquad (2.5)$$

wherein we abuse notation by applying the observation function to a vector of states. Given this latter form, we can make inferences about future observations. In this sense, future sensing progress is uncertain to the extent that the environment remains uncertain but can also be optimized by selecting robot motions.

Numerous recent works seek to optimize common measures of future sensing progress. However, common sensing functions (such as those we discuss in this thesis) have sometimes surprising differences in important functional properties. For this reason, we do not yet specify the functional form of the sensing quality J. For example, Golovin and Krause [79] demonstrate that certain sensor coverage functions have a monotonicity property called *adaptive submodularity* even though important measures of information gain do not [45, 80]. We will frequently revisit similar functions for coverage and information gain. As these often have similar properties, distinctions such as this are important. Here, adaptive submodularity is one factor that determines whether greedy methods can solve (2.3) to

---

[3]We could write this more generally for noisy observations using Bayes' rule. However, some relevant solution methods [96] only apply to deterministic models. We emphasize the deterministic case to avoid unduly restricting applicable solutions and because the sensing noise for common depth sensors in mapping [87] is frequently small compared to range (meters) and the environment discretization (typically 10 cm).

near-optimality [79]. However, we will also find that adaptive submodularity does not apply to many of the problems in this thesis.

2.1.1c Robot dynamics and available observations

The principal effect of the dynamics model is that it changes which information gathering actions are available at a given time. Here, the information available at a given state consists of state, observation pairs $(\mathbf{x}, \mathbf{y})$. Robots gain information about the environment by visiting states and inspecting such pairs as $\mathbf{y} = h(\mathbf{x}, E)$. The information gathering actions that the robot has available at a state $\mathbf{x}$ consists of the set of states the robot can visit next

$$\{\mathbf{x}' : \mathbf{x}' = f(\mathbf{x}, u) \text{ for all control inputs } u\} \tag{2.6}$$

As such, the set of available observations at any given time depend on the prior selections. This excludes some methods for adaptive sensing [47, 79] which require the set of information gathering actions not to change over time.

Another important property of the system dynamics f is whether it produces a directed or undirected relationship between states. The system dynamics *induce an undirected relationship* on the system states if, for any possible state $\mathbf{x}_1$ and control u_1 pair

$$\mathbf{x}_2 = f(\mathbf{x}_1, u_1)$$

there exists some u_2 such that

$$\mathbf{x}_1 = f(\mathbf{x}_2, u_2). \tag{2.7}$$

Alternatively, the dynamics produce a *directed* relationship if the above does not hold. Which of these properties holds can determine which kinds of algorithms and analysis [41, 43] apply to the planning problems we discuss in this thesis.

Many of the systems in this thesis will involve trivialized kinematic models with undirected dynamics. However, Chapters 4 and 8 each provide results for systems with more general, directed dynamics models. Likewise, the methods for distributed coordination which we develop are independent of the robots' dynamics.

2.1.1d Stopping and inevitability of collision

Let us consider how dynamics affect safety. Often, we will consider problems where robots can always stop immediately. In terms of control theory, this means that for all possible states $\mathbf{x}_t$ there exists some control input u_t that stops the robot so that

$$\mathbf{x}_{t+1} = \mathbf{x}_t = f(\mathbf{x}_t, u_t) \quad \text{(for some } u_t\text{)}, \tag{2.8}$$

ensuring that the robot remains in the same state. Individual states where (2.8) holds are called *invariant states*[4] [150]. Not all states are invariant states in practice—consider

---

[4]Similar properties also apply for invariant *sets* where each state in the set has some control action which can keep the robot in the invariant set.

14

a robot with non-zero velocity and non-trivial dynamics. Now, consider a robot flying rapidly toward a brick wall; that robot will eventually reach a point of no return where it cannot stop without hitting the wall—such is the heightening intensity of two children playing "chicken" who must decide when to act to avoid collision. States past such points of no return, where no sequence of control inputs can prevent the robot from leaving the safe set $\mathcal{X}_{\text{safe}}(E)$, are *inevitable collision states*. Controllers that ensure safety by preventing robots from entering inevitable collision states are a frequent topic of study in contemporary control theory in the form of control barrier functions [2, 142, 147] and also for applications to mobile robots [77, 95, 120, 121, 192]. This will arise in Chapter 4 where we will maintain safe stopping actions for all robots at all times. Further, some works have begun to investigate the relationship between safety and inference in mapping [95]. For example, a robot can take a turn widely while accelerating around a corner to obtain advance warning of what lies beyond. Now, building on these questions about safety, we can also ask whether a robot will safely complete a sensing task—whether (2.3) is feasible.

2.1.1e Feasibility, failure, and completeness

Our definition of time-sensitive sensing (2.3) sets a high bar for producing even just an approximate solution. A feasible policy must:

- Complete the sensing task (or else the completion time is infinite),
- Guarantee that the robot will remain in the safe set,
- And must do so for all of the environments that remain consistent with prior observations (2.4).

Given an instance of (2.3), we may ask several relevant questions about feasibility:

- Does a feasible solution exist?
- Given a policy, does that policy constitute a feasible solution? (A policy that is feasible and thereby always eventually finishes the sensing task is said to be *complete*)
- Does any of the above hold for an individual environment and starting position pairing or a restricted collection of such pairs?
- Does any of the above hold for a given partial history $(\mathbf{X}_{0:t}, \mathbf{Y}_{0:t})$?
- Is the converse true for any of the above? (e.g. Is there no feasible solution?)

We will answer a small subset of this class of questions. Questions about safety already answer parts of the converse forms (e.g. Is failure imminent? Can a given policy violate the safety constraint?). Classical frontier-based exploration [198] approaches are also complete given reasonable assumptions: The robot will navigate to and observe unknown space until there is nothing left to observe. We will then attempt to replicate aspects of such classical approaches when considering more nuanced problems involving dynamics, unstructured environments, and cameras with limited fields of view.

2.1.1f   Decision processes and adaptivity

Time-sensitive sensing problems form *decision processes* by nature of control via policies $\pi(\mathbf{X}_{0:t}, \mathbf{Y}_{0:t})$, functions of prior decisions and observations, rather than fixed sequences of control actions. However, does (2.3) readily fit the form of a classical Partially Observable Markov Decision Process (POMDP) [104]? At this point, it is important to realize that the reward (or, more accurately, cost) in (2.3) is the completion time. Although time is easy to compute, completion depends on the sensing quality $J(\mathbf{X}_{0:t}, \mathbf{Y}_{0:t})$ which is a function of the sequence of states. Adapting these problems to the form of a traditional POMDP then requires an exponential number of states for the same reasons as described by Golovin and Krause [79] regarding the *adaptive submodular maximization* problems which they define. Alternatively, certain POMDP variants allow for more complex rewards which we discuss in Sec. 3.3.3

2.1.2   The receding-horizon approximation

Because solving full time-sensitive sensing problems is challenging, this thesis takes the approach of solving a simpler sub-problem. Given prior states and actions $\mathbf{X}_{0:t}$ and $\mathbf{Y}_{0:t}$ we seek to maximize the sensing quality over the next L steps for a fixed sequence of future control actions $u_{1:L}$ (known as the *receding-horizon*) and adopt the policy:

$$
\begin{aligned}
\pi(\mathbf{X}_{0:t}, \mathbf{Y}_{0:t}) = \underset{u_{1:L}}{\arg\max} \ & \mathbb{E}_E[J(\mathbf{X}_{0:t+L}, \mathbf{Y}_{0:t+L})] \\
\text{s.t. } & \mathbf{x}_{t+l} \in \mathcal{X}_{\text{safe}}(E) \\
& \mathbf{x}_{t+l} = f(\mathbf{x}_t, u_l) \\
& \mathbf{y}_{t+l} = h(\mathbf{x}_{t+l}, E) \\
& \text{for all } l \text{ in } \{1 \dots L\}
\end{aligned}
\tag{2.9}
$$

Receding-horizon methods such as this are common in robotics and control [150] and for sensing problems like those we discuss in this thesis [20, 36, 116, 160]. We adopt this approach as a heuristic and would not expect to obtain approximation guarantees. However, we will be able to guarantee certain properties of the resulting system such as safe operation. Furthermore, this sub-problem does not involve adaptation and therefore no longer requires specialized analysis for adaptation [79]. This feature can then enable application of informative path planning techniques for non-adaptive problems.

2.1.2a   Informative path planning

The solution to (2.9) corresponds to a path (as $\mathbf{x}_{t:t+L}$ is a function of $u_{1:L}$) instead of a policy. This path may be directed or undirected as discussed earlier. Assuming that $\mathbb{E}_E[J(\mathbf{X}_{0:t+L}, \mathbf{Y}_{0:t+L})]$ satisfies certain properties, existing methods for informative path planning can solve (2.9) with various guarantees [41, 43, 90, 169, 200]. Much of this thesis will focus on how to apply those and similar methods to large instance of (2.9) where the states and dynamics represent *teams of robots*.

2.1.3 Exploration

In this thesis, *exploration* refers to the process of mapping some environment. This section presents a simplified but also useful model of the exploration process. To begin, we model the environment as a sequence of n_{m} cells $E = [c_1, \ldots, c_{n_{\mathrm{m}}}]$ which respectively represent free and occupied space $c_i \in \{0, 1\}$. This environment is drawn from some probability distribution, and we will make no assumptions about that distribution at this point.

   While navigating this environment the robot must avoid occupied cells while observing cell occupancy values with a camera or other sensor. For some state $\mathbf{x}$, the camera function $F^{\mathrm{cam}}(\mathbf{x}, E) \subseteq \{1, \ldots, n_{\mathrm{m}}\}$ returns the set of cells which the robot observes from that state and given the specific environment. This camera function may then refer an arbitrary projective camera model.[5] Then, the observation function indicates which cells the robot observes along with their values $h(\mathbf{x}, E) = \{(i, c_i) : i \in F^{\mathrm{cam}}(\mathbf{x}, E)\}$. Similarly, the occupancy function $F^{\mathrm{occ}}(\mathbf{x}) \subseteq \{1, \ldots, n_{\mathrm{m}}\}$ determines which cells that the robot occupies. A robot is in the safe set if all the cells it occupies are free $\mathcal{X}_{\mathrm{safe}}(E) = \{\mathbf{x} : c_i = 0 \text{ for all } i \in F^{\mathrm{occ}}(\mathbf{x})\}$. We will reward the robot for the number of cells that it observes. So, the sensing quality in exploration for given states $\mathbf{X}_{0:t}$ and observations $\mathbf{Y}_{0:t}$ is

$$J(\mathbf{X}_{0:t}, \mathbf{Y}_{0:t}) = \left| \bigcup_{i=0}^{t} F^{\mathrm{cam}}(\mathbf{x}_i, E) \right|. \tag{2.10}$$

When written as a function of a set (2.10) is a kind of coverage objective and has useful monotonicity properties (Chapter 3 discusses both coverage and monotonicity in detail). Additionally, the expectation, which appears in the receding-horizon problem (2.9), retains similar properties. Yet, unlike some coverage objectives [79], (2.10) is not necessarily adaptive submodular (see Appendix A.1).

2.2 Time-average tracking

Chapter 6 deviates from the rest of this thesis by focusing on sensor planning for observation of *dynamic systems*. Unlike exploration where mobile (dynamic) robots seek to completely map a static environment, (target) tracking problems are ongoing and can be evaluated based on average performance. In this case, we can consider tracking a target whose state $\mathbf{x}^{\mathrm{t}}$ evolves according to a Markov process so that $\mathbf{x}_t^{\mathrm{t}} \sim \mathbb{P}(\mathbf{x}_t^{\mathrm{t}} | \mathbf{x}_{t-1}^{\mathrm{t}})$, (here, the super-script "t" refers to target states) although more general processes are also relevant to this discussion. In this case, the sensing quality $J(\mathbf{X}_{0:t}, \mathbf{Y}_{0:t})$ might correspond to the number of times that the robot has observed a target or a measure of the uncertainty in the target's entire

---

[5]An appropriate sanity condition for F^{cam} would be to require the set of cells which the robot observes $A = F^{\mathrm{cam}}(\mathbf{x}, E)$ to depend only on the state and the values of the cells that the robot observes. Consider the occupancy values of those cells E_A. We can require for all environments E' that whenever the values of the observed cells are the same $E_A' = E_A$, then set of cells that the robot observes $A = F^{\mathrm{cam}}(\mathbf{x}, E')$ must also be the same. This avoids inadvertently providing information about cells outside of the field of view.

trajectory $\mathbf{X}_{0:t}^{\mathrm{t}}$. Adapting (2.3) as appropriate, the time-average performance is

$$\max_{\pi} \; \lim_{t\to\infty} \frac{\mathbb{E}[J(\mathbf{X}_{0:t}, \mathbf{Y}_{0:t})]}{t}, \tag{2.11}$$

given some observation model and subject to robot dynamics. Then, given the focus on time-average performance, the tracking problems we study (unlike exploration) are not time-sensitive sensing problems as described in Sec. 2.1.

We will now discuss these problems briefly because they are of secondary importance to this thesis and because many of the same properties and challenges apply.

2.2.1 Long duration Markov processes

The target tracking problem (2.11), as described, is a kind of long duration Markov Decision Process (MDP) [4], and if the target states $\mathbf{X}^{\mathrm{t}}$ are not known exactly, such tracking problems are also *partially observable* as for time-sensitive sensing problems. Here, the key feature of the problem is that (2.11) focuses on average performance for all time and does not favor near-term gains. Although some solution methods exist for these problems, the problems that we study also incorporate features that make these problems difficult to solve such as very large state and action spaces.

2.2.2 Ergodicity

One important property in such problems is *ergodicity*. For an ergodic system, the distribution of states for a single trajectory, taken over a very long duration, is the same as the distribution for many trajectories. In short, ergodicity states that no transitions in a system have irrevocable effects. This could refer to termination conditions or a system that falls off a ledge and cannot get back up again.

Long duration problems are typically studied under various sorts of ergodicity conditions [4]. In this sense, ergodicity conditions determine whether (2.11) describes rewards for an individual trajectory or whether rewards could vary greatly across even very long trials. Unsurprisingly, ergodicity has also been used in heuristics to solve similar problems [8, 127, 129].

2.2.3 Information dynamics

We briefly note that concepts from information theory and information dynamics are highly relevant to target tracking problems and will go into more detail on related concepts later in this text [23, 58]. Information theory can describe how quickly uncertainty in target states can grow [58] and can provide further connections to ergodicity or characterize interactions between system components [23].

2.2.4 Stability

Stability properties are also relevant to tracking problems. In this case stability can describe whether robots may travel far from the targets that they are tracking or whether

uncertainty can grow indefinitely [156]. Such conditions will generally produce worst case rewards in (2.11) (such as zero or infinitely negative reward). Although we will not seek to characterize stability in the systems we study, this property characterizes the kind of behavior that could arise in results.

2.3   Ramifications and challenges of multi-robot sensing

In an abstract sense, the discussion so far applies to both individual robots and multi-robot teams when viewed as a collective. Yet, this chapter has not explicitly addressed the ramifications of multi-robot sensing. To begin, the following chapters will describe various methods to decompose and plan for multi-robot sensing problems. These methods are all much more efficient than applying planners for individual robots to the joint state space of the entire team.

Additional challenges and constraints also arise when considering multi-robot teams. For example, safety constraints (Sec. 2.1.1d) can be amended to account for inter-robot collisions. We also encounter new challenges when considering distributed planning (via processors onboard each robot) and accounting for communication between robots. Regarding distributed planning, robots should be able to obtain solutions in a timely manner, preferably while taking advantage of parallel computation. Likewise, a distributed planner may require systems for sharing sensor data or maintaining consistent beliefs via communication between robots. This, in turn, leads to challenges with respect to communication as robots may have to maintain connectivity across the team or respect bandwidth constraints. We will begin to explore these challenges next with the background and related work (Chapter 3), and later Chapter 8 will address challenges related to communication more directly.

Chapter 3

Background and Related Work

This chapter will establish the theoretic and practical foundations for this thesis and for sensing and information gathering in unknown environments. We will begin with a high-level discussion of sensor planning, exploration, and navigation in unknown environments (Sec. 3.1). This section builds somewhat on the challenges for sensing problems (Chapter 2) and introduces related works in this area. The foundations for questions about information and uncertainty in robotics come from information theory (Sec. 3.2). More directly, there is a wide body of literature related to robotics that applies information theory to planning and control for sensing and information acquisition (Sec. 3.3). And for the purpose of this thesis, we are particularly interested in applying techniques for sensing and information gathering to multi-robot settings (Sec. 3.4). Many sensing objectives, especially those based on information theory, share useful monotonicity properties. One of the best known monotonicity properties is submodularity, and many single- and multi-robot planning problems can be solved efficiently and near-optimally using techniques for submodular maximization (Sec. 3.5). In particular, we are interested in applying methods for submodular maximization to distributed planning for multi-robot teams. There are already a number of works on parallel and distributed submodular maximization (Sec. 3.6). However, not all of these works are readily applicable to multi-robot settings. Further, no existing works are able to scale to large numbers of robots due to increasing planning time. Therefore, in addition to developing methods for aerial autonomy and exploration, this thesis also presents new methods for submodular maximization that can scale to arbitrary numbers of robots.

3.1 Autonomy and exploration in unknown environments

While sensing goals and objectives can vary significantly, we expect urban emergency and disaster response tasks to feature cluttered and at least partially unknown environments and refer to scenarios where mapping is the primary sensing task as exploration problems. We may then cast navigation and mapping in unknown environments as a sensing problem [37, 103] and plan to maximize some notion of information gain. However, operation

in unknown environments is distinguished by several unique challenges:

- Robots must avoid collision with objects and each other;

- The navigable subset of the environment is revealed incrementally while the robot observes free and occupied space;

- Robot states and positions are uncertain; and

- Environments may be large and informative observations distant.

Further, the non-trivial dynamics of aerial robots exacerbate these challenges. The rest of this section will address information-based formulations and each of these challenges in turn.

3.1.1   Safe navigation

Safety reduces to planning collision-free paths for slow-moving ground robots with large fields of view [116, 172, 198], and authors frequently overlook challenges related to safety constraints. On the contrary, the aerial robots that we study can move quickly and cannot stop instantaneously. These same robots can also move side-to-side, outside of the camera field of view, and may not be able observe what they are moving toward. Constraints on reachability that ensure set-invariance [150] can ensure that robots always avoid collisions with the environment and each other or can ensure that robots can return to a starting position [71]. Safety requirements also constrain how quickly robots move toward [76, 77] and plan through [95] unmapped space. As such, learning to avoid collisions [152] can also improve speeds to the extent that the learner can generalize about the environment.

3.1.2   State uncertainty

Thus far, we have emphasized observations of the state of the environment rather than the states of the robots themselves. As studied in robotics, uncertainty in the states of robots leads to simultaneous localization and mapping (SLAM) problems and some additional challenges: estimates of the robot state can drift dangerously over time, and uncertainty in the state history contributes to uncertainty in the map. Direct observation of position by GPS or motion capture can mitigate or eliminate these challenges. Except, GPS access can also be limited when operating in urban environments [137] which motivates development of reliable SLAM systems for the problems we study.

Robots can also plan to minimize uncertainty via information-theoretic methods for sensor planning [7, 93, 172] forming methods for what is known as active SLAM. While classical SLAM algorithms that apply particle filters to two-dimensional SLAM readily admit joint optimization to minimize uncertainty in the map and robot state [172], methods for sensing with modern smoothing-based frameworks [7, 93] are somewhat more abstract and often represent uncertainty in environment geometry indirectly. Additionally, some works in this area employ the same greedy algorithms for submodular maximization [7, 93] which we study in this thesis. However, submodularity generally only applies to robots' histories of states and observations but not to future states, due to dependence on the robots' own control actions [7].

As such, even though this thesis does not address state uncertainty explicitly, methods similar to those we propose could play a role in systems for active SLAM. Alternatively, designers may also wish to address uncertainty in future states through constraints on uncertainty as a matter of safety, and such constraints could be incorporated into the problem formulations we propose.

3.1.3 Information-based exploration

The mobility of aerial robots exploring some environment contrasted against limitations on field of view motivate methods that can predict information gain at different view points such as with mutual information [37, 103, 201] and coverage-like [29, 62] objectives. Authors have also applied a variety of such objectives along with a numerous techniques for planning paths and actions [20, 36, 116].

We will apply methods based on motion primitives [139, 179] and Monte-Carlo tree search [39, 116], and, more recently, our work has addressed motion primitive design in this framework [76, 77]. Additionally, Chapter 7 will revisit objective design and draw connections between a number of relevant objectives for robotic exploration.

3.1.4 Frontiers and the distribution of information

Some of the earliest works on exploration [198] emphasize the geometry of the environment: observations of unknown space by a robot in known free space all pass through the *boundary between the known free space and unknown space* which is called the *frontier*. Providing the weak assumption that robots are always able to observe nearby frontiers, navigating toward the nearest frontier can serve as a simple and reliable technique for exploring an environment to completion. Although frontiers can be useful for selecting local control actions [50, 62], they also model the distribution of information sources in exploration tasks: after mapping one part of an environment, a robot may traverse a significant distance to reach the next nearest frontier. In this sense, we can generalize frontiers as sets of sufficiently informative points in the robot state space [57], and robots can alternate between maximizing information gain locally and navigating to new information-rich regions of the environment. Moreover, in either case, designers may consider routing robots between information-rich regions or frontiers [111] to reduce travel distance or completion time.

3.1.5 Multi-robot exploration

Most works on multi-robot exploration focus on assignment of frontiers [29, 30] and environment regions [170] or maintaining maps via communication [40, 57]. Although authors do not typically apply arguments based on submodularity in this domain, greedy assignment mechanisms are common [29, 30] but at the level of task assignment rather than receding-horizon planning as in our approach (which we would expect to run at faster rates than global assignment processes).

As such, designers seeking to map an environment quickly may deploy large numbers of robots that interact on short time scales, and this thesis addresses the concomitant challenges via coordination over finite horizons. Our approach then complements works on exploration that focus on larger spatial and temporal scales [29, 30, 134], and such methods may run in outer loops or on centralized nodes to supplement the distributed planners we propose (as in Fig. 1.2).

3.1.6 Connectivity and communication

Exploration scenarios, such as for search and rescue [138] have the potential to impair communication between robots via damage to infrastructure and harsh environments (e.g. due to stone and concrete). Teams of robots may then jointly plan to explore the environment and to maintain or regain communication links to communicate data to the operator and each other [13, 14, 181]. Such, methods for maintaining communication are diverse: Banfi et al. [14] allow robots to become disconnected but enforce recurrent connectivity constraints to enable periodic communication; Tatum [181] proposes a method where robots jointly explore and place communication nodes with emphasis on subterranean environments; or designers may simply require the team to maintain connectivity at all times [128, 187].

This thesis does not address connectivity directly. However, the planners we propose are robust to loss of communication, and our approach could reasonably be extended by applying any of the cited connectivity-aware methods as part of a high-level planning component.

3.2 Information theory

Having discussed a few sensing problems in robotics, let us delve into the foundations for quantifying uncertainty in states and information gained via sensing actions. To begin, entropy quantifies the uncertainty in a random variable X—such as the position of a robot or a target—with the average number of bits necessary to encode a realization of that variable and is denoted as

$$\mathbb{H}(X) = \sum_i -\mathbb{P}(X = i) \log_2 \mathbb{P}(X = i). \tag{3.1}$$

The joint entropy then expresses the combined uncertainty of two (or more) variables X and Y and simply involves another sum:

$$\mathbb{H}(X, Y) = \sum_i \sum_j -\mathbb{P}(X = i, Y = j) \log_2 \mathbb{P}(X = i, Y = j). \tag{3.2}$$

Then, conditional entropy encodes the expected uncertainty in X given that Y will be observed:

$$\mathbb{H}(X|Y) = \sum_i \sum_j -\mathbb{P}(X = i, Y = j) \log_2 \mathbb{P}(X = i|Y = j). \tag{3.3}$$

This can be written as an expectation over possible outcomes for Y:

$$= \sum_i \sum_j -\mathbb{P}(Y = j)\mathbb{P}(X = i|Y = j) \log_2 \mathbb{P}(X = i|Y = j)$$

$$= \mathbb{E}_{j \sim Y}[\mathbb{H}(X|Y = j)] \tag{3.4}$$

or in terms of the joint entropy:

$$= \mathbb{H}(X, Y) - \mathbb{H}(Y).$$

The goal of exploration and other sensor planning problems is often to reduce uncertainty and therefore entropy of a target variable (say X) by obtaining observations Y.

Mutual information quantifies that expected reduction of the entropy of X given an observation Y

$$\mathbb{I}(X; Y) = \sum_i \sum_j -\mathbb{P}(X = i, Y = j) \log_2 \frac{\mathbb{P}(X = i)\mathbb{P}(Y = j)}{\mathbb{P}(X = i, Y = j)}, \tag{3.5}$$

and can be written in terms of entropies

$$= \mathbb{H}(X) - \mathbb{H}(X|Y) = \mathbb{H}(X) + \mathbb{H}(Y) - \mathbb{H}(X, Y). \tag{3.6}$$

In some cases, we will also apply the conditional mutual information which simply involves modifying the definition to include conditioning on the given variable

$$\mathbb{I}(X; Y|Z) = \mathbb{H}(X|Z) - \mathbb{H}(X|Y, Z) \tag{3.7}$$
$$= -\mathbb{H}(Z) + \mathbb{H}(X, Z) + \mathbb{H}(Y, Z) - \mathbb{H}(X, Y, Z).$$

Please refer to Cover and Thomas [58] for further detail on information theory and the properties of entropy and mutual information and in reference to further discussion in this section, except when otherwise noted.

3.2.1 Continuous variables

The discussion at the beginning of this section focuses on information theory for the simple case of discrete variables. Analogous expressions exist for continuous variables and probability densities. These collectively refer to differential information. For example, the differential entropy can be written as

$$\mathbb{h}(X) = \int -p(X = x) \log_2 p(X = x)\, \mathrm{d}x \tag{3.8}$$

where X is a continuous variable, and $p(X = x)$ is its probability density.

The expression for (3.8) is nearly identical to the original expression for entropy in (3.1) aside from replacing the sum with an integral. Likewise, differential information satisfies many of the same properties as the discrete form so that the two are frequently

interchangeable. In particular, this thesis will make use of monotonicity properties such as submodularity which are discussed later in this chapter; both forms of information satisfy these properties equally. As such, this thesis will not typically disambiguate differential and discrete information, except in cases where the two behave differently. One such difference is that differential entropy may be negative so that $\mathbb{h}(X) \not\geq 0$. By extension, $\mathbb{I}(X;Y) = \mathbb{h}(X) - h(X|Y) \not\leq \mathbb{h}(X)$ so that mutual information is also not upper-bounded by differential entropy. Fortunately, even though differential entropy may be negative, mutual information between continuous variables is always non-negative.

3.2.2 Properties of mutual information

Information theoretic expressions exhibit a number of intuitive properties from which this theory gains much of its elegance and generality.

Entropy and mutual information satisfy *chain rules* [58, Theorem 2.5.1–2]. For entropy

$$\mathbb{H}(X_1, \ldots, X_n) = \mathbb{H}(X_n | X_1, \ldots, X_{n-1}) + \mathbb{H}(X_1, \ldots, X_{n-1}), \tag{3.9}$$

and for mutual information

$$\mathbb{I}(X; Y_1, \ldots, Y_n) = \mathbb{I}(X; Y_n | Y_1, \ldots, Y_{n-1}) + \mathbb{I}(X; Y_1, \ldots, Y_{n-1}). \tag{3.10}$$

These equations allow us to write both kinds of quantities in terms of sums of marginal gains (the conditional term in each expression). We will later find that we are able to take advantage of the behaviors of these marginal gains when optimizing information-theoretic quantities.

Entropy also decreases under conditioning so that $\mathbb{H}(X|Y) \leq \mathbb{H}(X)$ which expresses that new information cannot increase the uncertainty of a random variable. If we write the mutual information in terms of entropies, we can rearrange this inequality to prove that the mutual information between two variables is non-negative $\mathbb{I}(X;Y) = \mathbb{H}(X) - \mathbb{H}(X|Y) \geq 0$.

Also, independent variables do not carry information about each other. If X and Y are independent $\mathbb{H}(X) = \mathbb{H}(X|Y)$ and therefore $\mathbb{I}(X;Y) = 0$. Similar statements hold for conditioning when variables are conditionally independent. Additionally, these properties can be used to prove that common cases of mutual information are submodular, and we will come back to that later.

Having an upper bound on mutual information is often useful such as when we would like to establish when a sensing task is almost complete. For *discrete* random variables $\mathbb{H}(X) \geq 0$ is non-negative (with or without conditioning). As a result of non-negativity, mutual information for discrete variables can be upper-bounded by entropy as

$$\mathbb{I}(X;Y) = \mathbb{H}(X) - \mathbb{H}(X|Y) \leq \mathbb{H}(X). \tag{3.11}$$

However, entropy for continuous variables may be negative because, with enough observations, we may learn the value of a continuous variable with infinite precision and thereby gain infinite information. However, if observations share a noisy channel, mutual information between the observation and target variable is bounded by the mutual information

between the target variable and the output of the channel. Specifically, the *Markov inequality* establishes that if $X \to Y \to Z^1$

$$\mathbb{I}(X;Y) \leq \mathbb{I}(X;Z). \tag{3.12}$$

This will be important later for target tracking problems where we will discuss channel capacities between robots and targets.

3.3 Robotic information gathering

This thesis applies techniques from information theory and combinatorial optimization to robot sensing and information gathering. As a robot moves through an environment it may obtain sensor data, reduce uncertainty in its environment model, and gain information rewards. Problems related to selecting actions to maximize quality of sensor data are prominent in robotics and control and studied with a wide variety of tools. The methods we propose for coordinating teams of robots complement algorithms for individual robots as both can contribute to improving completion times in time-sensitive sensing tasks. Here, we provide an overview of such techniques moving from general applications (Sec. 3.3.1), to more formal informative path planning problems (Sec. 3.3.2), and theory that characterizes how and why robots benefit from replanning and updating their decisions based on incoming sensor data (Sec. 3.3.3).

3.3.1 Active sensing and information-based control

Sensing and information gathering problems span a wide spectrum that includes tracking and localization problems [35, 59], robot state estimation [7, 93], sensing problems based on manipulation and mechanical interactions [1, 21, 53], and active perception problems for control and reconstruction with image data [10, 153]. Although methods for decision-making vary, submodular functions and information-theoretic objectives not only generalize across these domains but are also applied regularly [7, 35, 53, 59, 93, 153]. While we focus on time-sensitive mapping and sensing problems such as urban disaster response, the methods we propose may also be applied more broadly. Likewise, we desire planning techniques that apply across multiple sensing tasks as aspects of different tasks such as localization [35, 59] and mapping [37, 103] may arise simultaneously.

3.3.2 Informative path planning

Algorithms for informative path planning seek to plan a path for a robot to maximize information gain or some other submodular function, subject to a limited travel budget. Such problems have been studied extensively. Performance guarantees exist for algorithms on graphs [41, 169, 200], and sampling-based methods exist for optimization of paths in continuous spaces [90].

---

[1]The expression $X \to Y \to Z$ is a Bayes graph and states that the joint distribution of the random variables can be factored as $\mathbb{P}(X = x, Y = y, Z = z) = \mathbb{P}(X)\mathbb{P}(X = x|Y = y)\mathbb{P}(Z = z|Y = y)$.

(a) No obstacle **(b)** Obstacle
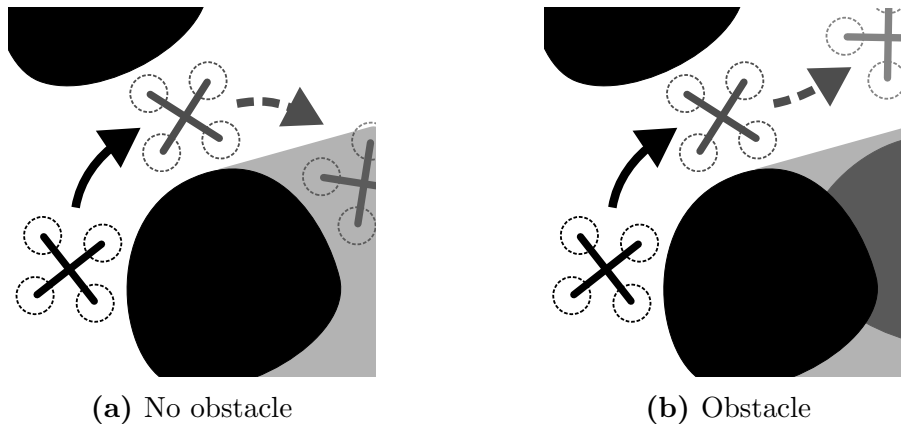
Figure 3.1: The illustrations above provide a typical example of adaptation (or replanning) during robotic exploration. Here, a robot will observe unknown space behind a boulder only as that robot navigates around to the back. This robot may benefit from *quick adaptation* by (a) navigating into the unknown space if there is no obstacle. Else, (b) the robot must be prepared to remain in free space if that unknown space turns out to be occupied.

These and related algorithms can also serve as suboptimal oracles to greedily optimize paths for individual robots in multi-robot problems [169], and performance guarantees can be rewritten in terms of the performance of the given planner. In this sense, path planners can be chosen independently of methods for coordinating the multi-robot team.

3.3.3 Decision processes and adaptation

Decision processes such as Markov decision process (MDPs), partially-observable Markov decision process (POMDPs), and variants thereof directly model continuing processes of selecting actions, obtaining observations, and selecting new actions to maximize some reward. Although rewards are typically functions of states and actions, specialized formulations permit rewards in terms of beliefs e.g. based on mutual information or entropy [5, 171]. Greedy algorithms have been applied to solve combinatorial subproblems when actions are sets of sensors [159] and in multi-agent problems to optimize sequences of policies [112], and those lines of work are complementary to our own.

Instead, adaptive submodularity [79] seeks to directly extend techniques for submodular optimization to decision processes. However, such approaches require strict regularity on objectives and the sets of actions to maintain monotonicity and diminishing returns. Mobile robots generally violate these conditions as sets of available sensing actions change while the robot moves through the environment. Other problems in robotics, such as manipulation, are more amenable to the requirements of adaptive submodularity [45, 96].

Few methods for decision processes are applicable to mobile robots and large-scale problems. However, Monte-Carlo techniques for POMDPs provide an avenue to addressing large-scale decision processes [167]. These techniques and applications in robotic exploration [116] inspired our use of Monte-Carlo tree search for single-robot planning in this thesis.

Otherwise, rather than seeking to optimize adaptation via policies—which is often

intractable—we may instead consider the benefit of adaptation and the cost of not adapting [6, 78, 91]. Operation in unknown environments, as in urban disaster response, also induces rapidly changing beliefs. Robots moving through cluttered environments may benefit from quickly responding to observations of occluded objects [76, 77] (see the example in Fig. 3.1), and further, teams of robots should maintain this ability to adapt and replan as appropriate for the time-scales of a given sensing problem.

3.4 Multi-robot sensing and coordination

The sensing problems that we have discussed to this point also come with multi-robot counterparts. We are interested in scenarios where robots interact locally while operating in close proximity and observing nearby regions of an environment, and our methods seek to enable robots to react to new observations both individually and collectively over short time spans. The techniques we develop leverage simple greedy algorithms and are scalable while providing strong performance guarantees. However, various other techniques exist and are applicable to the problems we consider. Best et al. [18] propose a multi-robot planner that applies Monte-Carlo tree search as we do but uses a different mechanism for multi-robot coordination—probability collectives—a generic optimization approach based on game theory. In this case, our approach will provide stronger guarantees on solution quality for planning in finite time. We also note that Sung et al. [176] propose a max-min formulation for tracking problems that addresses similar short time scales that satisfies performance guarantees before rounding. Sung et al. [176] also address a similar challenge as we do by developing an algorithm with computation time that is independent of the problem size. Although we study a slightly different class of sensing problems, we note that our mechanisms for coordination, based on sharing incremental solutions, are somewhat simpler and more general than the linear programming methods that they employ.

3.4.1 Sequential greedy maximization for multi-robot sensing

Singh et al. [169] are largely responsible for developing sequential planning methods for multi-robot sensing (based on methods for submodular maximization which we will discuss next in Sec. 3.5) and informative path planning.[2] In these methods, robots plan in a sequence, (as illustrated in Fig. 3.2) and each robot seeks to maximize a sensing objective given all prior decisions. For many objectives, solutions by sequential planning are also guaranteed to be within half of the optimal objective value, whereas obtaining optimal solutions is NP-Hard.

While Singh et al. [169] studied a mutual information objective, sequential planners

---

[2]Historical note: Singh et al. [169] are most commonly cited for specializing matroid-constrained submodular maximization to partition matroids for sensor planning and, more specifically, multi-robot path planning. Goundan and Schulz [81] observed around the same time that this kind of approach recreates a variant of the locally greedy algorithm described by Fisher et al. [69]. In fact, Singh et al. [169], Goundan and Schulz [81], and even also Williams [193] all appear to have independently (re)developed similar specializations to partition matroids (implicitly or explicitly) at around the same time.
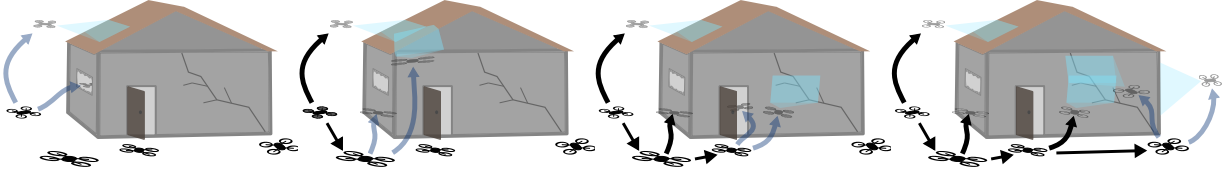
Figure 3.2: Four robots plan sequentially. Each robot selects an action to maximize a sensing objective given the decisions by the prior robots.

can provide suboptimality guarantees for wide variety of objective functions (which we will discuss in the next section). As a result, their work influenced not only a variety of works on informative planning [7, 151] but also problems with more varied objectives such as search for a moving target [89] and survivability-aware problems where robots may fail en-route to their destinations [102]. While most of these authors study similar multi-robot path planning problems, Jorgensen et al. [102] and Williams et al. [194] also apply related greedy planning methods to more complex assignment problems such as with constraints on coverage, resources (such as a limited number of runways), and network connectivity.

3.4.1a Single-robot sub-problems in sequential planning

A useful feature of sequential planners is that specialized planners like those in Sec. 3.3.2 can be used to implement the maximization step while retaining similar approximation guarantees [169] (that is, they serve as *maximization oracles*). This is particularly important because the alternative is to iterate over all possible solutions for a given robot—doing so is typically intractable for path planning problems as the number of possible solutions is generally exponential in both the time horizon and the number of available actions at each step. Moreover, this approach enables designers to take advantage of advances for planning for individual robots and to apply those methods to planning for multiple robots with minimal effort.

3.4.1b Adaptation in sequential planning

In non-adaptive settings, (where robots do not replan after obtaining new information) solutions can be found offline and without strict constraints on time or computation. For example, Singh et al. [169] plan paths for a sensing task on a lake and compute the entire paths beforehand. However, this thesis studies adaptive settings (Sec. 2.1.1f) where the plan or sensing actions change in response to incoming sensor data during execution of the sensing task, and sequential planning forms a component of a solver for a receding-horizon optimization problem (Sec. 2.1.2) [20, 38, 116].

Sequential planners for these receding horizon problems then must satisfy time constraints for replanning. When planning with large numbers of robots, meeting time constraints may require improving efficiency via parallel versions of these planners or by parallelizing the single-robot planners. The following sections will describe the theory for these sensing problems and provide more detail regarding parallel and distributed planning.

3.5   Submodular maximization

The sequential planning and informative path planning problems that arise in this thesis broadly fall into the framework of so-called submodular maximization problems. Specifically, many sensing objectives *increase monotonically* and exhibit *diminishing returns* (submodularity). These classes of functions have been studied extensively from the perspective of combinatorial optimization, and there are numerous results for exact and approximate optimization in various settings. In this way, monotonic and submodular functions studied in combinatorial optimization are analogous to the convex functions studied in nonlinear optimization: each has been studied intensively and optimization problems involving either class of functions can be solved efficiently[3]. Objective functions with these properties appear frequently in robotics and especially in sensing problems. For example, notions of monotonicity or diminishing returns intuitively describe how the value of a collection of sensing actions changes as actions are added or removed. Later, we will discuss submodular objective functions such as common cases of mutual information objectives and different notions of sensor coverage.

Going a step further, we are also interested in different kinds of constraints on these optimization problems, just as constraints are common in other kinds of optimization problems. Whereas this section discusses optimizing functions of sets, the constraints in question will determine which sets to consider. These constraints model concepts such as which robots can perform an action, how many actions a robot can perform at once, or which sets of actions will result in collisions between robots. In this sense, these constraints will describe which sets of actions teams of robots can feasibly execute.

3.5.1   Set functions and submodularity

Consider a finite—but possibly extremely large—set of sensing actions $\mathscr{U}$, commonly referred to as the *ground set*. The ground set describes the universe of set elements that are available in a given optimization problem e.g. the set of all sensing actions or finite-horizon plans available to any robot. A *set function* $g : 2^{\mathscr{U}} \to \mathbb{R}$ maps a collection of such actions to a real value—often a reward—where $2^{\mathscr{U}}$ is the power set of the ground set, the set of all subsets of $\mathscr{U}$.

Most objectives have zero value at the empty set, or else that value can be subtracted to normalize the objective.

Definition 1 (Normalized). *A set function g is normalized if* $g(\emptyset) = 0$.

A function is monotonically increasing[4] (or monotonic) if its value cannot decrease.

Definition 2 (Monotonically increasing). *A set function g is monotonically increasing if for any* $A \subseteq B \subseteq \mathscr{U}$ *then*

$$g(A) \leq g(B) \tag{3.13}$$

---

[3]There are connections between submodular functions and both convex and concave functions, but those are unrelated to this analogy.

[4]Some texts describe the functions that we call monotonically increasing as being "non-decreasing" to emphasize the weak inequality in the definition.

Next, a set function is submodular if it exhibits diminishing returns.

Definition 3 (Submodular (supermodular)). *A set function g is submodular if for any $A \subseteq B \subseteq \mathcal{U}$ and $C \subseteq \mathcal{U} \setminus B$ then*

$$g(A \cup C) - g(A) \geq g(B \cup C) - g(B). \tag{3.14}$$

Here, the marginal gain for C decreases after adding elements to A to obtain B. The negation $-g$ is supermodular.

3.5.1a Relevant submodular functions

Objective functions for robot sensing are frequently monotonic and submodular. Mutual information (3.5) and entropy (3.1) naturally characterize reduction of uncertainty with respect to an unknown environment, and robots may select sensing actions which reduce that uncertainty. Mutual information with respect to a target variable satisfies monotonicity and submodularity when observations are conditionally independent [107]. Specifically, if we associate elements of a ground set $a \in \mathcal{U}$ with conditionally independent observations Y_a, the set function $g(A) = \mathbb{I}(X; Y_A)$ is normalized, monotonic, and submodular [107]. However, designers may consider other kinds of objectives, either for different tasks or because mutual information can be expensive to evaluate. For example, we will also study simpler coverage-like objectives which satisfy the same properties and can be thought of as representing sums of rewards for observing objects in subsets of the environment.

3.5.1b Notation for sets and set functions

For brevity, we will treat set functions as multi-variate functions with implicit unions so that $g(A, B) = g(A \cup B)$ where $A, B \subseteq \mathcal{U}$. We will also implicitly convert elements of the ground set to subsets so that $g(x) = g(\{x\})$ for $x \in \mathcal{U}$. We also write incremental changes as $g(A|B) = g(A, B) - g(B)$ which is analogous to conditioning for mutual information (3.7) and also expresses the discrete derivative of g at B with respect to A (discussed next in Sec. 3.5.1c). Using this notation, the expression for submodularity (3.14) can be written more concisely as

$$g(C|A) \geq g(C|B), \tag{3.15}$$

recalling that $A \subseteq B$.

When dealing with sets and set functions, being able to concisely specify and index into sets will be advantageous. Often sets will be specified with an implied ordering as in $X = \{x_1, \ldots, x_n\}$. Subscripts will then be used for indexing subsets as in the following range $X_{1:i} = \{x_1, \ldots, x_i\}$ for $i \leq n$ or sometimes using sets of indices so that $X_A = \{x_j \mid j \in A\}$.

3.5.1c Discrete derivatives of set functions

The definitions of monotonic (3.13) and submodular (3.14) functions are closely related. The value of a monotonic function may only stay the same or increase, and the values of incremental changes of submodular functions may only stay the same or decrease.

(a) $g(A) = g(A|\emptyset)$ **(b)** $g(B|A)$ **(c)** $-g(B;C|A)$
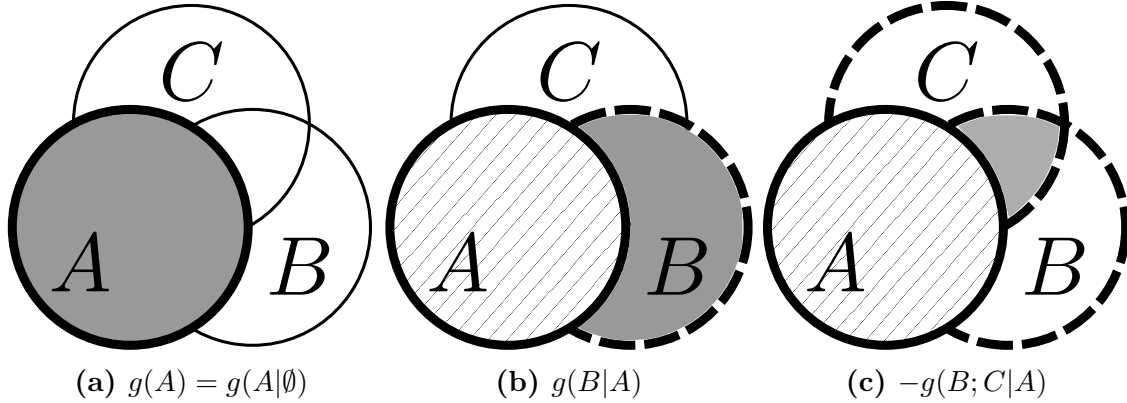
Figure 3.3: In this figure, areas shown in gray are equal in size to (a) zeroth, (b) first, and (c) second derivatives of a set function g, each evaluated at A. As will be discussed in more detail, later, the derivatives of some set functions are monotonic such as in this illustration of area coverage. If we interpret A as a subset of $\mathbb{R}^2$, (a) area coverage increases monotonically in A; (b) marginal gains decrease monotonically; (c) and the second derivative increases monotonically (the negation of which, the highlighted area, decreases).

We can go further and define higher derivatives of set functions and corresponding monotonicity conditions. To assist in understanding, Fig. 3.3 provides examples of such derivatives for an area coverage objective. Given disjoint variables $A, B, C \subseteq \mathscr{U}$ the second derivative of g at C in directions A and B is

$$g(A; B|C) = g(A, B, C) - g(A, C) - g(B, C) + g(C).$$

Expressions of second derivatives will also arise with respect to the empty set. Such expressions take the form of

$$g(A; B) = g(A|B) - g(A) = g(A, B) - g(A) - g(B),$$

given that $g(\emptyset) = 0$, and being able to recognize them will be useful. For higher derivatives, given $X \subseteq \mathscr{U}$ and $Y_1, \ldots, Y_n \subseteq \mathscr{U}$, all disjoint, we can define the nth derivative of g with respect to $Y_1, \ldots, Y_n$ recursively as

$$g(Y_1; \ldots; Y_n|X) = g(Y_1; \ldots; Y_{n-1}|X, Y_n) - g(Y_1; \ldots; Y_{n-1}|X). \tag{3.16}$$

Note that, just as for a continuous derivatives, derivatives of set functions are not sensitive to the order of the directions.

This notation for set functions and their discrete derivatives also intentionally overlaps with notation for entropy and mutual information. Continuing with this analogy, the mutual information could be thought of as the *negation* of the second derivative of entropy were it interpreted as a set function as $\mathbb{I}(A; B|C) = \mathbb{H}(A|C) - \mathbb{H}(A|B, C)$. Therefore, one might adopt the colloquialism $\mathbb{I}(A; B|C) = -\mathbb{H}(A; B|C)$, *although no such notation will be used in this text.*

Foldes and Hammer [70] provide a slightly different definition of the derivative. As this thesis will reference their work repeatedly, we provide a brief proof of equivalence in Appendix B.1.
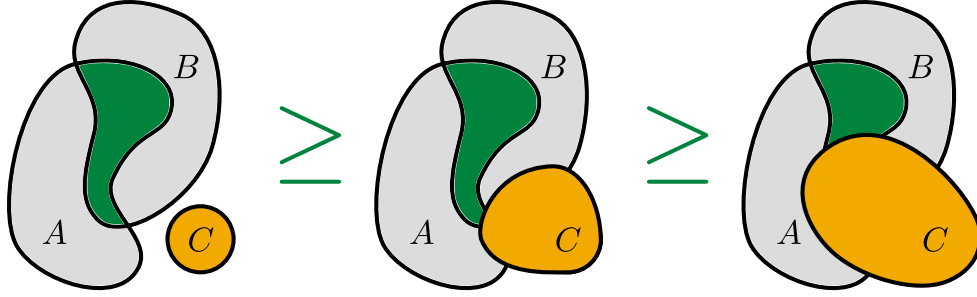
Figure 3.4: The above illustrates the 3-increasing condition for coverage. Let the area covered by a number of sets (or sensing actions) be a function g. The green area corresponds to an amount of sensing redundancy between two sensor coverage actions A and B conditional on a third observation C. The size of that area in green is equal to the negation of a second derivative $-g(A; B|C) = g(A|C) + g(B|C) - g(A, B|C)$, and the area decreases monotonically as the region represented by C increases. Then, second derivatives $g(A; B|C)$ *increase* monotonically for coverage objectives so we say that g is 3-increasing (as the first monotonicity condition corresponds to the zeroth derivative).

3.5.1d Higher-order monotonicity and derivatives

Foldes and Hammer [70] also describe monotonic and submodular set functions as being, respectively, 1-increasing and 2-decreasing and provide detailed discussion of these conditions. More generally, if the nth derivative of a function is non-negative, then derivatives of order $n-1$ increase monotonically, and we say that the function is n-increasing. Then, if that derivative is non-positive, the function is instead n-decreasing. Later, we will apply this terminology in a requirement that certain objective functions are 3-increasing. In particular, weighted set coverage is 3-increasing, and we illustrate the intuition in Fig. 3.4. Later, we will provide more direct analysis when we encounter this condition again.

Higher-order monotonicity conditions, such as this, are not yet common in the literature on optimizing set functions but are experiencing increasing interest [44, 84, 106, 191]. Wang et al. [191] apply the 3-increasing condition to develop a variation of the continuous greedy algorithm, and Chen et al. [44] study functions with alternating monotonicity conditions, something we will also encounter. Measures of redundancy [68] and analogues of mutual information [94] also fit the form of second derivatives and are highly relevant to works on 3-increasing functions.

3.5.1e Chain rule for derivatives of set functions

When proving bounds for submodular maximization techniques, we will often want to decompose a derivatives with respect to sets (say $A \subseteq \mathcal{U}$) in terms of the contributions of individual elements (i.e. $a \in A$). Chain rules allow us to do so.

Previously, we have discussed such chain rules in the context of information theory (Sec. 3.2). Set functions and their derivatives satisfy analogous chain rules. Later, we will require chain rules for both first and second derivatives of set functions. Toward this end, the following defines a chain rule for any derivative of a set function.

Lemma 1 (Chain rule for derivatives of set functions). *Consider sets $Y_1, \ldots, Y_n, X \subseteq \mathcal{U}$, all disjoint. Then, writing the elements of Y_n as $Y_n = \{y_{n,1}, \ldots, y_{n,|Y_n|}\}$, the derivative of*

g can be rewritten in terms of derivatives with respect to the individual elements of Y_n as

$$g(Y_1; \ldots; Y_n | X) = \sum_{i=1}^{|Y_n|} g(Y_1; \ldots; Y_{n-1}; y_{n,i} | Y_{n,1:i-1}, X). \tag{3.17}$$

The proof of Lemma 1 is in Appendix B.2.

3.5.2 Matroids and independence systems

This thesis focuses primarily on multi-robot planning problems that can be described using combinatorial constraints. Such constraints that describe feasible solution sets are called *set systems*. However, a completely general constraint would not be useful for the purpose of optimization. Independence systems describes constraints that admit subsets of any feasible set and which is the most general kind of set system that we will find useful.

Definition 4 (Independence system). *A tuple $(\mathscr{U}, \mathscr{I})$ where $\mathscr{I}$ is composed of subsets of the ground set such that $I \in \mathscr{I} \to I \subseteq \mathscr{U}$ is an independence system if $\mathscr{I}$ satisfies a heredity property so that for all $I_2 \subseteq I_1 \in \mathscr{I}$ then $I_2 \in \mathscr{I}$ and, additionally, $\mathscr{I}$ is non-empty ($\emptyset \in \mathscr{I}$).*

Multi-robot non-collision constraints form one example of an independence system; eliminating a robot from a non-colliding multi-robot plan will not cause the remaining robots to be in collision.

A matroid is a special case of an independence system and generalizes the notions of linear independence from vector spaces to set systems. Matroids have been studied extensively in the optimization literature (discussed in more detail in following sub-sections) and find use in this work to model product spaces in multi-robot planning problems.

Definition 5 (Matroid). *An independence system $(\mathscr{U}, \mathscr{I})$ that also satisfies the exchange property is a matroid, that is if for all $I_1, I_2 \in \mathscr{I}$ such that $|I_1| > |I_2|$ there exists some $x \in I_1 \setminus I_2$ such that $I_2 \cup \{x\} \in \mathscr{I}$.*

The reader may think of elements that can be added to a given set as being effectively perpendicular to that set. For more detail on this topic, the reader may wish to refer to Schrijver [162]. A consequence of the exchange property is that all maximal independent sets—those to which we cannot add any more elements—have the same cardinality which is referred to as the *rank* of the matroid or rank($\mathscr{I}$). Additionally, in this work, the rank will also be equal to the number of robots or other sensing agents.

Although matroids are the focus of numerous related works on optimization, a matroid is a somewhat more general structure than is necessary in this thesis, and we will gain in efficiency from using slightly more specialized constraints. A *partition matroid* is a special case of a matroid and is nearly directly analogous to a product space.

Definition 6 (Partition matroid). *Consider a partitioning of the ground set into blocks $\mathscr{B}_i \subseteq \mathscr{I}$ for all $i \in \{1 \ldots n\}$ such that $\mathscr{B}_i \cap \mathscr{B}_j = \emptyset$ for all $i \neq j$. Then $(\mathscr{U}, \mathscr{I})$ is a partition matroid if*

$$\mathscr{I} = \{I \subseteq \mathscr{U} \mid a_i \geq |I \cap \mathscr{B}_i|, \ \forall i \in \{1 \ldots n\}\}$$

where $a_i \in \mathbb{Z}_{>0}$ is the maximum number of elements that can be selected from a given block.

When $a_i = 1$ for all i, the partition matroid will be referred to as a *simple partition matroid* [162, Sec. 39.4]. These matroids will be used to model joint action spaces of multi-robot teams where $\mathscr{B}_i$ is the space of trajectories or actions available to robot i, assuming that there are no inter-robot constraints on action selection. Although the techniques developed in this thesis will all use simple partition matroids, they all extend to general partition matroids with slight modification.

Finally, we will discuss one more (and simpler) type of matroid, the uniform matroid, which expresses a cardinality constraint on sets in the set system.

Definition 7 (Uniform matroid). *A uniform matroid is an independence system $(\mathscr{U}, \mathscr{I})$ that fits the form fits the form $\mathscr{I} = \{I \subseteq \mathscr{U} \mid n \geq |I|\}$ where $n \in \mathbb{Z}_{>0}$.*

3.5.3 Submodular maximization and greedy algorithms

The problems studied in this thesis involve selecting control actions to maximize notions of sensing quality such as mutual information with a map or sensor coverage. When such objectives are submodular (and generally also monotonic and normalized), the resulting optimization problems fit broadly into the realm of submodular maximization problems. Then, when these problems involve planning actions for a multi-robot team, as hinted in Sec. 3.5.2, we will often also introduce a matroid constraint of some form.

Problem 1 (Submodular (monotonic, normalized) maximization problem). *Let $g : 2^{\mathscr{U}} \to \mathbb{R}$ be a submodular, monotonic, and normalized set function and $(\mathscr{U}, \mathscr{I})$ a set system. Together, these define a submodular maximization problem*

$$X^{\star} \in \arg\max_{X \in \mathscr{I}} g(X)$$

with optimal solution(s) $X^{\star}$.

Now, unless otherwise specified, the objectives in any submodular maximization problems discussed in this text will also be monotonic and normalized. We will primarily only refer to other cases when discussing related work on other problems or in passing. Instead, we now focus on the form of the constraint $\mathscr{I}$ and, eventually, further conditions on g.

A general set system may present $2^{|\mathscr{U}|}$ feasible solutions. Although $|\mathscr{I}|$ will generally be smaller, solving submodular maximization problems by iteration is typically intractable. Recall that when $(\mathscr{U}, \mathscr{I})$ is an independence system (Def. 4), subsets of feasible solutions are also feasible. Solutions can then be constructed incrementally by searching $\mathscr{U}$ for new elements to add to partial solutions. In fact, algorithms that construct such solutions greedily often guarantee solutions within a constant factor of the optimal solution.

Algorithm 1 (Greedy submodular maximization). *Greedy submodular maximization produces feasible (suboptimal) solutions to submodular maximization problem (Prob. 1) where $(\mathscr{U}, \mathscr{I})$ is an independence system. Define an incremental solution X_i such that $X_0 = \emptyset$ and*

$$X_i \in \arg\max_{\{x\} \cup X_{i-1} \in \mathscr{I}} g(x|X_{i-1}). \tag{3.18}$$

This algorithm outputs a solution, designated $X^{\mathrm{g}} = X_n$, once no more solution elements can be added, i.e. once there is no $x \in \mathscr{U}$ such that $\{x\} \cup X_n \in \mathscr{I}$.

Solutions can be obtained far more efficiently than by iterating over $2^{\mathscr{U}}$ when using this algorithm, but keep in mind that solving (3.18) may still be intractable when $\mathscr{U}$ is large. Developing methods that avoid incurring costs of such sequential maximization steps is a recurring theme in related work and this thesis.

3.5.3a Greedy submodular maximization with cardinality constraints

Cardinality-constrained submodular maximization forms a simple and particularly common submodular maximization problem. In this case, the statement that the cardinalities of feasible solutions are no greater than some value produces a uniform matroid (Def. 7) constraint.

Problem 2 (Cardinality-constrained submodular maximization). *Any instance of Prob. 1 where $(\mathscr{U}, \mathscr{I})$ is a uniform matroid is an instance of cardinality-constrained submodular maximization.*

Cardinality-constrained problems arise frequently in applications [32, 51, 101] due to the simplicity of the constraint as well as because of strong tightness and hardness guarantees. These tightness and hardness guarantees are important to us because uniform matroids are a special case of set systems such as partition matroids and because many results for cardinality constrained problems also apply to problems with more complex constraints (such as by forming lower bounds). Nemhauser et al. [140] proved that greedy solutions to cardinality-constrained problems satisfy a constant-factor suboptimality guarantee $g(X^{\mathrm{g}}) \geq (1 - 1/e)g(X^{\star}) \approx 0.632g(X^{\star})$. Soon after, some of the authors of that work also proved that this approximation guarantee is optimal in the value oracle model (where g is treated as a black box) for algorithms that evaluate g a polynomial number of times [141]. Still, certain classes of these problems could be easier to solve or could be solved more quickly given complete access to the problem representation. Feige [66] proved that no better bound can be achieved by polynomial time algorithms for set coverage objectives[5] unless P=NP. Not only is this a stronger guarantee, but the simplicity of the set coverage objective enables reductions to other relevant objectives. This includes generalizations of coverage discussed later in this text and, thanks to Krause and Guestrin [107], Shannon mutual information on Bayes graphs.

3.5.3b Greedy submodular maximization on general matroids

While cardinality-constrained problems provide hardness results that are relevant to this thesis, matroid-constrained problems will *generalize* the multi-robot planning problems that we will discuss.

Problem 3 (Matroid-constrained submodular maximization). *An instance of Prob. 1 where $(\mathscr{U}, \mathscr{I})$ is a matroid is an instance of matroid-constrained submodular maximization.*

Fisher et al. [69] proved that when $(\mathscr{U}, \mathscr{I})$ is a matroid, solutions by Alg. 1 satisfy $g(X^{\mathrm{g}}) \geq 1/2 g(X^{\star})$. Although this thesis applies greedy algorithms, more recent works [31, 67] propose more advanced methods such as the continuous greedy algorithm [31, 185]

---

[5]Set coverage refers to "covering" a set $\mathscr{C}$. The elements of the ground set $x \in \mathscr{U}$ are subsets of this set $x \subseteq \mathscr{C}$, and the objective is to maximize the cardinality of their union $g(X) = \left| \bigcup_{x \in X} x \right|$.

which guarantee results within $1 - 1/e$ which is the best possible guarantee given the hardness results for cardinality-constrained problems.

3.5.3c   Locally greedy (sequential) planning on simple partition matroids

Although direct application of Alg. 1 provides solutions within half of optimal, as mentioned, the greedy step (3.18) involves iteration over the entirety of $\mathscr{U}$ which may be expensive. This can be avoided in cases such as when the matroid constraint is a partition matroid. For simplicity, we will focus on the special case of simple partition matroids.

Problem 4 (Simple partition matroid-constrained submodular maximization). *An instance of Prob. 1 where $(\mathscr{U}, \mathscr{I})$ is a simple partition matroid is an instance of partition matroid-constrained submodular maximization.*

Problems constrained with simple partition matroids admit a relaxation of the greedy algorithm: rather than computing the incremental solutions by maximizing over the entire ground set, increments can be computed by maximizing over the blocks of the partition.

Algorithm 2 (Locally greedy algorithm for simple partition matroids). *The locally greedy algorithm produces feasible (suboptimal) solutions to partition matroid-constrained maximization submodular maximization problems (Prob. 4). Define solution elements x_i such that*

$$x_i \in \arg\max_{x \in \mathscr{B}_i} g(x|\{x_1, \ldots, x_{i-1}\}) \tag{3.19}$$

to produce the output $X^{\mathrm{g}} = \{x_1, \ldots, x_n\}$ where n is the rank of the partition matroid.

Notably, this algorithm implements the same sequential planning process as in Sec. 3.4.1 and will gain guarantees on solution quality for submodular functions. This algorithm is also similar to the prior algorithm for greedy submodular maximization (Alg. 1). However, because the maximization steps are now over individual blocks, computation of a complete solution requires only a single pass over the ground set. Yet, crucially, solutions via Alg. 2 satisfy the same bounds as for Alg. 1: $g(X^{\mathrm{g}}) \geq 1/2 g(X^{\star})$.

The form of the locally greedy algorithm is particularly desirable for distributed implementations on multi-robot systems. Recall that a simple partition matroid can be used to model the joint space in a multi-robot planning problem. In this case, $\mathscr{B}_i$ is the local set of actions available to the ith robot. Robots can then plan for themselves by search over these local sets of actions and given access to prior robots' plans but without access to the entire ground set. However, dependency on prior solution elements is a barrier to parallel computation in distributed implementations. One of the contributions of this thesis is to eliminate that barrier in multi-robot sensing problems.

3.5.3d   Inequalities and intuition for submodular maximization

Monotonicity serves the starting point in the proofs of many common performance bounds. If $X^{\star}$ is an optimal solution to a submodular maximization problem (Prob. 1) and $X \subseteq \mathscr{U}$, then

$$g(X^{\star}) \leq g(X^{\star}, X). \tag{3.20}$$

Typically, X is the full solution obtained by the greedy algorithm, but that is not strictly necessary [82]. We can exchange an element of the optimal solution $x^\star \in X^\star$ for an element of a greedy solution by applying the principle of greedy choice

$$g(x^\star | \hat{X}) \leq g(x^g | \hat{X}) \tag{3.21}$$

where $\hat{X}$ is generally a subset of the greedy solution and $x^\star \in B$ belongs to the feasible set $B \subseteq \mathscr{U}$ of a greedy maximization step that solves $x^g = \max_{x \in B} g(x | \hat{X})$. Observe that if we instead have an expression for $g(x | \bar{X})$ where $\hat{X} \subseteq \bar{X} \subseteq \mathscr{U}$ we can apply submodularity to replace $\bar{X}$ with $\hat{X}$ because $g(x | \bar{X}) \leq g(x | \hat{X})$. If the greedy solutions are also suboptimal, we may also apply the expression for the suboptimality to (3.21) to replace the optimal greedy solution elements with a suboptimal ones.

3.5.3e Online bounds

As Minoux [132] observes, this intuition can be applied more generally to obtain online bounds [79, 109] for solutions obtained by any algorithm such as by applying (3.20) and bounding the terms of the sum in $g(X^\star, X) \leq \sum_{x^\star \in X^\star} g(x^\star | X)$ e.g. by maximizing over ground set $\mathscr{U}$ or an independence system $\mathscr{I}$. These online bounds have been used to demonstrate that real solutions by greedy algorithms can often perform significantly better than worst case bounds in practice [79, 109]. Similar bounds will later prove to be useful for characterizing solution quality for multi-robot exploration.

3.6 Distributed and parallel algorithms for submodular maximization

An important challenge in this thesis is to reduce computation time for submodular maximization in multi-robot sensor planning. Ideally, this computation time would be independent of the number of robots in the team, and we would achieve this by taking advantage of parallel computation. However, that is easier said than done due to sequential constraints on the decision process and constraints on information access. Further, although there is a growing body of work on parallel and distributed submodular maximization, many existing methods are impractical for multi-robot sensor planning. Still, a few works propose methods that are relevant to this thesis, and we will seek to take advantage of their contributions.

3.6.1 Assumptions and model for distributed computation

Figure 3.5 illustrates the distributed setting for the submodular maximization problems that we will study. In this work, robots will solve submodular maximization problems (Prob. 4) where each robot $i \in \mathcal{R}$ selects an action from a block of a partition matroid $x \in \mathscr{B}_i$. We assume robots have access to a locally approximate model of the environment which we refer to here as θ_i (and which we will omit for most of this text) via a distributed
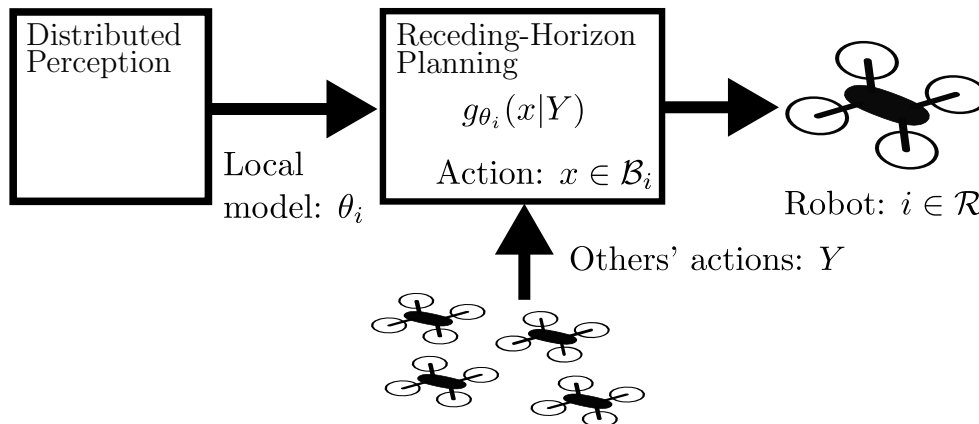
Figure 3.5: Illustration of the model of computation for distributed sensor planning. Each robot $i \in \mathcal{R}$ has access to a block of a partition matroid $\mathcal{B}_i$ which represents that robot's space of control actions. The perception system then provides a locally consistent model θ_i. Given this model, a robot can accurately approximate marginal gains in the sensing objective $g_{\theta_i}(x|Y)$ for its own actions $x \in \mathcal{B}_i$ given a set of others' actions Y (e.g. numbers of newly observed cells in an occupancy map for x or the reduction in uncertainty of positions of nearby targets). But, that robot *cannot compute marginal gains for other, possibly distant, robots* (and distant targets or regions of an occupancy map).

perception system. With this model a robot can approximate marginal gains for its own actions $g_{\theta_i}(x|Y)$ given others $Y \subseteq \mathcal{U}$. However, a robot may not evaluate marginal gains for assignments to other robots because that robot may not have an up-to-date model of distant regions of an occupancy map or estimates of positions of distant targets.

Additionally, while evaluating different parallel and distributed algorithms, we will sometimes write results in terms of the number of robots $n_{\mathrm{r}} = |\mathcal{R}|$ based on the following assumption:

Assumption 1 (Proportional problem size)**.** *Consider a team of n_{r} robots and a submodular sensor planning problem (in the form of Prob. 4). The size of the ground set is proportional to the number of robots $|\mathcal{U}| \in \Theta(n_{\mathrm{r}})$; the rank of the matroid constraint is n_{r}; and the blocks of the partition matroid have constant size $|\mathcal{B}_i| \in \Theta(1)$ for all $i \in \mathcal{R}$.*

3.6.2 Quantities for evaluating of parallel and distributed algorithms

A number of properties of parallel and distributed submodular can contribute to increases in computation time in multi-robot sensor planning. Consider a team of n_{r} robots that plan using $O(n_{\mathrm{r}})$ onboard processors.[6] We are interested in the following quantities:

- **Adaptivity:** The number of sequential calls to the objective g. See the next section (Sec. 3.6.3)

- **Parallelism:** The number of parallel queries to the objective g that an algorithm assumes.

---

[6]Or, for the purpose of this section, the robots may be supported by a centralized resource with the same number of processors.

40

- **Query number:** The maximum number of queries to the objective g. Given an algorithm that runs with k queries, the n_r processors require at least $O(k/n_r)$ time.

- **Query size:** This is the cardinality of the largest set $|X|$ that an algorithm uses to query the objective g. For derivatives (3.16), (such as the marginal gain) this includes elements of the sets representing the derivative directions.

Previously, Sec. 3.5 mentioned that different algorithms take different numbers of passes over the ground set. In the same direction, the general greedy algorithm (Alg. 1) involves $\Theta(n_r{}^2)$ queries because it maximizes over the entire ground set at each step. On the other hand, the locally greedy algorithm (Alg. 2) only completes a single pass for $\Theta(n_r)$ queries. Parallelizing the general greedy algorithm over n_r processors may then obtain similar computation times as the local algorithm on a single processor.

The total number of queries for the continuous greedy algorithm is more subtle. However, we can note that *each integration step* requires at least $\Omega(n_r)$ queries (technically, this scales with the rank of the matroid, but the rank is n_r in all cases that we encounter). Algorithms that solve subproblems with the continuous greedy algorithm on single processors require at least $\Omega(n_r)$ time.

Additionally, we are not aware of any algorithm (other than what we propose) that obtains constant-factor suboptimality with a maximum query size that is less than the number of robots.[7] Instead, we are able to achieve constant query size and suboptimality in certain settings by allowing robots to select actions for themselves given limited access to prior selections.

3.6.3   Adaptivity and bounds on numbers of sequential steps

Nominally, greedy methods (as in Alg. 2) operate sequentially (e.g. with one step per robot) because each decision in the sequence takes all prior decisions into account. A few works, that we will discuss later in more detail, consider variations of such greedy algorithms where agents can ignore some prior decisions [75, 82]. However, if we fix the degree of suboptimality, the best algorithms in this class require a number of sequential steps that is *proportional to the number of robots*, just as for other greedy algorithms.

More broadly: How much can parallel computation reduce planning time for submodular maximization with any algorithm? To answer this question, recent works have begun to study the *adaptive complexity* of submodular maximization problems. Balkanski and Singer [11] describe the adaptive complexity as "the minimal number of sequential rounds required to achieve a constant factor approximation when polynomially-many queries [to the submodular objective] can be executed in parallel at each round." Moreover, *adaptivity* is the worst-case number of sequential steps for a given algorithm; an algorithm that executes in k steps is k-adaptive. By virtue of parallel execution, none of the queries to the objective during a given round can depend on another, and the set of queries for a given adaptive round is known at the beginning of the round.

---

[7]The query size for R-lRSP depends on the size of the largest communication neighborhood.

Caution: Adaptivity is related to but distinct from computational complexity and the (parallel) duration of computation. Adaptivity provides a *lower bound on the growth of the computation time*, but the actual computation time may grow more quickly, depending on factors such as the number of available processors and the application context. Nevertheless, decreasing adaptivity is a necessary first step toward developing constant-time algorithms in this thesis.

Looking toward applications in multi-robot sensing, the best known algorithm for submodular maximization with a matroid constraint has adaptivity: $O(\log|\mathscr{U}|\log(\text{rank}\,\mathscr{I}))$ [12, 42]. Further, Balkanski and Singer [11] proved that no algorithm can obtain adaptivity better than $\tilde{\Omega}(\log|\mathscr{U}|)$ (lower-bounded by $\log|\mathscr{U}|$ up to lower-order logarithmic terms). Critically, these results improve on prior algorithms for submodular maximization that obtain constant factor guarantees, including the continuous greedy algorithm [31], which has adaptivity of at least $\Omega(|\mathscr{U}|)$ [12, Sec. 3.1] in general.[8] However, we caution that the continuous greedy algorithm may have better or even constant adaptivity on the class of problems that we study—the same analysis techniques that we apply to sequential planning in this thesis might also be applicable to the continuous greedy algorithm to obtain performance bounds where the number of integration steps is independent of the number of robots.

Further, one limitation of low-adaptivity methods is that converging to non-trivial constant factor bounds (e.g. to obtain a $1 - 1/e - \epsilon$ for small ϵ) can require a large (polynomial) number of steps in both theory and practice as discussed by [24] and [119]. However, Breuer et al. [24] presented an algorithm for cardinality-constrained problems which obtains both low-adaptivity and efficiency in practice and is the first to do so. Still, no such algorithm exists for more general matroid constraints. Likewise, although low-adaptivity results characterize what is possible for submodular maximization algorithms, they also depend on shared memory models of computation[9] which makes them difficult to adapt to distributed computation on multi-robot teams.

Table 3.1 summarizes the results we have discussed in this section in terms of the number of robots n_{r} in a sensing problem. Moreover, the adaptive complexity of submodular maximization with a matroid constraint lies in $\tilde{\Omega}(\log n_{\text{r}})$. In order to develop algorithms with constant adaptivity, we will need to restrict the problem class further and rely on additional structural properties of the sensing problems we study.

---

[8]The adaptivity of the continuous greedy algorithm depends on the number of discrete steps in an integral that are necessary to obtain a performance bound. Balkanski et al. [12] (informally) provide a *lower* bound on this number. Additionally, suboptimality guarantees [31] typically produce upper bounds on the adaptivity.

[9]Parallel Random Access Machines (PRAM)

Description	Lower bound (best possible adaptivity)	Upper bound (best known algorithm)
For *any* algorithm that achieves constant-factor performance (general adaptive complexity) [11, 12]*	$\tilde{\Omega}(\log n_{\mathrm{r}})$	$O(\log^2 n_{\mathrm{r}})$
For variations of the local greedy algorithm (Alg. 2) that allow robots to ignore prior decisions [75, 82]	$\Omega(n_{\mathrm{r}})$	$O(n_{\mathrm{r}})$

*The tilde in $\tilde{\Omega}$ or "Soft-Omega" indicates that the expression absorbs lower-order logarithmic terms.

Table 3.1: Adaptivity results (numbers of sequential steps) for submodular maximization on partition matroids within a constant factor of optimal in terms of the number of robots n_{r}. This thesis presents algorithms that achieve $O(1)$ adaptivity on a slightly restricted class of submodular maximization problems.

3.6.4 Parallel algorithms for submodular maximization

Many parallel algorithms for submodular maximization apply the popular *MapReduce*[10] model to solve problems with cardinality [133] and matroid [15, 16, 113] constraints. Although these algorithms may appear widely different at first glance, they share features that make them less suitable for multi-robot sensing. Each of these approaches [15, 16, 113, 133] seeks to reduce computation time by running greedy algorithms (sequential greedy algorithms or the continuous greedy algorithm) on small subsets of the ground set $\mathscr{U}$. In doing so, they distribute such subsets across processors and run some greedy algorithm on each subset in parallel to produce candidate solutions. Although the mechanisms for producing subsets differ, those that run sequential greedy algorithms [15, 113, 133] on those subsets all have high adaptivity (at least $O(n_{\mathrm{r}})$) which means that computation times will inevitably increase and become intractable for large enough numbers of robots. Likewise, all approaches, including those that apply the continuous greedy algorithm [16], produce full rank solutions when solving subproblems and so have $\Omega(n_{\mathrm{r}})$ queries and computation time on each processor.

Moreover, these approaches are also generally impractical in the multi-robot settings that we consider in this thesis as each robot would have to compute candidate solutions for the entire team. Additionally, distributing necessary information for that computation and collecting the candidate solutions would involve a significant amount of communication. By contrast, this thesis proposes approaches with low adaptivity where robots plan for themselves and communicate individual solution elements rather than full solutions.

3.6.5 Distributed algorithms for sensing and submodular maximization

A number of works have proposed distributed algorithms that are applicable to submodular maximization for different kinds of networked multi-robot and multi-agent systems [46, 75,

[10]The MapReduce model [60] is popular due to its simplicity, and roughly consists of three steps: 1) Distribute data across processors 2) Execute (map) some function on that data in parallel (say by squaring values) 3) Combine (reduce) the outputs from the prior step (e.g. by summing the squares).

82, 135, 194, 197]. However, none of these are able to provide suboptimality guarantees for in constant time as we would like for the problems we study.

Additionally, not all networked systems are relevant to the multi-robot settings that we study. Mokhtari et al. [135] and Xie et al. [197] present algorithms based on a setting where the objective is distributed across processors (e.g. $g(X) = \sum_{i=1}^{n_r} g_i(X)$) such as if the objective is derived from data that is spread out across all processors. Their approaches are interesting due to how they apply continuous consensus techniques [28, 143] to distribute the continuous greedy algorithm across processors. On the other hand, all processors operate on the entire ground set and constraint structure which requires $\Omega(n_r)$ time (much like algorithms in Sec. 3.6.4).

Instead, Robey et al. [154] extend the work of Mokhtari et al. [135] to the same setting that we apply for multi-robot sensing with the ground set and partition matroid distributed across the computation nodes. However, convergence still slows with increasing numbers of robots as evident from inspection of the error bound [154, Theorem 1] and due to limitations on adaptivity of the continuous greedy algorithm (refer to Sec. 3.6.3, [12]). Although we will not present novel variants of the continuous greedy algorithm, the methods for analysis of redundancy that we develop could possibly also be applied to algorithms like what Robey et al. [154] describe to provide constant time guarantees for the problems we study. Otherwise, the primary challenge for applying distributed versions of the continuous greedy algorithm [154, 164] to the problems we study is adapting the continuous greedy algorithm to use single-robot planners as maximization oracles. Doing so would require solving path planning problems on the multilinear extension of the objective [31] which requires costly sampling. Additionally, the large solution spaces endemic to path planning problems may adversely impact suboptimality as error bounds for the continuous greedy algorithm depend on the size of the ground set.

3.6.5a   Distributed task assignment for submodular maximization

Multi-robot sensor planning is also closely related to task assignment as both can often written in the form of similar submodular maximization problems. Specifically, there is a special case of submodular maximization with a partition matroid constraint (Prob. 4) that is relevant to task assignment, the submodular welfare problem.[11] Seeking to solve such problems, Choi et al. [46] developed the consensus-based bundle algorithm (CBBA) which implements a distributed version of the general greedy algorithm (Alg. 1). Later, Williams et al. [194] demonstrated that this approach also applies to problems with more complex constraints such as intersections of matroids. Most importantly, several works have extended [98, 166] CBBA, and others have demonstrated distributed implementations have been demonstrated on aerial robots [146]. Although CBBA-like approaches can provide significant speedups on relevant problems compared to the general greedy algorithm (which these approaches implement), they still have $O(n_r)$ adaptivity in the worst case. However, CBBA can also converge more quickly in practice which is an important point for comparison.

---

[11]Vondrák [185] provides an example of the reduction from a welfare problem to Prob. 4.

3.6.5b   Game theory for distributed submodular maximization

Distributed submodular maximization has also been studied from the perspective of game theory. Specifically, various equilibria satisfy constant factor suboptimality guarantees [155]. In fact, the strategy sets for games with submodular utility form a partition matroid, and Nash equilibria satisfy the same performance guarantees as greedy algorithms on matroid-constrained problems [184].

When agents employ algorithms that converge to such equilibria, game theoretic methods can be interpreted as producing algorithms for distributed optimization. Moreover, Goundan and Schulz [81] observed that one method of doing so would implement the locally greedy algorithm (Alg. 2) exactly. Additionally, this line of work has been applied to planning and sensing problems [112] and has produced interesting results for coverage on Voronoi decompositions [126]. Still, game theoretic methods do not yet provide tools for reducing the time to find a near-optimal solution which is a key challenge for this thesis.

3.6.5c   Distributed planning on directed acyclic graphs

Increasing numbers of robots, unreliable and constrained communication, and limited planning time all motivate a closer look at algorithms and models of computation for submodular maximization applied to multi-robot systems. A few works [75, 82] have begun to study the locally greedy algorithm (Alg. 2) on distributed computation models that are relevant to multi-robot sensor planning (where each agent in a network selects an action from a block of a partition matroid). Specifically, Gharesifard and Smith [75] consider variants of the locally greedy algorithm where agents have limited information about others' decisions according to a directed acyclic graph (DAG) Fig. 3.6). Crucially, this framework can describe planners that employ parallel computation [175], and for this reason, we apply similar models throughout this thesis. Gharesifard and Smith [75] also provided worst-case analysis on this model (see also Sec. 3.6.3) which Grimsman et al. [82] then extended with tighter bounds. However, planners based on these worst-case results have $\Omega(n_\mathrm{r})$ adaptivity when the suboptimality is held constant. To address this, we will focus on developing planners that can take advantage of features of problem structure beyond submodularity in order to provide near-optimal results with a constant number of sequential steps.
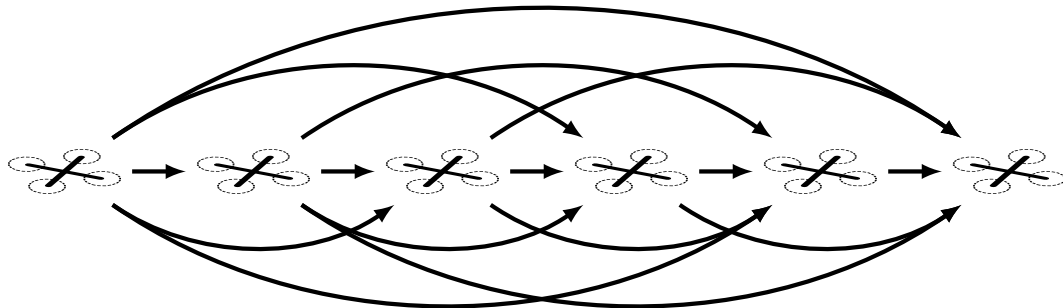
(a) A full sequential communication graph
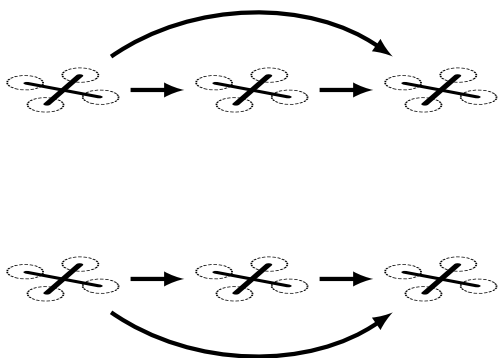
(b) A graph consisting of two cliques
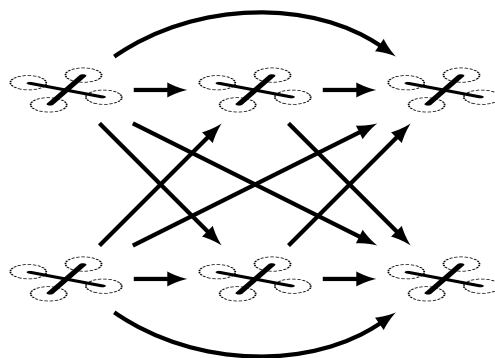
(c) A maximal graph for parallel planning

Figure 3.6: This figure illustrates several directed acyclic communication graphs for submodular maximization using the model proposed by Gharesifard and Smith [75]. (a) The standard locally greedy algorithm (Alg. 2) corresponds to a completed directed acyclic graph; (b) and although analysis by Gharesifard and Smith [75] and Grimsman et al. [82] leads to planners consisting of cliques, (c) algorithms proposed in this thesis use subsets of edges from maximal communication graphs with parallel computation.

Chapter 4

Toward Distributed Multi-Robot Exploration

This chapter (which originally appeared in [54, 56]) introduces an early version of the methods developed in this thesis in the context of a team of robots exploring an unknown environment. We will continue to use Monte-Carlo tree search for single-robot planning throughout this thesis as we also do in other related works [57, 76]. The parallel algorithm for submodular maximization (DSGA) that we describe is also a precursor to methods based on Randomized Sequential Partitions (RSP) which we will introduce later on. Although possibly interesting in its own right, an actual distributed implementation of DSGA would be more complex than the distributed implementation of RSP which we introduce in Chapter 8. Moreover, DSGA requires more communication than RSP methods and does not fully eliminate certain elements of sequential computation.

However, some of the contributions of this chapter are unique in this thesis. For example, this is the only chapter to address inter-robot collisions. Although this introduces some additional complexity, similar methods could be applied to the RSP planners that we discuss later. Similarly, this chapter provides the only results involving physical robots.

4.1   Introduction to exploration

We pose multi-robot exploration as the problem of actively mapping environments by planning actions for a team of sensor-equipped robots to maximize informative sensor measurements. In this work, we address the problem of planning for exploration with large teams of robots using distributed computation and emphasize online planning and operation in confined and cluttered environments.

Informative planning problems of this form are known to be NP-Hard [110]. Rather than attempt to find an optimal solution in possibly exponential time, we seek approximate solutions with bounded suboptimality that can be found efficiently in practice. Sequential planning techniques (Alg. 2) are common in active sensing and exploration [7, 34, 151]. In this chapter, we will refer to Alg. 2 as performing sequential greedy assignment (SGA). Here, SGA assigns plans to robots in sequence using a single-robot planner to maximize
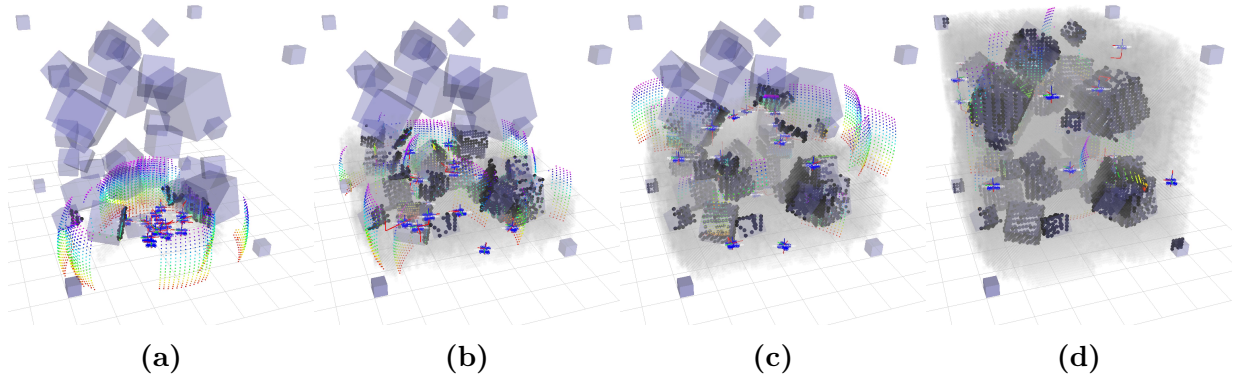
Figure 4.1: An example exploration experiment. A multi-robot team explores a three-dimensional environment cluttered with numerous obstacles (cubes) while using an online distributed planner. Known empty space is gray, and occupied space is black. Robots are shown in blue with red trajectories and obtain rainbow-colored point-cloud observations from their depth cameras. (a) The robots begin with randomized initial positions near a lower edge of the exploration environment which is bounded by a cube. (b) After entering the environment robots spread out to cover the bottom of the cubic environment (c) and then proceed upward to cover more of the volume. (d) Given enough time the robots explore the entire environment.

mutual information between a robot's future observations and the explored map given knowledge of plans assigned prior robots in the planning sequence. Recall that the properties of mutual information objectives ensure that techniques such as SGA achieve sub-optimality within half of optimal [69] and that suboptimal single-robot planners achieve similar bounds [169].

The sequential nature of SGA implies that planning time increases at least linearly with the number of robots and precludes online planning for large teams. Increasing computation time is especially relevant to exploration problems as new information about occupancy significantly affects both feasible and optimal plans and necessitates reactive planning to enable high rates of exploration. We propose a modified version of SGA, distributed sequential greedy assignment (DSGA), which consists of a fixed number of sequential planning rounds. At the beginning of each round, robots plan in parallel using a single-robot planner. A subset of those plans is chosen to minimize the difference between the information gain for the entire subset and the sum of the information gains for each robot individually, which does not consider redundancy between robots. We obtain a performance bound in terms of the result by Singh et al. [169] that explicitly describes the additional suboptimality due to parallel planning as the accrual of differences between actual and computed objective values during the planning process. In doing so, we reduce the distributed planning problem to selection of subsets of plans after each parallel planning phase that minimize these differences. Thus, we are able to take advantage of decoupled observations and distributed computation to improve computation times for online planning without compromising exploration performance.

However, robots may collide with each other and the environment, and aerial robots have non-trivial dynamics. Therefore, we also introduce inter-robot collision constraints and modify our algorithms to guarantee collision-free operation. We apply this approach

to teams of aerial robots and demonstrate collision-free exploration both in simulation and experimentally in a motion capture arena.

4.2   Multi-robot exploration formulation

This section describes the problem of distributed multi-robot exploration. We begin by describing the system and environment models and then introduce the planning problem as a finite-horizon optimization.

4.2.1   System model

Consider a team of robots, $\mathcal{R} = \{r_1, \ldots, r_{n_{\mathrm{r}}}\}$, engaged in exploration of some environment m. The dynamics and sensing are described by

$$x_t = f(x_{t-1}, u), \tag{4.1}$$

$$y_t = h(x_t, m) + \nu \tag{4.2}$$

where x_t represents a robot's state at time $t \in \mathbb{Z}_{\geq 0}$ and $u \in \mathcal{U}$ is a control input selected from the finite set $\mathcal{U}$. For aerial robots as studied here $x_t \in \mathbb{SE}(3)$. The observation, y_t, is a function of both the state and the environment and is corrupted by noise, ν. We use capital letters to refer to random variables and lowercase for realizations so M and Y_t represent random variables associated with the environment and an observation, respectively.

4.2.2   Occupancy grids and ranging measurements

The environment itself is unknown and can be with the random variable M which is modeled as an occupancy grid [64] and with an associated approximations of mutual information for ranging sensors [37, 103, 201]. This occupancy grid is in turn discretized into cells, $M = \{C_1, \ldots, C_{n_{\mathrm{m}}}\}$, that are either occupied or free with some probability. Cells occupancy is independent such that the probability of a realization m is $\mathbb{P}(M = m) = \prod_{i=1}^{n_{\mathrm{m}}} \mathbb{P}(C_i = c_i)$ where $c_i \in \{0, 1\}$. The conditional probability of M given previous states and observations is then written

$$\mathbb{P}(M = m | x_{1:T}, y_{1:T}) = \frac{\prod_{t=1}^{T} \mathbb{P}(y_t | M = m, x_t) \mathbb{P}(M = m)}{\sum_{m' \in \mathcal{M}} \prod_{t=1}^{T} \mathbb{P}(y_t | M = m', x_t) \mathbb{P}(M = m')} \tag{4.3}$$

where $\mathcal{M} \in \{0, 1\}^{n_{\mathrm{m}}}$ is the set of possible environments. As representing an unconstrained joint distribution between cells is intractable, the conditional probabilities of the cells given previous observations are also treated as being independent with probability $p_{i,t}$ such that the conditional probability is

$$\mathbb{P}(M = m | x_{1:T}, y_{1:T}) = \prod_{i=1}^{n_{\mathrm{m}}} p_{i,t} \tag{4.4}$$

We denote the collection of probabilities as the belief, $b_t = \bigcup_{i=1}^{n_\mathrm{m}} p_{i,t}$.

The robots considered in this work are equipped with depth cameras, a form of ranging sensor. Each sensor observation consists of a collection of range observations which each provide the distance along a ray from the sensor origin to the nearest occupied cell in the environment, subject to additive Gaussian noise. For more detail, we refer the reader to work by Charrow et al. [37].

4.2.3 Problem description and objective

For one robot and one time-step the optimal control input in terms of entropy reduction is

$$u_1^\star = \arg\max_{u \in \mathcal{U}} \ \mathbb{I}(Y_{t+1}; M | b_t, x_t). \tag{4.5}$$

subject to the dynamics (4.1) and observation model (4.2). Consider an l-step horizon. The problem becomes a belief-dependent, partially observable Markov decision process (POMDP) as is discussed in more detail by Lauri and Ritala [116] and is an optimization problem over policies. We instead optimize over a fixed series of actions (see Sec. 2.1.2) which results in a simpler problem. To simplify notation, let $\mathcal{Y}_i$ indicate the space of possible observations available to robot i over the finite horizon induced by the finite set of control inputs, the dynamics (4.1), and the observation model (4.2). The optimal multi-robot, finite-horizon informative plan is then

$$Y_{t+1:t+l,1:n_\mathrm{r}}^\star = \arg\max_{Y_{1:l,1:n_\mathrm{r}} \in \mathcal{Y}_{1:n_\mathrm{r}}} \ \mathbb{I}(Y_{1:l,1:n_\mathrm{r}}; M | b_t, x_{t,1:n_\mathrm{r}}) \tag{4.6}$$

where the indexing $x_{1:t,1:n_\mathrm{r}}$ represents values at times 1 through t and for robots 1 through n_r. In the following sections, we will drop the time and robot index as well as robot states and belief when appropriate.

Solutions to (4.6) will be constructed incrementally using greedy algorithms. Using the definitions in Sec. 3.5.2, the constraints in (4.6) can be interpreted as simple partition matroid where $\mathscr{U} = \bigcup_{i=1}^{n_\mathrm{r}} \mathcal{Y}_i$ and $\mathscr{I} = \{Y \subset \mathscr{U} \mid |Y \cap \mathcal{Y}_i| \leq 1, \ \forall i \in \mathcal{R}\}$.

4.2.4 Assumptions

We make the following assumptions regarding the exploration scenario: *1) all robots have the same belief state, operate synchronously, and communicate via a fully connected network; 2) incremental motions via f are bounded*; and *3) the sensor range is bounded*. The first assumption simplifies analysis in the context of this work. Here we emphasize scenarios where large numbers of robots operate in close proximity leading to redundant observations. Extending the proposed algorithm to incorporate additional considerations such as communication constraints is left to future work. The second and third assumptions ensure that the mutual information between observations made by distant robots is zero. These assumptions simplify the problem structure and are the key reason that the proposed efficient algorithm comes with little to no reduction in solution quality.

4.3 Single-robot planning

We employ Monte-Carlo tree search [27, 39] for the single-robot planner as previously proposed for active perception and exploration [18, 116, 144] and in multi-robot active perception [18].

In order to ensure bounded and similarly scaled rewards, constant terms from (4.6) are dropped when planning for the ith robot to obtain a local objective

$$\mathbb{I}(Y_{t+1:t+l,i}; M | Y_{t+1:t+l,A}), \tag{4.7}$$

the mutual information between $Y_{t+1:t+l,i}$ and the map conditional on observations $Y_{t+1:t+l,A}$ obtained by some set of robots A such that $A \cap \{i\} = \emptyset$.

Denote solutions obtained from the Monte-Carlo tree search single-robot planner maximizing (4.7) as

$$\hat{Y}_i = \text{SingleRobot}(i, Y_A) \tag{4.8}$$

and assume this planner has suboptimality $\eta \geq 1$, such that

$$\eta \mathbb{I}(M; \hat{Y}_i | Y_A) \geq \max_{Y \in \mathcal{Y}_i} \mathbb{I}(M; Y | Y_A) \tag{4.9}$$

as in to the approach of Singh et al. [169].

Although Monte-Carlo tree search does not come with a suboptimality guarantee, some existing algorithms for informative path planning [41, 169] do. Monte-Carlo tree search is used instead on account of limited computation time and later for ease in incorporation of inter-robot collision constraints which are time-varying.

4.4 Multi-robot planning

The main contribution of this work is the design and analysis of a new distributed multi-robot planner that extends the single-robot planner discussed in Sec. 4.3 or any planner satisfying (4.9) to multi-robot exploration. In development of a distributed algorithm, we first present a commonly used algorithm, SGA, which provides suboptimality guarantees [169] but requires robots to plan sequentially. We then propose a similar distributed algorithm, DSGA, and analyze its performance in terms of time and suboptimality.

4.4.1 Sequential greedy assignment

Consider an algorithm that plans for each robot in the team by maximizing (4.7) given all previously assigned plans and continues in this manner to sequentially assign plans to each robot. We will refer to this as sequential greedy assignment (SGA). Singh et al. [169] use the properties of mutual information discussed in Sec. 3.5.1 to establish that SGA obtains an objective value within $1 + \eta$ of the optimal solution. The greedy solution using an

Algorithm 3 Distributed sequential greedy assignment (DSGA) from the perspective of robot i

---

1: $n_\mathrm{d} \leftarrow$ number of planning rounds
2: $n_\mathrm{r} \leftarrow$ number of robots
3: $Y_F \leftarrow \emptyset$ ▷ set of fixed trajectories
4: **for** $1, \ldots, n_\mathrm{d}$ **do**
5:      $Y_{i|Y_F} \leftarrow \mathrm{SingleRobot}(i, Y_F)$
6:      $I_{i,0} \leftarrow \mathbb{I}(M; Y_{i|Y_F} | Y_F)$ ▷ planner reward
7:      $I_{i,F} \leftarrow I_{i,0}$ ▷ updated reward
8:      **for** $k = 1, \ldots, \left\lceil \frac{n_\mathrm{r}}{n_\mathrm{d}} \right\rceil$ **do**
9:          $j \leftarrow \underset{j \in 1:n_\mathrm{r}}{\arg\min}\, I_{j,0} - I_{j,F}$ ▷ computed by distributed reduction across robots
10:          **if** $i = j$ **then**
11:              Transmit: $Y_{i|Y_F}$
12:              **return** $Y_{i|Y_F}$
13:          **else**
14:              Receive: $Y_{j|Y_F}$
15:              $Y_F \leftarrow Y_F \cup Y_{j|Y_F}$
16:              $I_{i,F} \leftarrow \mathbb{I}(M; Y_{i|Y_F} | Y_F)$ ▷ update reward

---

optimal single-robot planner can be defined inductively as $Y^\mathrm{g} = Y^\mathrm{g}_{0:n_\mathrm{r}}$ using a suboptimal planner as in (4.8) to obtain the solution $Y^\mathrm{g} = Y^\mathrm{g}_{0:n_\mathrm{r}}$ such that

$$
\begin{aligned}
Y^\mathrm{g}_0 &= \emptyset \\
Y^\mathrm{g}_i &= \mathrm{SingleRobot}(i, Y^\mathrm{g}_{1:i-1})
\end{aligned}
\tag{4.10}
$$

This algorithm satisfies the following suboptimality bound.

Theorem 2 (Suboptimality bound of sequential assignment [169]). SGA *obtains a suboptimality bound of*

$$
\mathbb{I}(M; Y^\star) \leq (1 + \eta)\mathbb{I}(M; Y^\mathrm{g})
\tag{4.11}
$$

This multi-robot planner is formulated as an extension of a generic single-robot planner and depends only on the suboptimality of the single-robot planner. As robots plan sequentially, this leads to large computation times as the number of robots grows.

4.4.2   Distributed sequential greedy assignment

Consider a scenario with spatially distributed robots such that the mutual information between any observations obtainable within a finite horizon by any pair of robots is zero. The union of solutions obtained for individuals independently is then equivalent to a solution to the combinatorial problem over all robots, $Y^\star$. A weaker version of this idea applies such that if the plans returned for a subset of robots are conditionally independent, those plans are optimal over that subset of robots regardless of the inter-robot distances. The

distributed planner, DSGA, is designed according to this principle and allows all robots to plan at once and then selects a subset of those plans while minimizing suboptimality.

DSGA is defined in Alg. 3 from the perspective of robot i. Planning proceeds in a fixed number of rounds, n_{d} (line 4). Each round begins with a planning phase where each robot plans for itself given the set of plans that are assigned in previous rounds (line 5), stores the initial objective value, $I_{i,0}$ (line 6), and copies this to a variable, $I_{i,F}$ (line 7) that represents the updated value as more plans are assigned. The round ends with a selection phase (line 8) during which a subset of $\lceil \frac{n_{\mathrm{r}}}{n_{\mathrm{d}}} \rceil$ plans are assigned to robots. The plans are assigned greedily to minimize the decrease in the objective values, $I_{j,F} - I_{j,0}$, and the robot whose plan is to be assigned is selected using a distributed reduction across the multi-robot team (line 9) [114], and ties are broken arbitrarily. The chosen robot sends its plan to the other robots (line 11), and these robots store this plan (line 15) and update their objective values (line 16).

Denote a planner with n_{d} planning rounds as $\mathrm{DSGA}_{n_{\mathrm{d}}}$. Let D_i be the set of robots whose trajectories are assigned during the ith distributed planning round and let $F_i = \bigcup_{j=1}^{i} D_j$ be the set of all robots with trajectories assigned by that round. Denote solutions to this new distributed algorithm as Y^{d}. Then, let $\hat{Y}_{r|Y_{F_i}^{\mathrm{d}}}$ represent the approximate solution returned by the single-robot planner given previously assigned trajectories. The result of DSGA can then be written as $Y_{D_{i,j}}^{\mathrm{d}} = \hat{Y}_{D_{i,j}|Y_{F_{i-1}}^{\mathrm{d}}}$ where $D_{i,j}$ is the jth robot assigned during round i. DSGA achieves a bound related to Theorem 2 with an additive term based on the decrease in objective values from initial planning to assignment that DSGA seeks to minimize (Alg. 3, line 9).

Theorem 3. *The excess suboptimality of the distributed algorithm (Alg. 3) compared to the bound for greedy sequential assignment is given by the sum of conditional mutual informations between each selected observation and prior selections for the same planning round[1]*

$$\mathbb{I}(M;Y^{\star}) \leq (1+\eta)\mathbb{I}(M;Y^{\mathrm{d}}) + \eta\psi \tag{4.12}$$

where $\psi = \sum_{i=1}^{n_{\mathrm{d}}} \sum_{j=1}^{|D_i|} \mathbb{I}(Y_{D_{i,j}}^{\mathrm{d}}; Y_{D_{i,1:j-1}}^{\mathrm{d}} | Y_{F_{i-1}}^{\mathrm{d}})$ represents excess suboptimality. The proof is provided in Appendix B.3.

This is an online bound in the sense that it is parametrized by the planner solution. However, as will be shown in the results, ψ tends to be small in practice indicating that DSGA produces results comparable to SGA. In this sense, small values of ψ serve to certify the greedy bound of $1+\eta$ empirically without needing to obtain the objective value returned by SGA explicitly.

4.4.3 Worst-case suboptimality

The main focus of this work is to investigate how aspects of problem structure can be used to achieve efficient planning with minimal impact on suboptimality. This contrasts starkly

---

[1]Although we address multi-robot exploration, this result applies generally to informative planning problems and general monotone submodular maximization with partition matroid constraints (aside from notation and problem specialization).

to the worst-case suboptimality which is inversely proportional to the number of robots. While worst-case results by Gharesifard and Smith [75] and Mirzasoleiman et al. [133] are relevant, Grimsman et al. [82] recently proved a somewhat tighter bound that is readily applicable to Alg. 3.

Theorem 4 (Worst-case suboptimality). *The following worst-case bound for Alg. 3 holds:*

$$\mathbb{I}(M; Y^\star) \leq (1 + \eta \lceil n_{\mathrm{r}}/n_{\mathrm{d}} \rceil) \mathbb{I}(M; Y^{\mathrm{d}}). \tag{4.13}$$

The proof can be found in Appendix B.4.

According to this bound, the planner may entirely fail to take advantage of additional robots if additional planning rounds are not introduced. However, this does not account for locality which can lead distant robots to become decoupled in the solution. This bound can then be interpreted as a limiting scenario for when locality does not hold such as due to extremely close proximity or complex interactions between observations and the environment model.

4.4.4 Subset selection strategies

The subset selection step of DSGA itself uses a greedy strategy. Looking at this more directly, the negation of the contribution of a single round is

$$\mathbb{I}(M; Y^{\mathrm{d}}_{D_i} | Y^{\mathrm{d}}_{F_{i-1}}) - \sum_{j=1}^{|D_i|} \mathbb{I}(M; Y^{\mathrm{d}}_{D_{i,j}} | Y^{\mathrm{d}}_{F_{i-1}}), \tag{4.14}$$

found by application of the chain rule of mutual information (see Sec. 3.2) to (B.7). Equation (4.14) is submodular and monotonically decreasing unlike the monotonically increasing objectives considered previously. While we apply a heuristic approach and evaluate results empirically, other works have evaluated this setting more directly [74].

More generally, approaches that seek to minimize (4.14) over the course of individual rounds may fail for sufficiently unbalanced problems. For example, a large number of robots with zero contribution could cause all robots with non-zero objective values to be selected at once at significant detriment to solution quality. While such scenarios are not encountered in this work, this is an important direction for future work.

4.4.5 Algorithm run time analysis

We compare the run time of DSGA to SGA for variable numbers of robots with run time defined as the time elapsed from when the first robot begins computation until the last robot is finished. We assume point-to-point communication over a fully connected network requiring a fixed amount of time per message. The messages have fixed sizes and correspond either to a finite-horizon plan or a difference in mutual information. Given these assumptions, broadcast and reduction steps each require $O(\log n_{\mathrm{r}})$ time [114]. Let $\alpha(n_{\mathrm{r}}, b_t)$ bound the run time of the selected single-robot planner, noting the dependence on the number of robots and the environment. Similarly, let $\beta(n_{\mathrm{r}}, b_t)$ bound cost of the mutual information computation in Alg. 3, line 16.

SGA consists of n_r planning steps, each with one broadcast step. The computation time of sequential greedy assignment is then

$$\text{SGA}: \quad O(n_\text{r}\alpha(n_\text{r}, b_t) + n_\text{r}\log n_\text{r}). \tag{4.15}$$

Each of the n_d rounds in DSGA begins with a single planning phase, coming to a cost of $O(n_\text{d}\alpha(n_\text{r}, b_t))$. The rest of the algorithm consists of subset selection, broadcast of the chosen plans, and computation of mutual information which cumulatively occur once per robot for a total cost of

$$\text{DSGA}_{n_\text{d}}: \quad O(n_\text{d}\alpha(n_\text{r}, b_t) + n_\text{r}\beta(n_\text{r}, b_t) + n_\text{r}\log n_\text{r}). \tag{4.16}$$

If n_d grows slowly or is constant, the performance of DSGA relative to SGA depends on the relative cost of the mutual information computation (β).

For the approximation developed by Charrow et al. [37], evaluation of mutual information is linear in the number of cells of the map that are observed. Given the assumption of bounded sensor range, evaluation of mutual information scales linearly in the number of robots so that $\beta(n_\text{r}, b_t) \in O(n_\text{r})$. When planning time is dominated by a constant number of mutual information computation evaluations, such as when using a Monte-Carlo tree search planner with a fixed number of sample trajectories, then $\alpha(n_\text{r}, b_t) \in O(n_\text{r})$ so that both algorithms are quadratic. However, in practice, α includes a large constant factor leading to a significant speedup for DSGA_{n_d} compared to SGA. In general, α may also depend non-trivially on factors such as the length of the plan or the scale or complexity of the environment. This further emphasizes the significance of the of eliminating the n_r coefficient from the single-robot planning step.

4.5 Persistent safe exploration given vehicle dynamics

Up to this point, the analysis has considered maximization of the mutual information objective over a joint space of finite-horizon trajectories which has the form of a partition matroid constraint. However, inter-robot collision constraints cannot be modeled using a matroid, and further challenges arise if the planner can fail to provide a solution or if no solutions exist.

4.5.1 Independence systems, collision constraints, and suboptimality

After including inter-robot collision constraints, the space of feasible joint plans no longer has the structure of a partition matroid or even a general matroid. Consider, a trajectory that is free of collisions with one assignment of trajectories to a subset of robots. This trajectory is not necessarily free of collisions with another assignment of trajectories to the same subset of robots and neither can a robot in the subset be assigned an additional trajectory from the first assignment. This violates the exchange property for matroids (Def. 5). Instead the constraints can be modeled using an independence system (Def. 4),

given that subsets of collision-free trajectories are also collision-free. In general, greedy algorithms perform arbitrarily poorly for general independence systems [69]. However, we can recover a lower bound of $1/(\eta n_{\mathrm{r}})$ by maximizing mutual information using the first selection in each planning round which is otherwise under-determined because the differences in Alg. 3, line 9 are uniformly zero for the first plan selected in each round. The bound follows by observing that the maximal feasible subsets have cardinality of at most n_{r} and by applying suboptimality of the single-robot planner. This result will be used later during discussion of system liveness in Sec. 4.5.3.

Although suboptimality bounds change with the constraint structure, this is not necessarily indicative of typical performance in exploration. Mutual information objectives as used in exploration often favor configurations with large inter-robot distances that minimize redundancy in sensor observations. Robots are also often small compared to their sensor footprints which further mitigates detrimental effects of collision constraints.

4.5.2   Sufficient requirements for safety

Safety is interpreted as meaning that robots do not enter any occupied part of the environment and do not collide with each other, and each condition is implemented using thresholds on distances as discussed in more detail in Sec. 4.6. Safety is guaranteed by enforcing the invariant that the current joint plan is safe and terminates in a controlled invariant state[2] as is common in model predictive control [150]. Plans are executed in a receding-horizon fashion so that robots do not necessarily enter the planned invariant states whereas in the case of planner failure, one or more robots may enter invariant states and remain in those states until a feasible plan can be found.

Safety will be guaranteed under the following assumptions: *1) the initial state, at the start of the exploration process, is an invariant state*; and *2) states that are believed not to be in collision with the environment are free of collisions with the environment and remain so for all time.* The first assumption ensures safety by induction. The second prevents arbitrary changes in the conditions for safety. This latter assumption does not always hold in practice. Approaches, such as selectively relaxing constraints, can address this if necessary.

Assume that the team of robots is tracking a plan that is safe for all time (i.e. meets the stated requirements for all time). We modify Alg. 3 to only commit to a single-robot plan if the resulting joint plan that results from swapping the old single-robot plan with the new one is also safe for all time. The following constraints are also implemented in the single-robot planner: *1) plans terminate in an invariant state*; and *2) plans respect collision constraints with other robots and the environment according to the full and current joint plan at planning time.* The joint plan which results from replacing a robot's plan with the output of the single-robot planner may not be safe either due to failure to meet the above constraints or due to updates to other robots' plans. In this case, the system falls

---

[2]In the implementation, robots are required to come to a stop and trajectories are then temporally extended as necessary. Although these conditions can be relaxed, such as to invariant sets, there do not appear to be clear or significant benefits from doing so.

back to the robot's prior plan which is known to be safe. As a result, joint plan is always safe for all time.

4.5.3 Liveness in multi-robot exploration

Informally, liveness properties refer to guarantees that a system will make progress toward a goal. In exploration, a natural statement of liveness for some system state is that the multi-robot team will eventually select an action that reduces the entropy of the map. Although liveness is not guaranteed in general, the proposed system design does admit a limited liveness guarantee. This prevents scenarios such as when a small number of conflicting robots can create a persistent state of deadlock in the entire system.

Assume that some robot finds a feasible single-robot plan with a mutual information reward of at least ϵ. Then, using the results and modifications described in Sec. 4.5.1, the resulting joint plan also provides a mutual information reward of at least ϵ which in turn corresponds to ϵ entropy reduction in expectation. This condition guarantees liveness in exploration with high probability (i.e. barring infinite sequences of disinformative observations) so long as some robot can find an action that reduces the entropy of the map.

4.6 Results and discussion

The results are divided into two main parts. Section 4.6.1 does not incorporate inter-robot collisions and addresses scalability and performance of Alg. 3 as well as the suboptimality in terms of Theorem 3 in the intended context of submodular maximization over a partition matroid. Section 4.6.2 introduces inter-robot collisions and the application to a team of aerial vehicles both in simulation and on real hardware. These experiments incorporate the approach described in Sec. 4.5 in order to address additional challenges related to dynamics and collision prevention.

4.6.1 Exploration with large numbers of kinematic quadrotors

The proposed approach is first evaluated in a simplified scenario with a team of kinematic quadrotors, disregarding dynamics and inter-robot collisions, so the discussion in Sec. 4.5 does not come into play. Performance is evaluated in terms of planner objective values, computation time, and entropy reduction with respect to the map.

Remark. *Chapter 7 revisits the results from this chapter. While developing that work we found that relative transformations for control actions were being applied a second time before computing the information gain which affects the results in this chapter for the kinematic model (but not results for the dynamic model or with physical robots). Applying transforms twice has little effect for translations (as the field of view does not change much) but significant effect for rotations. Although we include the original results, note that fixing this error improved completion time (Sec. 7.6.2) by about 18% for that version of the exploration system and eliminated some aberrant behavior.*

4.6.1a   Exploration scenario

We test the exploration methodology in a confined and cluttered environment with obstacles (cubes) of various sizes and with robot positions initialized randomly near a lower edge as depicted in Fig. 4.1. The environment is bounded by a $6\,\text{m} \times 6\,\text{m} \times 6\,\text{m}$ cube, and the robots model this environment using a 3D occupancy grid with a $0.1\,\text{m}$ resolution. The confines and clutter ensure that robots remain proximal to each other. This leads to redundant observations and potential for suboptimal joint plans over the planning horizon. The experiments compare DSGA_1 through DSGA_3 (noting that DSGA_{n_d} represents Alg. 3 with n_d planning rounds) to SGA for 4, 8, 16, and 32 robots over 3000 iterations (time-steps). Tests evaluating exploration performance are each run twenty times each with randomized initializations. Experiments that evaluate computation time use specially-instrumented planner and single trial for each configuration.

4.6.1b   Implementation details

We evaluate the proposed algorithm in simulation and run experiments on a computer equipped with an Intel Xeon E5-2450 CPU (2.5 GHz). The planner and other components of the system are implemented using C++ and ROS [148]. When timing the distributed planner, planning steps that would normally be performed in parallel (single-robot planning during each planning round and information propagation) are executed serially. The duration of each such step is taken as the maximum duration of the serially computed steps in order to mimic distributed computation. In practice, computing the reduction to find the minimum excess term (Alg. 3, line 9) requires an insignificant amount of time. So, although the analysis assumes a logarithmic-time parallel, we compute this by iteration over all elements in the implementation.

The simulated robots emulate kinematic quadrotors moving in a three-dimensional environment with planning, mapping, and mutual information computation each performed in 3D. The single-robot Monte-Carlo tree search planner is run for a fixed number of samples (200) using a discrete set of actions consisting of the choice of translations of $\pm 0.3\,\text{m}$ in the x–y–z directions or heading changes of $\pm \pi/2\,\text{rad}$. Each robot is equipped with a simulated time-of-flight camera with a range of $2.4\,\text{m}$ which is similar to typical depth cameras and having a 19×12 resolution and a $43.6° \times 34.6°$ field of view which is oriented with the long axis aligned vertically to promote effective sweeping motions. For efficient evaluation of the mutual information objective, we substitute the approximation of Cauchy-Schwarz mutual information (CSQMI) described by Charrow [33] (which is not necessarily submodular) for the Shannon mutual information (3.5) and downsample rays by two in each direction. Rather than the uniform prior for cell occupancy (50%) which is commonly used in mapping, we introduce a prior of 12.5% probability of occupancy during evaluation of mutual information [179]. This encourages selection of actions that observe larger volumes of unknown space. This set of experiments is not run in real-time although the next is. The planner maintains the robot states internally and triggers the camera after each iteration.

(a) Information reward, $n_r = 4$

(b) ψ terms, $n_r = 4$

(c) Information reward, $n_r = 8$

(d) ψ terms, $n_r = 8$

(e) Information reward, $n_r = 16$

(f) ψ terms, $n_r = 16$

(g) Information reward, $n_r = 32$

(h) ψ terms, $n_r = 32$
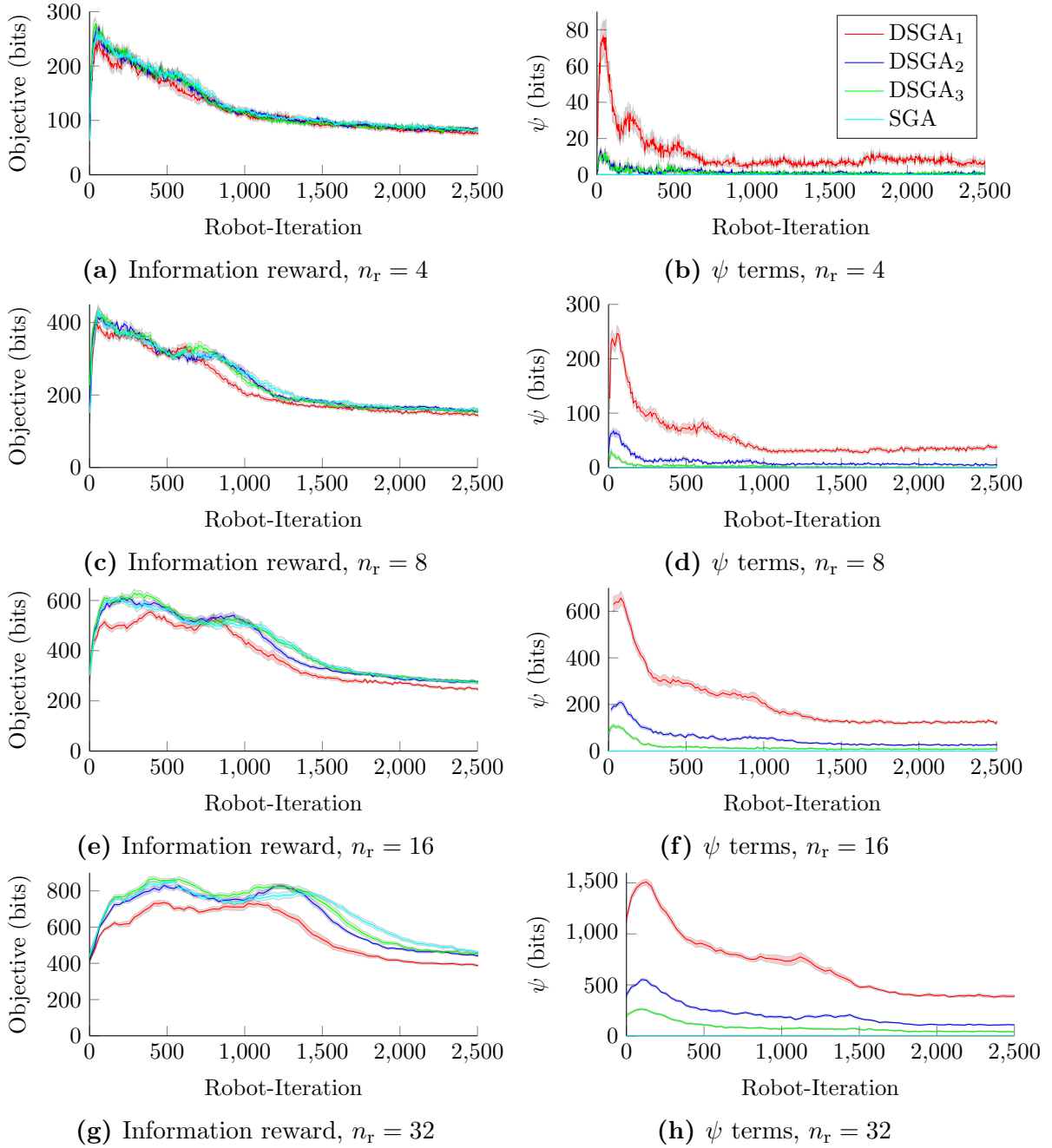
Figure 4.2: Objective and ψ values for varying numbers of robots and distributed planning rounds: (left) Mutual information objective values for $DSGA_2$ and $DSGA_3$ closely track SGA while the performance of $DSGA_1$ (which ignores inter-robot interactions) degrades with increasing numbers of robots. (right) Variations in the excess, ψ, terms correspond with the objective values—although ψ-values increase with the numbers of robots, they also decrease with increasing numbers of planning rounds and remain small compared to the objective for $DSGA_3$. Transparent patches indicate standard-error.

Alg.	n_r	Objective $\left(\frac{\text{bits}}{\text{robot}}\right)$		ψ $\left(\frac{\text{bits}}{\text{robot}}\right)$		Entropy red. rate $\left(\frac{\text{bits}}{\text{robot·iter.}}\right)$		Total entropy red. (bits)	
		avg.	std. dev.	avg.	std. dev.	avg.	std. dev.	avg.	std. dev.
DSGA_1	4	2.84e+01	1.29e+01	2.76e+00	3.90e+00	2.28e+02	4.83e+01	1.54e+05	1.21e+04
DSGA_2	4	2.99e+01	1.30e+01	3.39e-01	8.50e-01	2.36e+02	2.96e+01	1.61e+05	2.04e+03
DSGA_3	4	2.99e+01	1.34e+01	3.15e-01	8.75e-01	2.32e+02	2.31e+01	1.61e+05	2.41e+03
SGA	4	3.06e+01	1.34e+01	0.00e+00	0.00e+00	2.30e+02	3.68e+01	1.60e+05	2.85e+03
DSGA_1	8	2.63e+01	1.04e+01	6.49e+00	5.80e+00	2.07e+02	1.50e+01	1.60e+05	2.23e+03
DSGA_2	8	2.82e+01	1.08e+01	1.23e+00	1.66e+00	2.10e+02	1.78e+01	1.63e+05	1.89e+03
DSGA_3	8	2.81e+01	1.10e+01	2.67e-01	6.30e-01	2.04e+02	1.41e+01	1.64e+05	1.75e+03
SGA	8	2.84e+01	1.06e+01	0.00e+00	0.00e+00	2.05e+02	1.53e+01	1.64e+05	2.80e+03
DSGA_1	16	2.22e+01	7.47e+00	1.24e+01	8.14e+00	1.64e+02	1.11e+01	1.60e+05	1.77e+03
DSGA_2	16	2.46e+01	8.08e+00	3.04e+00	2.58e+00	1.77e+02	9.52e+00	1.66e+05	1.42e+03
DSGA_3	16	2.52e+01	8.17e+00	9.67e-01	1.30e+00	1.78e+02	8.97e+00	1.67e+05	1.67e+03
SGA	16	2.48e+01	7.86e+00	0.00e+00	0.00e+00	1.74e+02	7.60e+00	1.68e+05	1.65e+03
DSGA_1	32	1.69e+01	4.64e+00	2.00e+01	1.00e+01	1.26e+02	4.60e+00	1.61e+05	1.75e+03
DSGA_2	32	1.94e+01	4.99e+00	6.00e+00	3.59e+00	1.27e+02	5.45e+00	1.67e+05	1.44e+03
DSGA_3	32	2.02e+01	5.30e+00	2.44e+00	1.79e+00	1.29e+02	7.60e+00	1.69e+05	9.94e+02
SGA	32	2.03e+01	4.63e+00	0.00e+00	0.00e+00	1.31e+02	8.73e+00	1.72e+05	6.51e+02

Table 4.1: Exploration performance results for the kinematic exploration scenario: The reduction rate is computed with respect to the map over the first 250 robot-iterations and is representative of nominal performance. The total entropy reduction shows the entropy reduction with respect to the map at the end of the experimental trial (3000 robot-iterations).

4.6.1c   Evaluation of planner performance and objective values

Figure 4.2 and Table 4.1 show results for exploration experiments comparing DSGA_1 through DSGA_3 to SGA. The excess (ψ) is largest at the beginning of each exploration run as all robots are start near the same location. As the robots spread out, all planners approach approximately steady-state conditions in terms of both excess suboptimality and objective values before decaying once the environment is mostly explored. The ψ terms remain relatively large for DSGA_1—which assigns all plans in a single round and does not reason about conditional dependencies. However, the ψ terms decrease monotonically with increasing numbers of planning rounds. These values remain small for DSGA_3 and are approximately one-eighth of the objective value on average for 32 robots whereas values for DSGA_1 exceed the value of the objective for the same number of robots. Decreasing values of ψ are reflected in plots of the mutual information objective. Whereas DSGA_2 and DSGA_3 closely track SGA, objective values for DSGA_1 degrade with increasing numbers of robots.

When interpreting these results and entropy reduction results to be shown in Sec 4.6.1e, it is helpful to note that the scale of the environment remains constant and that increasing the number of robots serves to increase a notion of the density of the distribution of robots in the environment. In this sense, similar performance can be expected of larger teams operating in larger environments up to constraints on computation and communication. Toward this end, the next subsection evaluates computation times for DSGA and demonstrates that it scales much better than SGA in practice.

4.6.1d Computational performance

While planning performance is largely consistent, DSGA is able to take advantage of distributed computation in to provide significantly improved computation times (Fig. 4.3). Computation times for SGA scale approximately linearly in our simulation experiments. The average computation time for SGA increases more than seven times from $n_r = 4$ to $n_r = 32$ robots from $2.25\,\mathrm{s}$ to $16.8\,\mathrm{s}$ while times for DSGA remain small and decrease slowly. As a result, $DSGA_3$ provides a 2 to 8 times speedup compared to SGA. With 32 robots, the computation time increases by only a factor of 2.5 from $DSGA_1$ to $DSGA_3$, despite tripling the number of planning rounds. These results indicate that DSGA is able to provide multi-robot coordination for large numbers of robots while accommodating requirements for online performance well after doing the same with SGA becomes intractable.

4.6.1e Entropy reduction performance

Figure 4.4 compares DSGA to SGA for various numbers of robots and is summarized in Tab. 4.1. Exploration rates early in exploration runs remain consistent across planners and degrade by approximately 43% as the numbers of robots increase due to crowding. This translates to significant improvements in the total rate of exploration even as efficiency decreases.

The entropy reduction performance of $DSGA_2$ and $DSGA_3$ closely matches SGA for each number of robots. Interestingly, the performance of $DSGA_1$ is worst for $n_r = 4$ robots where the rate of entropy reduction degrades after initially performing well. For larger numbers of robots, differences in entropy reduction performance are most apparent in the total entropy reduction. This is likely a result of the planners involving inter-robot coordination being able to distribute robots across the environment more effectively. Because we use local finite-horizon planner, $DSGA_1$ then occasionally fails to allocate robots to and explore some portions of the environment. Given a spatially global planning strategy, longer exploration times could then be expected for $DSGA_1$ due to the cost of traveling to these unexplored regions of the environment.

4.6.2 Simulation and experiments of dynamic robots with inter-robot collision constraints

This section addresses more realistic systems and involves inter-robot collision avoidance and non-trivial dynamics in the form of teams of real and simulated quadrotors. Unlike the previous section, robots now track dynamic trajectories while planning is performed in real-time. Simulation results for a team of six robots demonstrate that DSGA retains performance comparable to SGA. Experiments using a three real quadrotors serve to ground the simulation results by demonstrating comparable results on a similarly configured system in practice.

Alg.	n_r	S.R. Planning		Prop.		Total	
		avg.	std.	avg.	std.	avg.	std.
$DSGA_1$	4	0.600	0.0603	0.0251	0.00529	0.625	0.0610
$DSGA_2$	4	1.19	0.0928	0.0181	0.00422	1.21	0.0934
$DSGA_3$	4	1.11	0.106	0.0189	0.00415	1.13	0.107
SGA	4	2.23	0.150	0.0184	0.00253	2.25	0.151
$DSGA_1$	8	0.630	0.0562	0.0524	0.00664	0.682	0.0577
$DSGA_2$	8	1.22	0.0863	0.0491	0.00696	1.27	0.0886
$DSGA_3$	8	1.82	0.105	0.0428	0.00708	1.86	0.108
SGA	8	4.22	0.243	0.0342	0.00381	4.25	0.245
$DSGA_1$	16	0.609	0.0565	0.106	0.00916	0.715	0.0616
$DSGA_2$	16	1.28	0.0867	0.106	0.00954	1.38	0.0912
$DSGA_3$	16	1.91	0.0911	0.106	0.00985	2.02	0.0923
SGA	16	8.55	0.329	0.0667	0.00543	8.63	0.332
$DSGA_1$	32	0.649	0.0565	0.217	0.0105	0.865	0.0596
$DSGA_2$	32	1.30	0.0547	0.222	0.0110	1.53	0.0583
$DSGA_3$	32	1.92	0.0949	0.218	0.0124	2.14	0.101
SGA	32	16.6	0.592	0.131	0.00816	16.8	0.596

(a) Table of timing data
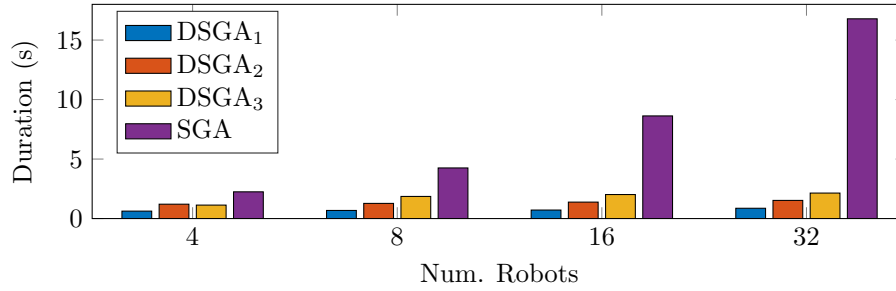
(b) Timing for SGA and $DSGA_3$

Figure 4.3: Computational performance (seconds) in terms of total computation time (time elapsed from when the first robot starts planning until the last robot stops). (a) Time per iteration spent in the single-robot planner, propagation of the information reward (DSGA only), and total computation time. (b) Comparison of the timing differences between SGA and DSGA.

(a) Information gain per robot ($n_r = 4$)

(b) Information gain per robot ($n_r = 8$)

(c) Information gain per robot ($n_r = 16$)

(d) Information gain per robot ($n_r = 32$)
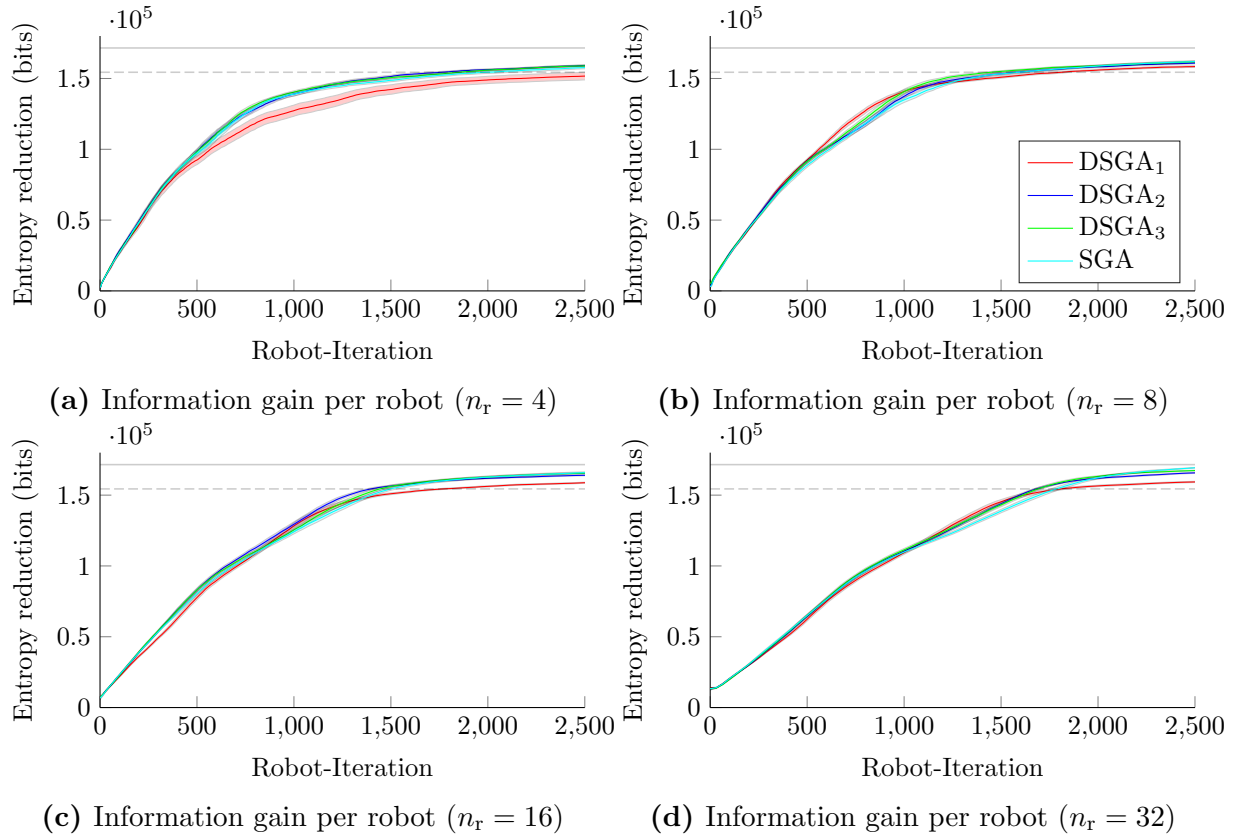
Figure 4.4: Entropy reduction performance with different numbers of robots and planner configurations: $DSGA_2$ and $DSGA_3$ closely track SGA in all cases while $DSGA_1$, which does not incorporate inter-robot coordination, performs worst for $n_r = 4$ and has reduced total entropy reduction for $n_r = 16$ and $n_r = 32$. Transparent patches show standard-error. Gray lines indicate the maximum mean entropy reduction of $1.72 \cdot 10^5$ bits (data continues through 3000 robot-iterations) and a 90% threshold for task completion which facilitates approximate comparison to results in Chapter 7.

4.6.2a Implementation details

As these experiments now involve inter-robot collision constraints, we now incorporate the details Sec. 4.5 into both the SGA and DSGA planners. Rather than discrete steps, the Monte-Carlo tree search planner now uses a library of polynomial motion primitives similar to that described by Tabib et al. [179] to provide actions. Because trajectories are executed in real-time, the Monte-Carlo tree search planner is run in a fully any-time fashion at a rate of $1/3\,$Hz rather than for a fixed number of samples, and time in single-robot planning is budgeted according to the number of sequential planning steps in the given multi-robot planner. The planner uses a finite $6\,$s horizon with motion primitives whose durations vary from $0.5\,$s to $10\,$s (2s is typical), and at each planning step as many as 162 motion-primitive actions are available although some may be infeasible due to collision constraints. Robot-environment collision checks are implemented use a truncated distance field for efficient lookups, and inter-robot collision checks use horizontal distances to prevent any robot from flying in another's downwash. This set of experiments uses a custom and simulation and control framework that implements a standard quadrotor dynamics model and a quadrotor controller that tracks polynomial trajectories in the differentially-flat outputs [125] and a team of custom quadrotors equipped with Structure brand depth sensors.

4.6.2b Simulation results

The simulation trials (Fig. 4.5) evaluate a team of six robots exploring a pseudo-planar rearrangement of the environment shown in Figure 4.1 using sequential planning (SGA), myopic planning (DSGA$_1$), and distributed planning with three rounds (DSGA$_3$) with twenty trials per each planner. Single-robot planning steps (Monte-Carlo tree search) were run in parallel on the same computer as in Sec 4.6.1b.

The simulation environment and results for entropy reduction are evident shown in Fig. 4.5. The exploration performance is largely similar across planners except that DSGA$_1$ exhibits a slight delay at the start due to a tendency for robots to choose conflicting trajectories. Because robots are initialized near each other, toward the center of the environment, (with blue lines marking the locations where robots took off) the single-robot planner returns plans that would result in inter-robot collisions most often early in each trial. This is evident in Fig. 4.6 which plots the cumulative number of times that the multi-robot planner has had to fallback to the existing safe trajectory due to either inter-robot collisions or failure of single-robot planning as described in Sec 4.5.2. This is most pronounced for DSGA$_1$ due to the lack of coordination in the planner. Using the fallback trajectories is most detrimental at the very beginning of the trial when the fallback consists of staying stationary at the starting position, causing the delay. Later, once they have been populated with prior planning results, resorting to fallback trajectories becomes less significant.

4.6.2c Experimental results

An experimental trial with a team of three quadrotors is also included as depicted in Fig. 4.7. This experiment serves to ground the simulation results described in the prior subsections on a real system. Due to the small size of the team, only SGA is used for
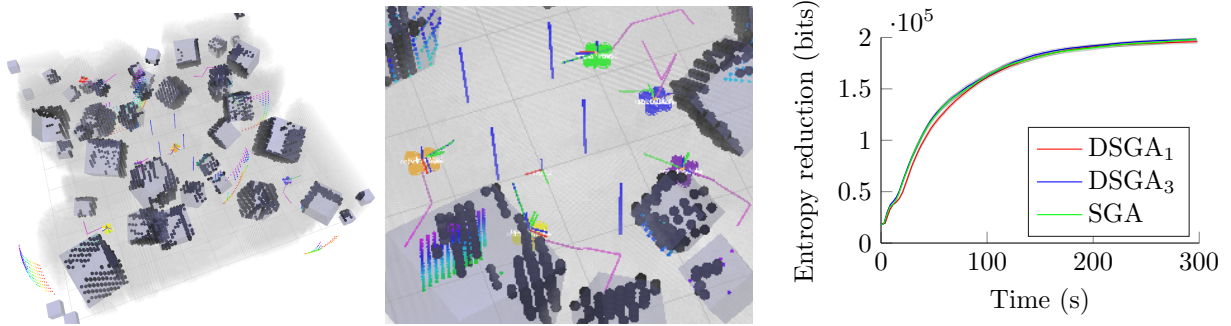
Figure 4.5: (left) Six robots explore a pseudo-planar environment using motion-primitive trajectories. (middle) Robots (such as the green and blue) plan informative and collision-free trajectories using $DSGA_3$ despite operating in close proximity. The portion of each robot's receding-horizon plan that is currently being executed is colored in green, and the rest of the horizon is magenta. (right) Planners perform similarly in terms of entropy reduction although the initial performance of $DSGA_1$ is impaired as it frequently resorts to using fallback trajectories due to conflicts in planned trajectories.
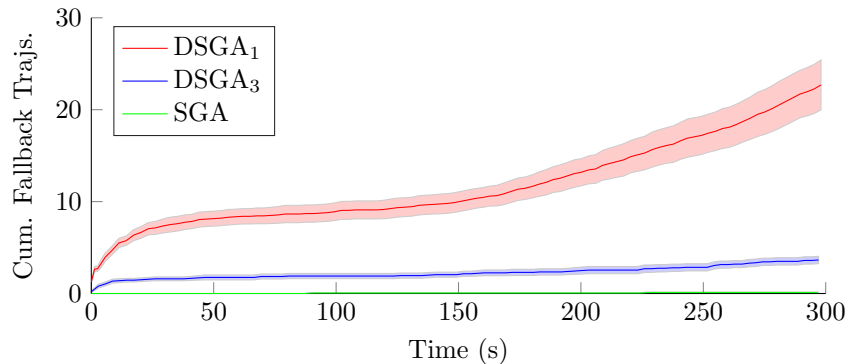
Figure 4.6: Cumulative numbers of fallback trajectories selected by multi-robot planner in simulation results with dynamic quadrotors: The multi-robot planners resort to fallback trajectories when the single-robot planner fails or returns a plan that would result in inter-robot collisions. $DSGA_1$ uses fallback trajectories frequently early in trials when robots are near each other and later after most of the environment has been explored.
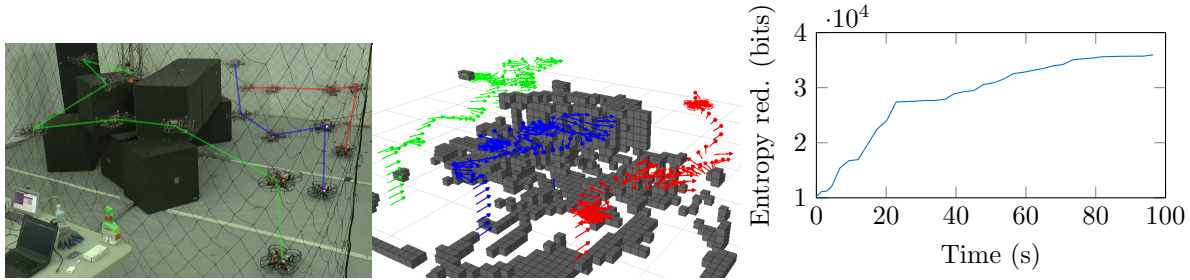
Figure 4.7: (left) A team of three robots explores a motion capture arena occupied by a geometric object consisting of foam boxes using Monte-Carlo tree search and motion-primitive trajectories and coordination via SGA and (middle) produce a voxel grid map of the environment (shown from a different vantage-point to also detail the robot in red which explores behind the obstacle). (right) The entropy of the map decreases as the robots move through the environment.

coordination as DSGA would be expected to provide little benefit (DSGA$_3$ is essentially equivalent to SGA for three robots). Planning is performed offboard on a laptop with a Intel i7-5600U CPU (2.6 GHz). Otherwise, the system configuration is identical to prior set of simulation results. Whereas the simulation results demonstrate application of the approach described in Sec. 4.5 for exploration with collision constraints and dynamic robots using a significant number of robots, these hardware results connect the implementation and its configuration to a real system.

The experiment is performed in a motion capture flight arena populated with a pile of foam blocks of various shapes and sizes. During the course of the exploration trial, the team of robots is able to successfully navigate and map the environment as evident in the trajectories and cumulative entropy reduction. The rate of entropy reduction per robot at the beginning of the trial is similar to Fig. 4.7 at approximately $400 \frac{\text{bits}}{\text{s·robot}}$. While the simulation results demonstrate efficiency of SGAand DSGAwith and without collision constraints these experimental results demonstrate that the planner design and configuration is also applicable to exploration with a real multi-robot team and performs similarly.

4.7 Conclusions and future work

The proposed distributed algorithm (DSGA) efficiently approximates sequential greedy assignment (SGA) and is appropriate for implementation on large multi-robot teams using online planning. We have applied this algorithm to the problem of multi-robot exploration, and demonstrated consistent entropy reduction performance in simulation for large numbers of robots exploring and mapping a complex three-dimensional environment. The results demonstrate that DSGA is able to effectively take advantage of parallel computation without degradation in solution quality at scales where application of SGA becomes intractable. We expect that this result will be instrumental in development of physical multi-robot systems that take advantage of distributed computation for exploration and other online informative planning problems.

In order to address realistic scenarios, we incorporated robot dynamics and inter-robot

collision constraints into the problem formulation. Although collision constraints result in increased problem complexity, we were able to modify the planning approach to guarantee safety and liveness under weak assumptions. The simulation results demonstrated consistent exploration performance and indicate that collision constraints have a relatively minor impact on exploration performance. Further, including collision constraints makes this approach viable for application to real robots, and we demonstrated this on a team of aerial robots flying in a motion capture arena.

Although DSGA can take advantage of parallel computation, the actual assignments in the subset selection step are fully sequential, and the asymptotic computation time is identical to sequential greedy assignment even though we obtained significant improvements in practice. The bound on suboptimality (Theorem 3) also only provides a post-hoc guarantee, and optimization performance *could* degrade for large numbers of robots (Theorem 4). Instead, the RSP methods that we introduce in the following chapters eliminate all sequential computation involving the entire multi-robot team and obtain strong suboptimality guarantees for any number of robots.

Chapter 5

Scalable and Near-Optimal Planning for Multi-Agent Sensor Coverage

Many objective functions that arise in sensor planning problems such as mutual information objectives [169] (and as in Chapter 4), objectives for sensing in hazardous environments [102], and various notions of area, set, and sensor coverage [153] are submodular. Intuitively, submodularity implies diminishing returns when constructing sets of sensing actions. This work explores multi-agent planning problems with submodular objective functions and especially variants of set and sensor coverage. We emphasize problems that feature networks of large or even unspecified numbers of agents seeking to maximize a global submodular objective, and the planners we propose obtain constant-factor performance under mild conditions such as for agents with limited sensor range and spatially local sensing actions.

Chapter 4 began to address the challenge that sequential planning scales poorly as the number of agents increases. Further, solving agents' local planning subproblems can be time-consuming on its own as the space of actions may be nearly infinite [169]. At the same time, we have already seen that dynamic environments and beliefs motivate real-time planning so that efficient multi-agent coordination is critical when scaling to large numbers of agents. We address this issue by proposing efficient distributed planners that consist of fixed numbers of sequential planning steps and approach existing constant-factor performance bounds (in expectation) when known pairwise interactions between agents are proportional to objective values.

Several recent works [75, 82] also address the core challenge of this work: design of parallel variants of sequential planners for multi-agent systems. Gharesifard and Smith [75] define a class of distributed planners based on directed acyclic graphs where agents perform greedy planning steps using only a subset of the decisions made by prior agents. Although they provide worst-case bounds on suboptimality, Grimsman et al. [82] provide tighter results for the same framework. However, both works obtain bounds that are inversely proportional to the maximum number of agents that can plan in parallel. In contrast, Chapter 4 demonstrated that such planners can be effective when agents that plan in parallel can find sets of decoupled actions. However, that approach also only provided post-hoc bounds and did not scale to arbitrary numbers of agents as some steps

(a) Area coverage   **(b)** Inter-agent redundancy

$$\max_{i,j} \text{overlap}(i,j)$$
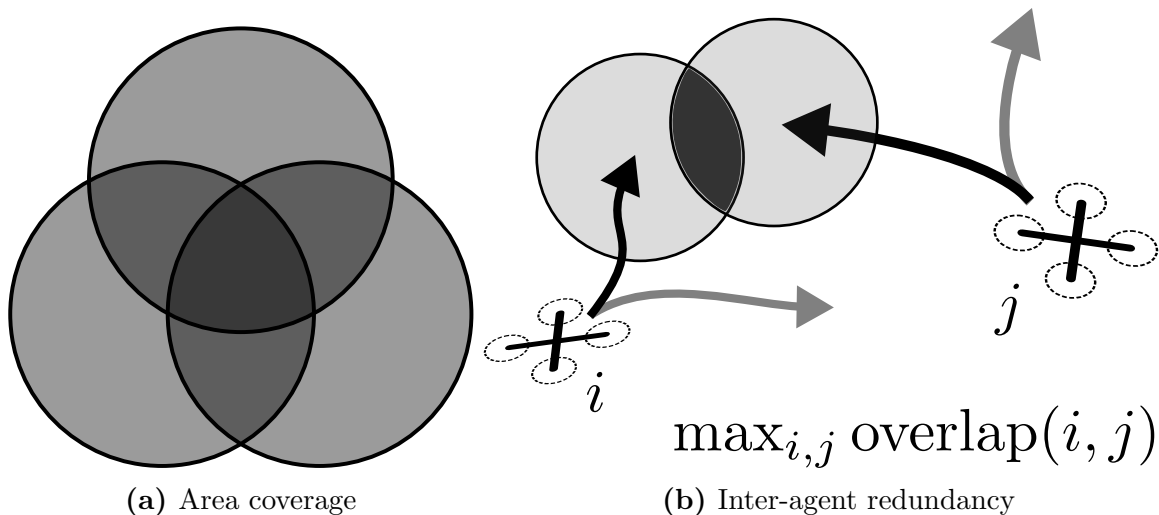
Figure 5.1: Consider a team of robots maximizing (a) sensor area coverage. Intuitively, distant agents may make decisions independently with no loss in optimality. We exploit such conditions to enable efficient distributed planning. Specifically, (b) inter-agent redundancy quantifies the maximum coupling between two agents. The distributed algorithm that we propose requires planning time that is independent of the number of agents and guarantees near-optimal performance when redundancy decreases monotonically subject to prior observations as is true for coverage objectives.

remained fully sequential.

We continue in the direction of the previous chapter by seeking to develop efficient distributed planners that exploit problem structure. As discussed in Fig. 5.1, our approach is inspired by the intuition that decisions by distant agents are often decoupled. We model this notion with a concept of inter-agent redundancy which describes how much one agent's marginal gain can decrease as a result of ignoring another agent. Our approach requires such redundancy to decrease monotonically in the presence of actions selected by previous agents in the planning sequence. This condition is equivalent to requiring the objective to satisfy the next higher-order monotonicity property after submodularity.

Together, these properties enable use of pairwise redundancy to bound the effect of ignoring an agent at any step of the planning process which relates total redundancy to suboptimality. We propose an algorithm—Randomized Sequential Partitions (RSP)—that randomly partitions agents and obtains a constant-factor bound when the optimal solution is proportional to the cumulative pairwise redundancy between all agents. This condition is generally satisfied by problems involving limited sensing range and distributions of agents with bounded density, and the approach further admits features such as local adaptation and limits on communication range. We refer to the latter case as Range-limited RSP(or R-lRSP). Finally, we prove that a generalized variant of weighted set coverage satisfies higher-order monotonicity conditions and provide simulation results for two cases, area coverage and a probabilistic detection scenario.

This chapter originally appeared in [55].

70

5.1 Background discussion

5.1.1 Properties of set functions and the 3-increasing condition

Consider a set function $g : 2^{\mathscr{U}} \to \mathbb{R}$ where $\mathscr{U}$ is the ground set. Just as for the previous chapter, the functions we study in this one are *normalized, monotonically increasing*, and *submodular* according to the definitions in Sec. 3.5.1.

We now introduce the next higher-order monotonicity property (see Sec. 3.5.1d) [70]: the objectives in this chapter are 3-increasing.[1] Given the definition of the derivative of a set function (3.16) and for $A \subseteq B \subseteq \mathscr{U}$ and disjoint subsets $X, Y \subseteq \mathscr{U} \setminus B$, 3-increasing functions satisfy

$$g(X; Y|A) \leq g(X; Y|B). \tag{5.1}$$

When expanded and negated, this expression takes the form $g(X|B) - g(X|B, Y) \leq g(X|A) - g(X|A, Y)$. Given that $A \subseteq B$, this can be interpreted as stating that conditioning reduces redundancy, and expressions of the form $-g(A; C) = g(A) - g(A|C)$ will be referred to as expressing the pairwise redundancy between A and C. Such higher-order monotonicity properties have not been used extensively in the literature on optimization of submodular functions although a few works study the same and similar properties [44, 70, 106, 149].

Weighted set cover is an example of a submodular function that is 3-increasing, and such objectives have been studied extensively and used to prove hardness results for submodular maximization [66] and tightness results for distributed planning [82]. Because existing results already use functions that are 3-increasing, requiring this condition does not impact hardness of an optimization problem. At the same time, some common submodular objectives are not necessarily 3-increasing. Figure 5.2 describes one such scenario for a submodular mutual information objective.

5.1.2 Partition matroids

Recall from Sec. 3.5.2 that *partition matroids* describe multi-agent problems where the joint action space is a product of local action spaces for each agent. The ground set of the partition matroid $(\mathscr{U}, \mathscr{I})$ is partitioned by a set of blocks $\{\mathscr{B}_1, \ldots, \mathscr{B}_n\}$, and the admissible sets of actions are $\mathscr{I} = \{X \subseteq \mathscr{U} \mid |X \cap \mathscr{B}_i| \leq \ell_i\}$ for $\ell_i \geq 0$.

5.2 Problem statement

Consider a multi-agent planning problem with agents $\mathcal{A} = \{1, \ldots, n_{\mathrm{a}}\}$ where each agent $i \in \mathcal{A}$ is associated with a set of actions $\mathscr{B}_i$ which is also a block of the partition matroid. Agents may select at most one action so that $|X \cap \mathscr{B}_i| \leq 1$ for each admissible set of

---

[1]Previously, we referred to the 3-increasing condition as *supermodularity of conditioning* [55]. Wang et al. [191] also refer to the combination of this condition with submodularity as "strong submodularity."
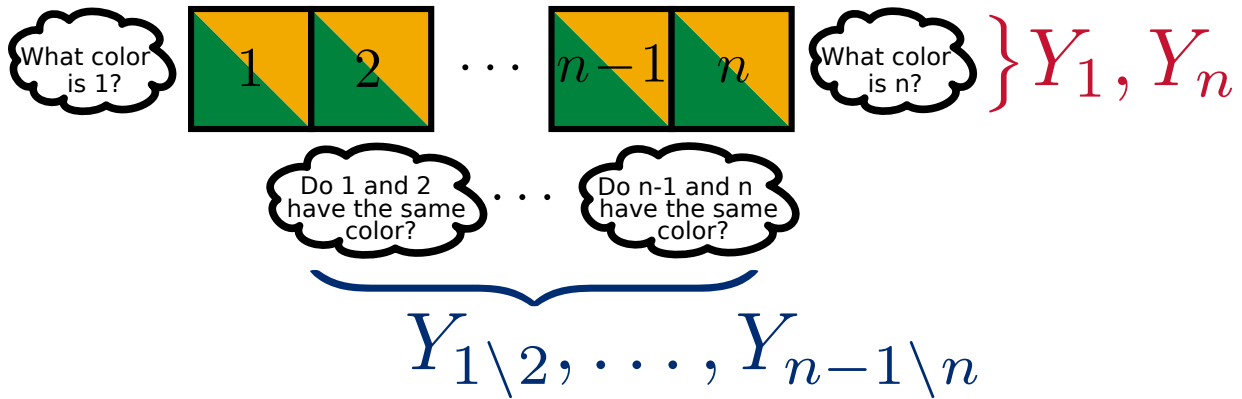
Figure 5.2: This illustration depicts an example of a submodular objective that is not 3-increasing and generalizes common examples for when mutual information increases under conditioning [58]. In this example, boxes in the set $C = \{1, \ldots, n\}$ are colored green or gold independently and with equal probability. Sensors may observe either end directly $\{Y_1, Y_n\}$ and differences colors of adjacent boxes $\{Y_{1\backslash 2}, \ldots, Y_{n-1\backslash n}\}$, to obtain a submodular mutual information reward $g(Y) = \mathbb{I}(C; Y)$ where Y is a set of such observations. Each individual observation provides one bit of information, and pairs obtain two bits and have no redundancy. This trend only changes once we consider all of the observations together: *we can determine the color of the last box in the sequence by observing the first box and each subsequent change in color.* This violates the 3-increasing condition (5.1) as the addition of last observation signifies an increase in redundancy and equivalently a decrease in the second derivative of the objective. That is $g(Y_1; Y_n) = 0$ but $g(Y_1; Y_n | Y_{1\backslash 2}, \ldots, Y_{n-1\backslash n}) = -1$.

actions $X \in \mathscr{I}$. Further, the agents are engaged in a sensing task with an objective g that satisfies the conditions outlined in Sec. 5.1.1 and so seek to solve

$$X^\star \in \arg\max_{X \in \mathscr{I}} g(X). \tag{5.2}$$

Previously, Sec. 3.5.3c described the local greedy heuristic (sequential maximization, Alg. 2) which obtains approximate solutions by recursively applying a greedy maximization step. Further, solutions via this approach are guaranteed to be within half of optimal.

5.3 Greedy planning on directed acyclic graphs

Sequential planning (Alg. 2) on a large network of agents is time-consuming as each agent must wait to receive the incremental solution from the previous agents before beginning computation. Gharesifard and Smith [75] propose a related class of planners where agents may ignore the decisions of previous agents according to a directed acyclic graph. Rather than planning with respect to all prior decisions as in Alg. 2, these planners obtain the solution $X^{\mathrm{d}} = \{x_1^{\mathrm{d}}, \ldots, x_{n_\mathrm{a}}^{\mathrm{d}}\}$ by evaluating

$$x_i^{\mathrm{d}} \in \arg\max_{x \in \mathscr{B}_i} g(x | X_{\mathcal{N}_i}^{\mathrm{d}}) \tag{5.3}$$

using incremental solutions from $\mathcal{N}_i \subseteq \{1, \ldots, i-1\}$, the set of in-neighbors of agent i in the directed acyclic graph. This model can be used to design distributed planners where

distant agents do not communicate or where subsets of agents execute their planning steps in parallel. Prior works studying such planners examine worst-case behavior for objectives that are submodular and monotonic [75, 82]. These fail to obtain constant-factor suboptimality when given only a fixed number of sequential planning steps but an arbitrary number of agents. Instead, this work examines sufficient conditions for a distributed planner with a fixed number of sequential planning steps to approach the performance of a sequential planner (half of optimal, see Sec 3.5.3c) We begin by analyzing (5.3) based on the redundancy of sensing actions between pairs of agents.

5.4 Analysis using inter-agent redundancy

The performance of the distributed planner will be analyzed by bounding decreases in marginal gains due to failure to condition on choices by prior agents i.e. $g(x_i^{\mathrm{d}}|X_{\mathcal{N}_i}^{\mathrm{d}}) - g(x_i^{\mathrm{d}}|X_{1:i-1}^{\mathrm{d}})$. The 3-increasing condition enables derivation of bounds on such changes in marginal gains using pairwise redundancies between elements.

Define the *inter-agent redundancy graph* as a weighted, undirected graph $\mathcal{G} = (\mathcal{A}, \mathcal{E}, \mathcal{W})$ with agents as vertices, edges $\mathcal{E} = \{(i,j) \mid i, j \in \mathcal{A}, i \neq j\}$,[2] and weights

$$\mathcal{W}(i,j) = w_{ij} = \max_{x_i \in \mathscr{B}_i, \, x_j \in \mathscr{B}_j} -g(x_i; x_j). \tag{5.4}$$

This connects the notion of redundancy to the multi-agent planning problem via maximum (and undirected) inter-agent redundancies.

Decreases in marginal gains can be bounded using the pairwise redundancies from the inter-agent redundancy graph using the following lemma.

Lemma 5 (Pairwise redundancy bound). *Consider disjoint subsets $A, B, C \subseteq \mathcal{U}$. Then*

$$g(A|B,C) - g(A|C) \geq \sum_{b \in B} g(A; b) \tag{5.5}$$

Proof. Lemma 5 follows from the chain rule (Lemma 1) and g being 3-increasing. Given an ordering $B = \{b_i, \ldots, b_{|B|}\}$, construct a telescoping sum

$$\begin{aligned}
g(A|B,C) - g(A|C) &= g(B|A,C) - g(A|C) \\
&= \sum_{i=1}^{|B|} g(b_i|B_{1:i-1}, A, C) - g(b_i|B_{1:i-1}, C) \\
&= \sum_{i=1}^{|B|} g(b_i; A|B_{1:i-1}, C)
\end{aligned}$$

Then, (5.5) follows because g is 3-increasing (5.1). Finally, note that, although the expression for (5.5) is tailored to its usage, the left hand side has the form of a second derivative of g so that $g(A; B|C) = g(A|B,C) - g(A|C)$. ∎

[2]Being undirected, (i,j) and (j,i) are the same edge.

Although produced independently, the above lemma is equivalent to [191, Lemma 1].

The total weight of the redundancy graph will characterize suboptimality for our approach. We will refer to a problem defined according to (5.2) as *α-redundant* for $\alpha > 0$ if

$$\alpha g(X^\star) \geq \sum_{(i,j)\in\mathcal{E}} w_{ij}. \tag{5.6}$$

Instances of (5.2) with finite objective values and numbers of agents are all α-redundant for some α although specific values are not guaranteed in general. We will use α-redundancy to absorb additive terms proportional to graph weights into constant-factor multiplicative bounds in terms of α.

5.4.1 Analysis of distributed planners using inter-agent redundancy

The inter-agent redundancy graph defined in the previous section can be applied in the analysis of distributed planners (5.3) using a similar approach as in the previous chapter (Chapter 4). Let $\hat{\mathcal{N}}_i = \{1, \ldots, i-1\} \setminus \mathcal{N}_i$ be the set of preceding agents that are ignored at step i of the assignment process according to (5.3) so that the set of all deleted edges is $\tilde{\mathcal{E}} = \{(i,j) \mid i \in \mathcal{A}, \ j \in \hat{\mathcal{N}}_i\}$. Then the planner suboptimality can be bounded in terms of the weights of these deleted edges.

Theorem 6 (Suboptimality of distributed planning). *The suboptimality of a planner obeying* (5.3) *can be bounded using the cumulative weight of deleted edges as*

$$g(X^\star) \leq 2g(X^\mathrm{d}) + \sum_{(i,j)\in\tilde{\mathcal{E}}} w_{ij}. \tag{5.7}$$

Proof. Theorem 6 follows from application of pairwise redundancy on the inter-agent redundancy graph to the standard proof technique for sequential maximization,

$$
\begin{aligned}
g(X^\star) &\leq g(X^\star, X^\mathrm{d}) \\
&= g(X^\mathrm{d}) + \sum_{i=1}^{n_\mathrm{a}} g(x_i^\star | X^\mathrm{d}, X_{1:i-1}^\star) \\
&\leq g(X^\mathrm{d}) + \sum_{i=1}^{n_\mathrm{a}} g(x_i^\star | X_{\mathcal{N}_i}^\mathrm{d}) \\
&\leq g(X^\mathrm{d}) + \sum_{i=1}^{n_\mathrm{a}} g(x_i^\mathrm{d} | X_{\mathcal{N}_i}^\mathrm{d}) \\
&= 2g(X^\mathrm{d}) + \sum_{i=1}^{n_\mathrm{a}} \left(g(x_i^\mathrm{d} | X_{\mathcal{N}_i}^\mathrm{d}) - g(x_i^\mathrm{d} | X_{1:i-1}^\mathrm{d}) \right).
\end{aligned}
\tag{5.8}
$$

The first equality is a telescoping sum. The inequalities follow first from g being monotonically increasing, second from submodularity, and third by greedy choice according to (5.3). The last equality is another telescoping sum. The main result (5.7) follows from application of (5.5) and (5.4) to the sum in (5.8) and the definition of $\tilde{\mathcal{E}}$. ∎

5.5 Randomized distributed planners

Let us now apply the analysis from the previous section to design of randomized distributed planners. A set of agents can execute planning steps in parallel if no pair of agents in the set shares an edge in the planner model (5.3). Considering this, we construct distributed planners by partitioning the agents and eliminating edges within blocks of the partition and then bound suboptimality for randomly assigned partitions. Finally, we present conditions for such planners to scale to an arbitrary number of agents and to admit features such as limited communication range.

5.5.1 Distributed planning on partitioned agents

Consider a planner where subsets of agents plan in parallel according to a partition $\{D_1, \ldots, D_{n_\mathrm{d}}\}$ of agents $\mathcal{A} = \cup_{i=1}^{n_\mathrm{d}} D_i$. In such a planner, n_d corresponds to the maximum number of sequential planning steps. Let d_i map each agent i to its block in the partition so that $i \in D_{d_i}$, and let the total ordering of agents respect a partial ordering induced by ordering the blocks of the partition so that $i < j$ implies $d_i \leq d_j$. We construct a planner (5.3) from the partition and ordering of agents and blocks by eliminating neighbors that share the same block from the complete directed acyclic graph

$$\mathcal{N}_i = \{1, \ldots, i-1\} \setminus D_{d_i}, \quad \hat{\mathcal{N}}_i = \{1, \ldots, i-1\} \cap D_{d_i}. \tag{5.9}$$

Ideally, the partition would minimize the cumulative weight of edges eliminated in the subgraphs of the blocks. However, that is equivalent to maximizing the weight of edges outside of the subgraphs which is the Max k-Cut problem on the inter-agent redundancy graph. Finding exact solutions is intractable because Max k-Cut is NP-Complete [105]. Therefore, the next section proposes randomized approaches that produce approximate solutions.

5.5.2 Planning with random partitions

As observed by Andersson [3], a random partition obtains a trivial $\frac{n_\mathrm{d}-1}{n_\mathrm{d}}$ by observing that edges are removed uniformly at random. The approach presented here is similar and is presented from the perspective of individual agents.

Consider a distributed planner as defined by (5.3) where agents share partial solutions with neighbors given a partition of the agents as in (5.9). Let each agent select its partition index d_i independently and uniformly at random from $\{1, \ldots, k_i\}$ so that $n_\mathrm{d} = \max_{i \in \mathcal{A}} k_i$. We refer to such planners as implementing Randomized Sequential Partitions ($\mathrm{RSP}_{n_\mathrm{d}}$) as this approach partitions agents into n_d groups which plan sequentially over the same number of sequential steps.

We consider two policies for selection of k_i based on the weights of the redundancy graph (5.4) and a per-agent budget for additive suboptimality $\gamma > 0$. For *global adaptive* planners agents $i \in \mathcal{A}$ select from a fixed number of partition indices proportional to the

total redundancy so that

$$k_i = n_{\mathrm{d}} = \left\lceil \frac{1}{n_{\mathrm{a}}\gamma} \sum_{(i,j)\in\mathcal{E}} w_{ij} \right\rceil. \tag{5.10}$$

With *local adaptive* planners k_i is proportional instead to the cumulative redundancy for that agent which may be large compared to other agents but involves less global knowledge

$$k_i = \left\lceil \frac{1}{2\gamma} \sum_{j\in\mathcal{A}\backslash\{i\}} w_{ij} \right\rceil. \tag{5.11}$$

Note that the factor of *two* in arises because the sum in (5.11) counts each edge twice unlike (5.10). Both local and global planners respect the following bound.

Theorem 7 (Suboptimality of RSP planning). *Given a budget $\gamma > 0$ for per-agent suboptimality and a planner defined according to (5.3) which partitions agents according to (5.9) by drawing partition indices d_i uniformly from $\{1, \ldots, k_i\}$ using (5.10) or (5.11) suboptimality is bounded in expectation as*

$$g(X^\star) \le 2\mathbb{E}[g(X^{\mathrm{d}})] + n_{\mathrm{a}}\gamma. \tag{5.12}$$

Proof. The expectation of the cumulative weight of deleted edges for either planner is bounded as

$$\mathbb{E}\left[\sum_{(i,j)\in\tilde{\mathcal{E}}} w_{ij}\right] = \frac{1}{2}\sum_{i=1}^{n_{\mathrm{a}}} \mathbb{E}\left[\sum_{j\in D_{d_i}\backslash\{i\}} w_{ij}\right]$$

$$\le \sum_{i=1}^{n_{\mathrm{a}}} \frac{1}{2k_i} \sum_{j\in\mathcal{A}\backslash\{i\}} w_{ij} \tag{5.13}$$

where the inequality accounts for when $d_j > k_i$. That is, when another agent j selects a partition index $d_j > k_i$ outside of the set considered by i, the corresponding edge cannot be deleted from the perspective of agent i. For global adaptive planners, $k_i = n_{\mathrm{d}}$ for all i and (5.13) simplifies to $\frac{1}{n_{\mathrm{d}}}\sum_{(i,j)\in\mathcal{E}} w_{ij}$ and holds with equality. Then (5.12) follows by applying (5.10) or (5.11) and substituting into the expectation of (5.7) over partitions of agents. ∎

Restating in terms of α-redundancy provides a stronger statement that is useful when varying the number of agents.

Corollary 7.1 (Constant factor suboptimality). *Problems with fixed α-redundancy satisfy the constant-factor bound*

$$\frac{1-\epsilon}{2} g(X^\star) \le \mathbb{E}[g(X^{\mathrm{d}})] \tag{5.14}$$

for $\epsilon > 0$ and a budget of

$$\gamma = \frac{\epsilon}{\alpha n_{\mathrm{a}}} \sum_{(i,j) \in \mathcal{E}} w_{ij} \tag{5.15}$$

by substituting (5.15) *into* (5.12), *applying* (5.6), *and rearranging.*

Corollary 7.2 (Fixed n_{d} for global planning)**.** *Given fixed α-redundancy, global adaptive planners* (5.10) *provide constant-factor suboptimality for $n_{\mathrm{d}} = \lceil \frac{\alpha}{\epsilon} \rceil$ sequential planning steps which follows by rearranging* (5.15) *to match* (5.10).

5.5.3   Near-optimality for varying numbers of agents

In this section, we present sufficient conditions to preserve these guarantees when increasing the number of agents. These conditions correspond intuitively to scenarios where agents have access to local actions and where the environment volume and rewards scale with the number of agents.

We say problems (5.2) with n_{a} agents exhibit *β-linear scaling* for $\beta > 0$ if

$$g(X^\star) \geq \beta n_{\mathrm{a}} \tag{5.16}$$

which expresses the condition that rewards scale with the number of agents.

In order to express the relationship between inter-agent distances—or the distribution of agents—and inter-agent redundancy, define a function of inter-agent distance $r : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ so that

$$w_{ij} \leq r(||\mathbf{p}_i - \mathbf{p}_j||) \tag{5.17}$$

where $|| \cdot ||$ is some norm and $\mathbf{p}_i, \mathbf{p}_j \in \mathbb{R}^d$ are appropriately defined agent positions associated with the blocks of the partition matroid. The following theorem identifies sufficient conditions for problems to have finite α on average and in turn to satisfy Theorem 7 and corollaries which implies a constant expected number of sequential steps (n_{d}) for the global planner design (5.10) and any number of agents.

Theorem 8 (Finite average redundancy)**.** *Consider a class of problems* (5.2) *with a distribution of agents in $\mathbb{R}^d$ with finite density of at most ρ that satisfies linear scaling* (5.16) *and has redundancy bounded in terms of inter-agent distance* (5.17) *for fixed β and r. If $\int_0^\infty r(s)s^{d-1}\,\mathrm{d}s$ is finite, the average value of α, interpreted as a random variable, is also finite.*

Proof. Let ρ be an upper bound on the marginal density of agents in $\mathbb{R}^d$, and let A_d be the surface area of the unit sphere under the chosen norm. Because the distribution of agents has fixed maximum density ρ, we may obtain results for arbitrarily large numbers of agents by taking the limit as this distribution of agents covers the entire Euclidean space. By applying (5.17) and integrating over spheres centered on $\mathbf{p}_i$ for an arbitrarily large environment, the expected redundancy for a given agent is at most $\hat{r} = \rho A_d \int_0^\infty r(s)s^{d-1}\,\mathrm{d}s \geq \mathbb{E}\left[\sum_{j \in \mathcal{A} \setminus \{i\}} w_{ij}\right]$ which is proportional to the integral in Theorem 8. Treating α as a random variable so that (5.6) is tight and applying (5.16) we get $\mathbb{E}[\alpha] \leq \mathbb{E}\left[\frac{\sum_{(i,j) \in \mathcal{E}} w_{ij}}{g(X^\star)}\right] \leq \frac{\hat{r}}{\beta}$ which is finite if $\hat{r}$ is also finite given that $\beta > 0$. $\blacksquare$

5.5.4 Limited communication range

Similar analysis can be used to analyze or design limits on communication range. Let r_c be the maximum communication range so that the set of agents that are in range is $\mathcal{B}_i = \{j \mid r_c > ||\mathbf{p}_i - \mathbf{p}_j||\}$. Then, any existing communications graph can be readily modified by intersecting the set of in-neighbors with the set of agents that are in range to obtain a new set of neighbors $\hat{\mathcal{N}}_i = \mathcal{B}_i \cap \mathcal{N}_i$. Doing so significantly reduces messaging: instead of each robot sending a number of messages proportional to the total number of robots, range limits reduce the number of messages per robot to the average size of $\mathcal{B}$. To emphasize this difference, we refer to this approach as Range-limited RSP (or R-lRSP).

To account for ignoring agents past a given range, applying (5.17) to Theorem 6, demonstrates that each agent incurs at most

$$\sum_{j \in \mathcal{A} \backslash (\{i\} \cup \mathcal{B}_i)} \frac{w_{ij}}{2} \leq \sum_{j \in \mathcal{A} \backslash (\{i\} \cup \mathcal{B}_i)} \frac{r(||x_i - x_j||)}{2} \tag{5.18}$$

additional suboptimality (i.e. increase to γ in (5.12)). Then, as in Sec. 5.5.3, the expectation of (5.18) over the distribution of agents is upper-bounded by $\rho A_d \int_{r_c}^\infty r(s) s^{d-1} \, ds$. This reduces the problem of limited communication range to a question of whether the additional suboptimality is acceptable and whether techniques such as multi-hop communication are necessary to extend the communication range. Note that even though the communication range can be designed to incur arbitrarily little additional suboptimality, limiting the communication range is not sufficient to reduce the number of sequential planning steps as those agents within range of any one agent may, in turn, depend on agents that are out of range of that agent and so on.

5.6 Probabilistic coverage objectives

Although submodular set functions have been studied extensively, set functions with higher-order monotonicity properties have received relatively little interest [44, 70, 149]. Before moving on to present simulation results, let us examine one such objective which satisfies the conditions presented in Sec. 5.1.1. The two scenarios that we will study in simulation involve special-cases of this following objective which is a mild extension of weighted set cover.

Consider a general event detection or identification problem with independent, probabilistic failures. We define a set of events $\mathscr{E}$ and let each event $e \in \mathscr{E}$ have value $v_e \geq 0$. Each event $e \in \mathscr{E}$ and element of the ground set $x \in \mathscr{U}$ is associated with an independent failure probability $0 \leq p_x^e \leq 1$. The expected value of identified events given a set of sensing actions $X \subseteq \mathscr{U}$ is then

$$g(X) = \sum_{e \in \mathscr{E}} \left(1 - \prod_{x \in X} p_x^e\right) v_e \tag{5.19}$$

and is equivalent to the well-known weighted set cover objective in the deterministic case ($p_x^e \in \{0, 1\}$).

The probabilistic sensor coverage objective is monotonically increasing, submodular, and 3-increasing and even satisfies alternating monotonicity conditions on higher derivatives [9, 44].[3] These conditions follow inductively by demonstrating that differences are similar in form to the original function. We note that similar results exist in the literature for coverage Wang et al. [191, Section 4.1] and related generalizations thereof [157, Theorem 13].

Theorem 9 (Probabilistic coverage is monotonic, submodular, and 3-increasing). *Coverage with sensor failure* (5.19) *and, by extension, weighted set cover satisfy alternating monotonicity conditions and are i-increasing and i-decreasing respectively for odd and even i. As such, each is monotonic, submodular, and 3-increasing.*

Proof. The probabilistic coverage objective (5.19) can be written in the form

$$g(X) = a - \sum_{e \in \mathscr{E}} \prod_{x \in X \setminus A} p_x^e \hat{v}_e \tag{5.20}$$

for $a \in \mathbb{R}$, $A \subseteq \mathscr{U}$, $\hat{v}_e \in \mathbb{R}_{\geq 0}$, and $0 \leq p_x^e \leq 1$ for $e \in \mathscr{E}$ and $x \in \mathscr{U}$. Observe that g is monotonic due to the form of the product and because the terms of the product are probabilities. Recall the recursive definition of the derivative (3.16), and consider the derivative $g(Y|X)$ in the direction $Y \subseteq \mathscr{U}$ evaluated at $X \subseteq \mathscr{U} \setminus Y$ which has the form

$$
\begin{aligned}
g(Y|X) &= g(Y, X) - g(X) \\
&= -\sum_{e \in \mathscr{E}} \prod_{j \in (X \cup Y) \setminus A} p_j^e \hat{v}_e + \sum_{e \in \mathscr{E}} \prod_{k \in X \setminus A} p_k^e \hat{v}_e \\
&= \sum_{e \in \mathscr{E}} \left(1 - \prod_{j \in Y \setminus A} p_j^e \right) \prod_{k \in X \setminus A} p_k^e \hat{v}_e.
\end{aligned} \tag{5.21}
$$

Because $-g(Y|\cdot)$ has the same form as g, this first derivative $g(Y|\cdot)$ is decreasing; that is, g is submodular. Applying the recursive definition of the derivative (3.16) to $g(Y|\cdot)$ then produces the second derivative which is, in turn, a monotonically increasing function with the same form as g. Therefore, g is 3-increasing (5.1). Moreover, by induction g also satisfies alternating monotonicity conditions on higher derivatives (Sec. 3.5.1d). Finally, probabilistic coverage (5.19), having the same form, satisfies the same monotonicity conditions. ∎

5.7 Results and discussion

The proposed distributed planning approach is evaluated through two sets of simulation experiments, each using a variant of the objective function analyzed in Sec. 5.6. The first evaluates the performance of distributed planners (RSP) that use various numbers of sequential planning steps (agent partition size n_{d}) in an area coverage task. The second set of experiments evaluates adaptive planning and limits on communication range (RSP

---

[3]See discussion of the Möbius inversion [9, Sec. 6.3].

	Area Coverage		Probabilistic Sensing	
	By n_a	$n_\mathrm{a}=50$	By n_a	$n_\mathrm{a}=50$
r_s	$\sqrt{2/(n_\mathrm{a}\pi)}$	0.113	$\sqrt{0.6/(n_\mathrm{a}\pi)}$	$6.18\cdot10^{-2}$
r_a	$2r_\mathrm{s}$	0.226	$4r_\mathrm{s}$	0.247

Table 5.1: Agent (r_a) and sensor (r_s) radii as a function of the number of agents ($n_\mathrm{a}=50$ in this chapter).

and R-lRSP) in a more complex problem with spatially varying rewards and probabilistic sensing.

5.7.1   Common parameters of experiment designs

Several aspects of experiment design are kept constant in each scenario. Each scenario is evaluated in 50 random trials[4] and features a large number of agents ($n_\mathrm{a}=50$) so that the proposed distributed planners utilize many times fewer sequential planning steps than a standard sequential planner (Alg. 2). Agent positions are distributed uniformly at random over the unit square, and each agent has a choice of 10 available sensing actions ($\mathcal{B}_i$) which are sampled from a uniform distribution over a disk with radius r_a centered on the agent position. Although the two sets of simulation experiments do not use the same sensor model, each is a function of sensor radius r_s. The sensor and agent radii used in each experiment[5] are listed in Tab. 5.1. In each case, the objective is designed to take on values no greater than one.

5.7.2   Area coverage and evaluation of distributed planning

The reward for the area coverage task is the area of the union of discs, each with radius r_s, intersected with the unit square. In terms of the sensor model defined in Sec. 5.6, this is equivalent to having a failure probability of one outside the disk and zero inside. An example of one simulation trial (using parameters tuned for visualization purposes) is depicted in Fig. 5.3. The experiments compare distributed planning with RSP and fixed partition sizes, $k_i = n_\mathrm{d} \in \{2, 4, 8\}$, to sequential planning (Alg. 2) and two naive planners: completely random action selection and myopic maximization of the objective over the local space of sensing actions (wherein agents ignore each entirely, equivalent to $n_\mathrm{d} = 1$). Figure 5.4 shows the results of these experiments.

Our RSP planners perform well, although the performance bounds (Theorem 7) are technically degenerate for these instances because the deleted edge weight generally exceeds the maximum possible objective value (one unit of area) which is evident from the

---

[4]Unless otherwise specified, each trial features both random planners and scenarios according to their respective designs.

[5]Agent and sensor radii are set according to a normalization over the number of agents and by using parameter search to minimize the ratio of the average performance of myopic and sequential planning to identify hard problem cases.
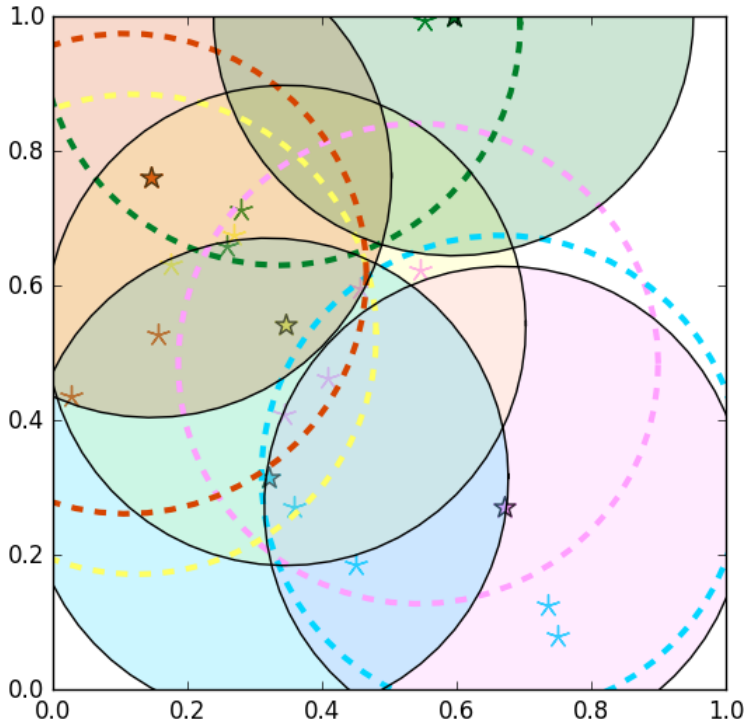
Figure 5.3: This figure depicts a maximum area coverage problem and a sequential solution. Agents each have a unique color and are distributed uniformly throughout the environment. Sensing actions (∗) are in turn distributed uniformly within agent radii (dashed lines). The solution consists of one selected action (⋆) for each agent; these actions are centered on the translucent disks which make up the covered area.

cumulative weights of the inter-agent redundancy graph.[6] However, the trend in performance is similar to what would be expected for increasing n_d as the objective values of our RSP planners approach the performance of sequential planning approximately with $1/n_d$: the difference in area coverage compared to sequential planning decreases by approximately half each time n_d is doubled and by 9.9 times from $n_d = 1$ to $n_d = 8$. Overall, the performance of the distributed RSP planner represents a significant improvement over sequential planning. Given the number of agents, even the greatest value of n_d provides a 6-times improvement in the number of sequential planning steps. Then, by the scalability analysis in Sec. 5.5.3, similar performance can be expected for larger problems given similar densities of agents.

5.7.3 Adaptive planning with probabilistic sensing and non-uniform events

The goal of the probabilistic sensing task is to maximize the value of correctly identified events (e.g. correct classification of objects moving through the environment). For each trial $n_e = 50$ events, each worth a value of $1/n_e$, are sampled from a fixed Gaussian

---

[6]Note that the bound ceases to be degenerate for larger values of n_d and that all results for suboptimality and scaling still apply.

(a) Objective values **(b)** Edge weight **(c)** Total graph weight
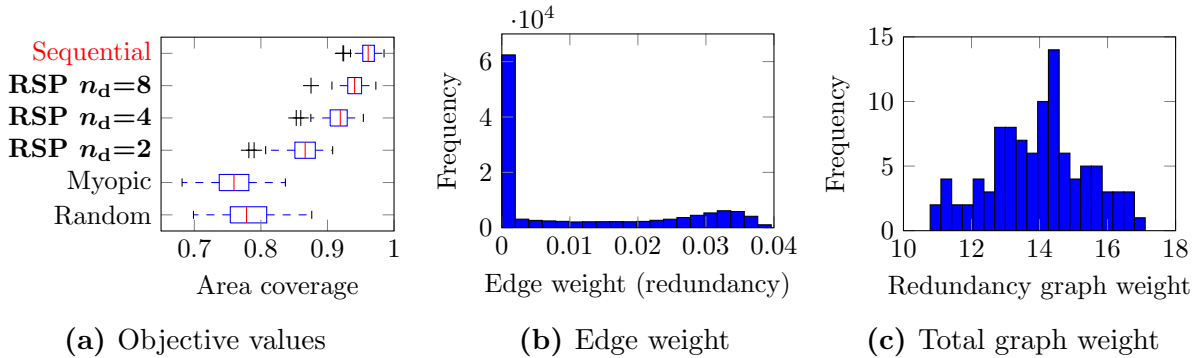
Figure 5.4: Results for the area coverage problem (Fig. 5.3): (a) Objective values for myopic and random planning (no coordination), variations of the **distributed RSP planner** with n_d sequential planning steps, and fully sequential planning (intractable for large numbers of agents). The performance of the proposed distributed planner approaches sequential planning given many times fewer sequential planning steps. (b) Redundancies are computed for each pair of agents. Sensing actions (disks) for distant agents cannot overlap resulting in many zero weighted edges, and remaining edges are distributed according to varying degrees of overlap in potential sensing actions. (c) The total weight of the redundancy graph is largely between 13 and 17. Even though the planners perform well, our suboptimality bounds would no longer be meaningful as deleted edge weights would exceed maximum objective values.

mixture, rejecting samples outside the unit square. An example is shown in Fig. 5.5 although using agent parameters more appropriate for visualization purposes but the same Gaussian mixture and number of events. This results in a spatially varying distribution of reward and redundancy. The success probability of the sensor model is e^{-x^2/r_s^4}, where x is the distance from sensor to event location. This success probability effectively amounts to area coverage with soft edges.

This set of experiments evaluates adaptive planning and limited communication range. The budget for deleted edge weight per-agent for the local and global planners is set to $\gamma = 0.4/n = 8 \cdot 10^{-3}$. Range-limited RSP planners are obtained by deleting edges from the respective instances of the distributed planners with a communication range of $r_c = 2r_a$ which allows for a small amount of redundancy at the limits of the sensor range r_s. Random planning is not included in this set of experiments because it vastly under-performs myopic planning as the problem design ensures that a large fraction of sensing actions provide little value.

Fig. 5.6 shows the results of these experiments. Local and global adaptation each perform almost identically in terms of distributions of objective values and with distribution only slightly below that of sequential planning. Because the objective and actions are highly local, enforcing limits on the communication range has little impact on the planner performance in terms of either objective value or cumulative weight of deleted edges. The global adaptive RSP planner obtains consistent partition sizes by averaging over all agents. In contrast with the local planners, agents sometimes select from as many as 33 planning steps. Sec. 5.5.3 provides some discussion of mitigation strategies to avoid such long planning times. More generally, mixing the extremes of the local and global RSP planning may be desirable and avoid computing averages over all agents.

(a) Scenario   **(b)** Planner result
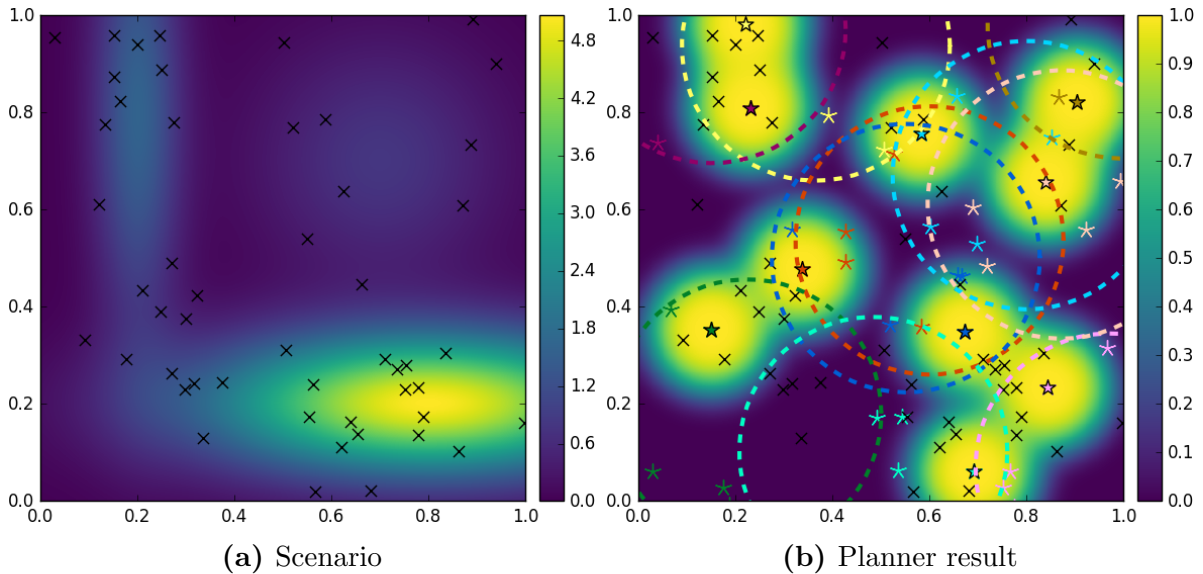
Figure 5.5: This figure shows an example of a probabilistic sensing scenario and a sequential solution. Parameters for the scenario are identical to the experimental trials, but the parameters for the agents have been tuned for purpose of visualization. The goal of this task is to maximize the expected number of successful detections or identifications. (a) For each trial, events (x) are sampled from a mixture of Gaussians and are identified correctly with some probability dependent on the sensing actions. (b) Agents, each shown in a different color, are distributed uniformly throughout the environment. Sensing actions (∗) are distributed uniformly within the agent radius (dashed lines), and each agent selects a single sensing action (⋆) and successfully identifies events according to a soft-coverage sensing model. The resulting identification probability given selected actions is shown in the background; identification probability is high (yellow) near selected sensing actions and, for good selections, near the events.

5.8 Conclusion

Efficiently solving submodular maximization problems on sensor networks is challenging due to the inherent sequential structure of common planning strategies. Whereas prior works [75, 82] have shown that worst-case performance degrades rapidly when reducing the number of sequential planning steps, we show that constant-factor performance approaching that of the standard sequential algorithm can be obtained via randomized planning (RSP) so long as cumulative redundancy between agents is at most proportional to the objective values. Toward this end, the inter-agent redundancy graph expresses the degree of coupling between agents in the submodular maximization problem, and functions that are 3-increasing admit performance bounds in terms of this graph structure. Further, we have demonstrated that the bounds we obtain with this approach are readily applicable for planner design such as for adapting the number of sequential planning steps or via range limits on communication.

Later chapters will apply these results to the design of online anytime receding-horizon planners for target tracking tasks and exploration (as in Chapter 4). Likewise, this same approach is applicable to other multi-robot sensing tasks which the reader might encounter. Ultimately, planning in real time will also require increased attention to timing for planning

(a) Objective values

(b) Partition sizes: global adaptive RSP

(c) Partition sizes: local adaptive RSP

(d) Deleted edge weight
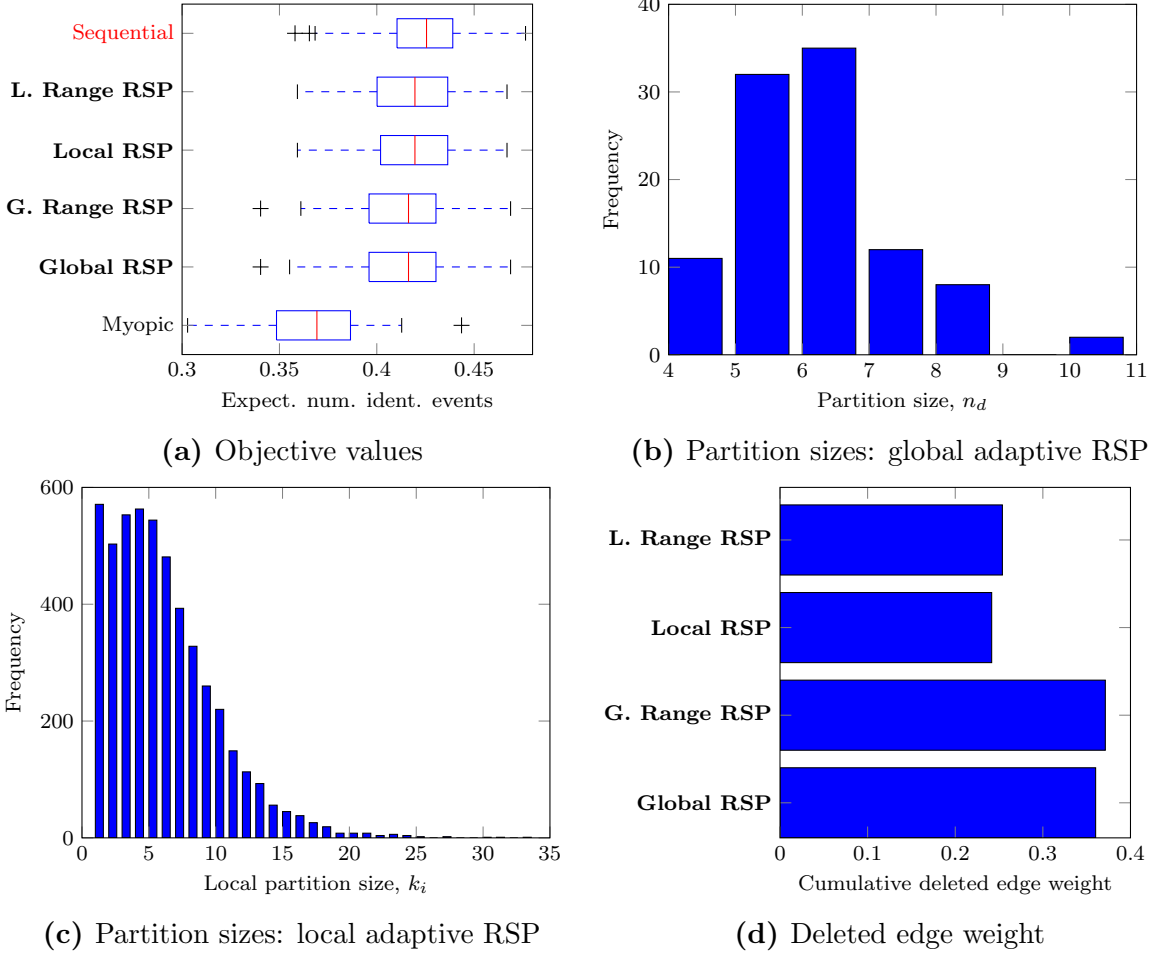
Figure 5.6: Results for the probabilistic sensing problem (Fig. 5.5): (a) The proposed **local** and **global adaptive RSP planners** along with their **Range-limited RSP counterparts** outperform myopic planning and approach the performance of the fully sequential planner (intractable for large numbers of agents) in terms of objective values. (b) The global adaptive planner uses 4 to 10 partitions in all trials while (c) the local adaptive planner occasionally uses local partitions sizes exceeding 20 for agents with high redundancy. (d) The cumulative weight of deleted edges is on the same order as the objective value and is below the desired limit of 0.4. Range-limited planners are obtained by deleting edges from the associated adaptive planner at the cost of relatively little additional deleted edge weight.

steps such as reasoning about the impact of available planning time on anytime planning performance.

Additionally, we have noted that submodular objectives such as mutual information are not necessarily 3-increasing. Identifying when objective functions are 3-increasing (exactly or approximately) is central to broad application of the results in this chapter, and we will find ways to obtain similar results for mutual information objectives for both target tracking and exploration.

Finally, high-order monotonicity conditions have been useful in other problems involving submodular objectives [84, 106, 157, 191], and developing these applications would be an interesting direction for future study.

Chapter 6

Receding-Horizon Planning for Target Tracking with Sums of Submodular Functions

Planners for robotics and sensor planning problems frequently introduce notions of locality, coupling, or redundancy amongst robots, sensors, or locations [26, 108]. Chapter 5 introduced one such notion of redundancy using properties of 3-increasing functions in order to design scalable planners. In this chapter, we expand on the idea of redundancy as a tool for planner design by analyzing a class of factored mutual information objectives that are relevant to target tracking and form sums of submodular functions.

Chapter 5 provided performance bounds for a slight generalization of weighted set coverage but also presented a counterexample for mutual information with conditionally independent observations (i.e. for typical submodular mutual information objectives, see Sec. 3.5.1a). Mutual information is particularly important for perception and control [37, 103, 163] and is the focus of Chapter 4. As such, generalizing to a variety of objective functions, as we will begin to do here, is an important part of this thesis.

Additionally, this chapter applies the RSP and R-lRSP planners from the previous chapter to produce anytime, receding-horizon planners and includes additional analysis for this setting.

6.1 Introduction for target tracking

In target tracking problems, robots seek to observe a number of discrete targets whose states may evolve in time, such as for surveillance, monitoring wildlife [52], and intercepting rogue UAVs [165]. Such sensing tasks typically involve planning to observe a number of discrete objects, *targets*, whose states change with time. When a large quantity of targets are spread over more space than a single robot can cover, deploying teams of robots can improve tracking performance.

Even simple target tracking problems, such as with noisy range sensors, produce multimodal posterior distributions that do not have closed-form solutions; planning and tracking

systems may then approximate both the posterior and sensing utility [34, 35]. Realistic environments can also induce complex motion models which arise in search on road networks [145] and in indoor environments [89]. This motivates selection of sufficiently general objectives and path planners to capture the degree of complexity in these problems. To address this, we provide analysis that applies to general problems and even adversarial problem selection so long as the targets' dynamics do not depend on the tracking robots.[1]

Systems for target tracking in multi-robot settings often rely on greedy algorithms [182, 202] for submodular maximization. However, existing results and those in Chapter 5 that seek to develop scalable planners that are relevant to target tracking problems are also limited to coverage-like objectives [177]. Analysis for sequential planners also typically assumes individual robots obtain either exact [7, 69] solutions or solutions within a constant factor of optimal [102, 169]. Although that analysis is appropriate for single-robot planners with guarantees on solution quality [41, 102, 169], assuming constant-factor suboptimality is less suited for approximate, anytime, and sampling-based planning [7, 90, 116] which we will address in this chapter.

6.1.1 Contributions

This chapter presents a distributed planning algorithm, analysis that accounts for common algorithmic approximations, and design of such a planner along with simulation results.

6.1.1a Analysis of pairwise redundancy for distributed planning

The analysis in this chapter demonstrates that RSP planners are applicable to target tracking problems by providing guarantees on solution quality in terms of pairwise redundancy between robots' actions via an extension to sums of submodular functions. This extends our prior results for coverage-like objectives (Chapter 5) to include more general objectives such as for mutual information[2] which can represent information gain with respect to targets that have complex states and dynamics.

6.1.1b Analysis of an approximate, anytime planning

We also account for the contributions of common sources of suboptimality *(approximation of the objective and suboptimal single-robot planning)*. Our results affirm that methods for submodular maximization are applicable for target tracking in the presence of approximate objective values and anytime planners that may occasionally produce poor results or fail.

6.1.1c Design of a planner for multi-robot multi-target tracking

Finally, we apply these results to develop a planner for multi-robot multi-target tracking with a mutual information objective, and we show that distributed planning for target

---

[1]The latter condition excludes pursuit-evasion problems [49, 165].

[2]Fig 5.2 describes a mutual information objective that violates requirements on "coverage-like" objectives.
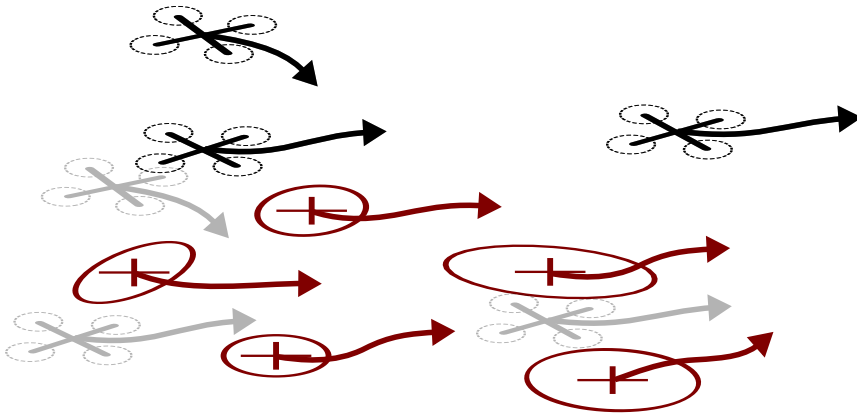
Figure 6.1: A team of aerial robots $\mathcal{R}$ (black) plan over a receding horizon to track a number of targets $\mathcal{T}$ (red). In doing so, robots select sensing actions to minimize uncertainty in the target states which evolve independent of each other and the robots.

tracking with Range-limited RSP *in constant time, independent of the number of robots* can guarantee suboptimality approaching that of fully sequential planning. Finally, simulation results support our claim that R-lRSP maintains consistent solution quality while planning for up to 96 robots. This produces over a $20\times$ reduction in the number of sequential planning steps and an even further improvement in computation time (sequential planning is intractable at this scale).

6.2 Target tracking problem

Consider a set of moving targets $\mathcal{T} = \{1, \ldots, n_\text{t}\}$ and robots tracking those targets $\mathcal{R} = \{1, \ldots, n_\text{r}\}$, seeking to minimize uncertainty (entropy [58]), as illustrated in Fig. 6.1. Let $\mathbf{x}_{i,t}^\text{r} \in \mathbb{R}^{d^\text{r}}$ and $\mathbf{x}_{j,t}^\text{t} \in \mathbb{R}^{d^\text{t}}$ be the respective states of robot $i \in \mathcal{R}$ and target $j \in \mathcal{T}$ at time $t \in \{0, \ldots, T\}$. The states of each evolve in discrete time, according to known dynamics

$$\mathbf{x}_{i,t+1}^\text{r} = f^\text{r}(\mathbf{x}_{i,t}^\text{r}, u_{i,t}), \qquad\qquad \mathbf{x}_{j,t+1}^\text{t} = f^\text{t}(\mathbf{x}_{j,t}^\text{t}, \epsilon_{j,t}^\text{t}), \qquad (6.1)$$

where $u_{i,t} \in \mathcal{U}$ is a control input from the finite set of control inputs $\mathcal{U}$ and $\epsilon_{i,t}^\text{t}$ is a random disturbance. The robots receive noisy observations $\mathbf{y}_{i,j,t}$ of the target states according to

$$\mathbf{y}_{i,j,t} = h(\mathbf{x}_{i,t}^\text{r}, \mathbf{x}_{j,t}^\text{t}, \epsilon_{i,j,t}^\text{y}) \qquad (6.2)$$

where $\epsilon_{i,j,t}^\text{y}$ is observation noise. We refer to states and observations collectively using boldface capitals as $\mathbf{X}_t^\text{r}$, $\mathbf{X}_t^\text{t}$, and $\mathbf{Y}_t$, each at time t.

6.2.1 Receding-horizon optimization problem

The robots plan to jointly maximize a mutual information (MI) objective over a receding horizon, starting at time t with duration l. Specifically, robots maximize a submodular, monotonic, and normalized objective g subject to a partition matroid constraint, and the optimal set of control actions is

$$X^\star \in \arg\max_{X \in \mathscr{I}} g(X) \qquad (6.3)$$

where $\mathscr{I}$ is a partition matroid that represents assignment of sequences of control actions to robots such that

$$\mathscr{B}_i = \{(i, u_{1:l}) \mid u_{1:l} \in \mathcal{U}^l\} \quad \forall i \in \mathcal{R}, \tag{6.4}$$

and g is the mutual information between observations and target states given the choice of control actions. Interpreting future states $\mathbf{X}_{t+1:t+l}^{\mathrm{t}}$ and observations $\mathbf{Y}_{t+1:t+l}$ as random variables (induced by the process (6.1) and observation (6.2) noise), and write $\mathbf{Y}_{t+1:t+l}(X)$ for $X \subseteq \mathscr{U}^{3,4}$

$$g(X) = \mathbb{I}(\mathbf{X}_{t+1:t+l}^{\mathrm{t}}; \mathbf{Y}_{t+1:t+l}(X) | \mathbf{Y}_{0:t}, \mathbf{X}_{0:t}^{\mathrm{r}}) \tag{6.5}$$

where $\mathbb{I}(X; Y|Z)$ is the Shannon mutual information between X and Y conditional on Z and quantifies the reduction in uncertainty (entropy) of one random variable given another. Although we discuss properties of entropy and mutual information, we refer interested readers to Cover and Thomas [58] for detail and definitions. Critically, mutual information objectives are normalized, monotonic, and submodular (defined in Sect. 3.5.1) when observations are conditionally independent of target states [107] although higher-order monotonicity conditions may not apply (see discussion in Fig. 5.2). Because the robot states are known and deterministic, individual targets and observations of the same are jointly independent so that (6.5) can be written as a sum over the targets

$$g(X) = \sum_{j \in \mathcal{T}} \mathbb{I}(\mathbf{X}_{j,t+1:t+l}^{\mathrm{t}}; \mathbf{Y}_{j,t+1:t+l}(X) | \mathbf{Y}_{j,0:t}, \mathbf{X}_{0:t}^{\mathrm{r}}) \tag{6.6}$$

because the mutual information is a difference of entropies [58, Eq. 2.45] which, in turn, decompose as sums over targets [58, Theorem 2.6.6].

6.2.2 Spatial locality

Spatial locality in target tracking problems arises when the capacity to sense a target decreases with some measure of distance. The variations in each robot's $i \in \mathcal{R}$ ability to sense different targets $j \in \mathcal{T}$ take the form of channel capacities $C_{i,j}$ from information theory [58, Chapter 7] so that

$$C_{i,j} = \max_{x \in \mathscr{B}_i} \mathbb{I}(\mathbf{X}_{j,t+1:t+l}^{\mathrm{t}}; \mathbf{Y}_{j,t+1:t+l}(x) | \mathbf{Y}_{j,0:t}, \mathbf{X}_{0:t}^{\mathrm{r}}). \tag{6.7}$$

This channel capacity is itself an informative planning problem although designers may apply existing results for channel capacities by relaxing (6.7) such as given bounded travel distances on the robot and target.

---

[3]More formally, $\mathbf{Y}_{t+1:t+l}(X)$ is a random variable that encodes the noisy observations (6.2) that robot i receives after executing $u_{1:l}$ according to (6.1) for each $(i, u_{1:l}) \in X$. When $X \subseteq \mathscr{U}$ includes multiple hypothetical assignments to the same robot—which arises in the analysis—duplicates obtain unique observations and observation noise ϵ^{t}.

[4]All analysis applies to discrete and continuous mutual information and transformations of target states e.g. for MI with position but not derivatives.

Uncertainty in target positions is closely related to spatial locality in target tracking problems. Even when sensing power falls off quickly with distance, distant robots may expect to gain small amounts of information about sufficiently uncertain targets. Rather than introduce additional machinery to characterize suboptimality in terms of uncertainty in target positions, we will quantify the effect of spatial locality in terms of channel capacities (6.7).

6.2.3 Computational model

A common feature of distributed planning problems is limited access to information. The central assumption for our computational model is that each robot $i \in \mathcal{R}$ is able to approximate the objective for its own set of actions $\mathcal{B}_i$. That is, each robot has access to an approximation of marginal gains $\widetilde{g}_i(x_i|A)$ for each action $x_i \in \mathcal{B}_i$ in its local set and for prior selections $A \subseteq \mathcal{U}$. This may reflect both approximate evaluation of mutual information and constraints on access to sensor data such as allowing robots to ignore distant targets.

Robots also have limited access to the ground set $\mathcal{U}$: they produce elements of their local sets $\mathcal{B}_i \subseteq \mathcal{U}$ implicitly via local planners and obtain access to other robots' actions when they communicate their decisions.

6.3 Distributed planning algorithm

Algorithm 4 Distributed algorithm for receding-horizon target tracking with Randomized Sequential Partitions (RSP) from the perspective of robot $i \in \mathcal{R}$ for execution starting at time t.

1: $\mathcal{N}^{\mathrm{in}} \leftarrow$ *in* neighbors of robot i
2: $\mathcal{N}^{\mathrm{out}} \leftarrow$ *out* neighbors of robot i
3: $\theta \leftarrow$ sensor data (or summary), accessible to robot i

4: RECEIVE: $X^{\mathrm{d}}_{\mathcal{N}^{\mathrm{in}}}$ from $\mathcal{N}^{\mathrm{in}}$
5: $\widetilde{g} \leftarrow$ approximation of g given θ
6: $x^{\mathrm{d}} \leftarrow$ PLANANYTIME$(\widetilde{g}, \mathbf{x}^{\mathrm{r}}_t, X^{\mathrm{d}}_{\mathcal{N}^{\mathrm{in}}})$
7: SEND: x^{d} to $\mathcal{N}^{\mathrm{out}}$
8: EXECUTE: x^{d} starting at time t and until the beginning of the result of the next planning round

---

The distributed planner that we present in this section extends methods from Chapter 5. Algorithm 4 provides pseudo-code for this distributed RSP algorithm based on a directed acyclic graph where robots are vertices with incoming edges $\mathcal{N}^{\mathrm{in}}$ (and outgoing $\mathcal{N}^{\mathrm{out}}$). Although we do not emphasize timing, this algorithm runs in synchronous epochs whereby the robots collectively maximize information gain to collectively solve instances of the joint receding-horizon optimization problems (6.3). The output of this planner is then the collection of the robots' decisions $X^{\mathrm{d}} = \{x^{\mathrm{d}}_1, \ldots, x^{\mathrm{d}}_{n_{\mathrm{r}}}\}$.

Each robot begins planning after receiving decisions from a set of in-neighbors and approximates the objective based on available data (θ) and computational resources (lines 4–6). Once the planner exits or runs out of time, the robot commits to an action which it sends to any out-neighbors and then executes in a receding-horizon fashion (lines 7–8). In this exposition, we assume the in- and out-neighbors are known for brevity. However, the planner graph can also be implicit based on the messages the robot has received when it starts planning. Through this process, the single-robot planners (PLANANYTIME) collectively run in n_d sequential steps. We assume robots are randomly assigned to planning steps via the RSP methods presented in Chapter 5.

6.4   Cost model for approximate distributed planning

Aside from applying RSP to receding-horizon optimization, we seek to account for approximations (sampling based planning and objective evaluation and ignoring distant targets) pursuant to constraints on computation time and information access. Specifically, the analysis (Sect. 6.5) will account for costs of approximations arising in *distributed planning*, *objective evaluation*, and *anytime (single-robot) planning*. These costs will then relate the suboptimality of Alg. 4 to the nominal bound for sequential planning.

6.4.1   Generalized cost of suboptimal decisions for individual robots

Before moving on to the individual costs, let us present a generalized expression of costs for approximations in decision making. Any of the approximations we encounter can inhibit exact evaluation of maximization steps (PLANANYTIME) in the distributed planner (Alg. 2). Rather than assume constant factor suboptimality at each step [169]—as would be appropriate if the local planner provided a consistent performance guarantee—we present a more flexible cost model that accounts for uncertain results that arise when planning in real time. Given an instance of (6.3) and a distributed solution X^d, the cost to robot $i \in \mathcal{R}$ for making a suboptimal decision x_i^d is the difference between the value of that decision and the true maximum over $\mathscr{B}_i$

$$\gamma_i(g, X) = \max_{x \in \mathscr{B}_i} g(x|X) - g(x_i^\mathrm{d}|X) \tag{6.8}$$

for the true objective g marginally specific prior decisions $X \subseteq X_{1:i-1}^\mathrm{d}$.

6.4.2   Cost of distributed planning on a directed acyclic graph

In the nominal sequential greedy algorithm (Alg. 2), robot $i \in \mathcal{R}$ plans with access to all previous decisions by robots $\{1, \ldots, i-1\}$. However, in our approach (Alg. 4) robots only have access to a subset $\mathcal{N}_i \subseteq \{1, \ldots, i-1\}$ of these decisions ($\mathcal{N}^\mathrm{in}$ in the algorithm description), which induces a directed acyclic graph with edges (j, i) for each robot $j \in \mathcal{N}_i$ whose decision i uses while planning [75, 82]. In a sense, the robots ignore decisions

by the remaining robots $\hat{\mathcal{N}}_i = \{1, \ldots, i-1\} \setminus \mathcal{N}_i$, and the cost of doing so is a second derivative (3.16):

$$\gamma_i^{\text{dist}} = g(x_i^{\text{d}}|X_{\mathcal{N}_i}^{\text{d}}) - g(x_i^{\text{d}}|X_{1:i-1}^{\text{d}}) = -g(x_i^{\text{d}}; X_{\hat{\mathcal{N}}_i}^{\text{d}}|X_{\mathcal{N}_i}^{\text{d}}). \tag{6.9}$$

Later, we will upper bound the right-hand-side in terms of $\hat{\mathcal{N}}_i$.

6.4.3 Cost of approximate evaluation of the objective

Although we do not focus on the communication and representation of sensor data, we assume robots have access to relevant data for exact or approximate evaluation of the mutual information objective (6.5). However, robots may ignore distant targets or approximate the objective via sampling. We say robot $i \in \mathcal{R}$ has access to a local approximation $\widetilde{g}_i$ of the objective, and the cost of this approximation (treating stochasticity implicitly) is at most the sum of the maximum over- and under-approximation of g

$$\gamma_i^{\text{obj}} = \max_{x_1, x_2 \in \mathcal{B}_i} \left(\widetilde{g}_i(x_1|X_{\mathcal{N}_i}) - g(x_1|X_{\mathcal{N}_i}) + g(x_2|X_{\mathcal{N}_i}) - \widetilde{g}_i(x_2|X_{\mathcal{N}_i}) \right). \tag{6.10}$$

Here, x_1 and x_2 are respectively the points where $\widetilde{g}_i$ most under- and over-approximates g over all decisions $\mathcal{B}_i$ available to robot i.

6.4.4 Cost of approximate (anytime) single-robot planning

Selecting sensing actions for individual robots produces informative path planning problems [41, 169]. Each robot has a limited amount of time available for planning and must terminate planning and transmit results soon enough so that later robots can make their own decisions before the plans go into effect (at time t in Algorithm 4).

Although some existing methods provide performance guarantees [41, 169, 200], designers who apply these methods may have to vary replanning rates or tune problem parameters to satisfy constraints on planning time for operation in real-time. On the other hand, methods such as randomized planning [90, 116] and gradient- and Newton-based trajectory generation [36, 93] converge to local or global maxima but provide no specific guarantees on solution quality before convergence for anytime planning. Along these lines, we will now provide analysis based on empirical performance and will apply Monte-Carlo tree search [27, 39] later for single-robot planning in the simulation results as in Chapter 4.

The cost of approximate single-robot planning in terms of empirical performance is then

$$\gamma_i^{\text{plan}} = \gamma_i(\widetilde{g}_i, X_{\mathcal{N}_i}^{\text{d}}). \tag{6.11}$$

This approach captures this inherent uncertainty of anytime planning and enables us to characterize collective performance in terms of the bulk suboptimality of single-robot planning.

6.5   Analysis of suboptimality of distributed planning

The distributed planner described in Alg. 4 achieves a performance bound that approaches that of sequential planning (Alg. 2) with additional suboptimality that arises from evaluating the objective, planning for individual robots, and distributed coordination of the team.

Theorem 10 (Suboptimality of Alg 4 for target tracking tasks). *Consider an instance of (6.3), any solution X^d that Alg. 4 produces satisfies*

$$g(X^\star) \leq 2g(X^d) + \sum_{i \in \mathcal{R}} \left(\gamma_i^{\mathrm{dist}} + \gamma_i^{\mathrm{obj}} + \gamma_i^{\mathrm{plan}} \right), \tag{6.12}$$

and the total cost of distributed planning is bounded by

$$\sum_{i \in \mathcal{R}} \gamma_i^{\mathrm{dist}} \leq \sum_{i \in \mathcal{R}} \sum_{j \in \hat{\mathcal{N}}_i} \widehat{\mathcal{W}}(i, j) \tag{6.13}$$

where $\widehat{\mathcal{W}}$ is a collection of edge weights which we will define later that describes redundancies that arise when pairs of robots may observe the same targets.

The proof of Theorem 10 is in Appendix B.6, and we provide a brief summary at the end of this section in Sect. 6.5.3 after introducing some preliminary results related to the first (Sect. 6.5.1) and second (Sect. 6.5.2) parts of this theorem.

Regarding the structure of Theorem 10 the bounds describe the performance of practical implementations of Alg. 4. An idealized version of this distributed algorithm would have access to exact objective values and maxima so that the associated costs γ^{obj} and γ^{plan} would all be zero. From this perspective, (6.12) describes how real implementations may deviate from this ideal and states that the total suboptimality is a simple accumulation of individual inefficiencies.

6.5.1   General suboptimality in multi-robot planning

The following lemma expresses the suboptimality of any decision as a sum of costs of suboptimal decisions.

Lemma 11 (Suboptimality of general assignments). *Given a submodular, monotonic, normalized objective g, any basis (assignment of actions to all robots) $X^{\mathrm{d}} \in \mathscr{I}$ on a simple partition matroid satisfies*

$$g(X^\star) \leq 2g(X^{\mathrm{d}}) + \sum_{i=1}^{n_{\mathrm{r}}} \gamma_i(g, X_{1:i-1}^{\mathrm{d}}). \tag{6.14}$$

The proof of Lemma 11 is in Appendix B.5.

Observe that if we obtain X^{d} via exact sequential maximization, the terms (γ) of the sum in (6.14) go to zero, and we obtain the original result by Fisher et al. [69]. Similarly, [169, Theorem 1] on constant factor suboptimal solvers follows after substituting the suboptimality into the cost model (6.8).

6.5.2 Bounding the cost of distributed planning for target tracking problems

This section provides tools for characterizing the cost of distributed planning (6.13) in target tracking problems. We begin by discussing decomposition of the objective as a sum over targets and derive a bound in terms of the result. Applying this bound produces a collection of weights that relate the cost of distributed planning to the channel capacities (6.7) between the robots and targets.

6.5.2a Decomposing objectives as sums

The objectives of sensing problems can often be written as sums is true for our target tracking objective (6.5) which is a sum over information sources (6.6). Specifically, let $\mathscr{G} = \{g_1, \ldots, g_{n_t}\}$ be a collection of set functions so that for $j \in \mathcal{T}$ and $X \subseteq \mathscr{U}$ then

$$g_j(X) = \mathbb{I}(\mathbf{X}^t_{j,t+1:t+l}; \mathbf{Y}_{j,t+1:t+l}(X) | \mathbf{Y}_{j,0:t}, \mathbf{X}^r_{0:t}). \tag{6.15}$$

This decomposition will enable characterization of interactions between distant robots in terms of each robot's capacity to sense near and distant targets.

Definition 8 (Sum decomposition). *A set of submodular, monotonic, and normalized functions $\mathscr{G} = \{g_1, \ldots, g_n\}$ decomposes a set function g if*

$$g(X) = \sum_{\hat{g} \in \mathscr{G}} \hat{g}(X), \quad \text{for all } X \subseteq \mathscr{U}. \tag{6.16}$$

Closure over sums [70] ensures that g is submodular, monotonic, and normalized if the same is true for each $\hat{g} \in \mathscr{G}$ as in Def. 8. Further, although some such sum decomposition always exists ($\mathscr{G} = \{g\}$), the choice of decomposition will determine the tightness of our performance bounds; here, we are interested in decompositions that express how interactions vary with distance. Choosing $\mathscr{G}$ according to (6.15) captures spatial locality arising out of distributions of robots and targets.

6.5.2b Derivatives and the sum decomposition

Given some $\mathscr{G}$ that decomposes g, the second derivative (3.16) at $X \subseteq \mathscr{U}$ with respect to $A, B \subseteq \mathscr{U}$, all disjoint, is

$$g(A; B|X) = \sum_{\hat{g} \in \mathscr{G}} \hat{g}(A; B|X) \tag{6.17}$$

and likewise for all other derivatives which are linear combinations of values of g at different points.

This derivative has the same form as the cost of ignoring prior decisions during distributed planning γ^{dist} (6.9). The rest of this section is devoted to obtaining upper bounds relating each robot's decision to the decisions they ignore (A and B) while eliminating the decisions each robot takes into account (X).

6.5.2c Bounding second derivatives using sum decompositions

Applying monotonicity and submodularity respectively provides a trivial upper bound on the second derivative of a set function

$$g(A; B|X) = g(A|B, X) - g(A|X) \geq -g(A|X) \geq -g(A) \tag{6.18}$$

where $A, B, X \subseteq \mathscr{U}$ are disjoint subsets of the ground set. By symmetry

$$g(A; B|X) \geq -\min(g(A), g(B)). \tag{6.19}$$

Then, expressing the second derivative of g in terms of the sum decomposition (6.17) and bounding the derivatives of $\hat{g} \in \mathscr{G}$ yields

$$g(A; B|X) \geq \sum_{\hat{g} \in \mathscr{G}} -\min(\hat{g}(A), \hat{g}(B)). \tag{6.20}$$

Remark. *Chapter 5 relies on $g(A; B|X)$ increasing monotonically in X, and we could make a similar statement here by writing the right-hand-side of (6.19) in terms of marginal gains with respect to X.*

More broadly, generalizing (6.20) to include any relevant lower bounds on second derivatives of $\hat{g} \in \mathscr{G}$ unifies the results we present here with those for 3-increasing functions in Chapter 5. This enables development of multi-objective applications such as for robots covering an environment while simultaneously localizing objects.

6.5.2d Quantifying inter-robot redundancy

Bounding the second derivative of g (6.20) leads to bounds on interactions between agents. The upper bounds on these interactions form a weighted undirected graph $\mathcal{G} = (\mathcal{R}, \mathcal{E}, \mathcal{W})$ connecting the robots with edges $\mathcal{E} = \{(i, j)|i, j \in \mathcal{R}, i \neq j\}$ and weights

$$\mathcal{W}(i, j) = \max_{x_i \in \mathscr{B}_i, x_j \in \mathscr{B}_j} \sum_{\hat{g} \in \mathscr{G}} \min(\hat{g}(x_i), \hat{g}(x_j)). \tag{6.21}$$

Evaluating the right-hand-side of the above expression is difficult as doing so involves search over the product of two robots' action spaces. To make evaluation of the weights tractable, relaxing this expression by taking the pairwise minimum of the maximum values of each objective component produces an upper bound in terms of the channel capacities (6.7) that avoids search over a product space

$$
\begin{aligned}
\widehat{\mathcal{W}}(i, j) &= \sum_{k \in \mathcal{T}} \min\left(C_{i,k}, C_{j,k}\right) = \sum_{\hat{g} \in \mathscr{G}} \min\left(\max_{x_i \in \mathscr{B}_i} \hat{g}(x_i), \max_{x_j \in \mathscr{B}_j} \hat{g}(x_j)\right) \\
&\geq \max_{x_i \in \mathscr{B}_i, x_j \in \mathscr{B}_j} \sum_{\hat{g} \in \mathscr{G}} \min(\hat{g}(x_i), \hat{g}(x_j)) = \mathcal{W}(i, j),
\end{aligned}
\tag{6.22}
$$

recalling that in (6.15) we chose $\mathscr{G}$ to decompose g by targets $\mathcal{T}$ so that the second equality follows from (6.7). As the terms of the sum are the smaller channel capacities, this bound captures the idea that if sensing quality decreases with distance so do interactions between robots.

Figure 6.2: Visualizations of eight and sixteen robots tracking same numbers of targets. Robots with dotted finite-horizon trajectories are blue and targets red. The background illustrates the sum of target probabilities at each grid space, increasing from purple to yellow.
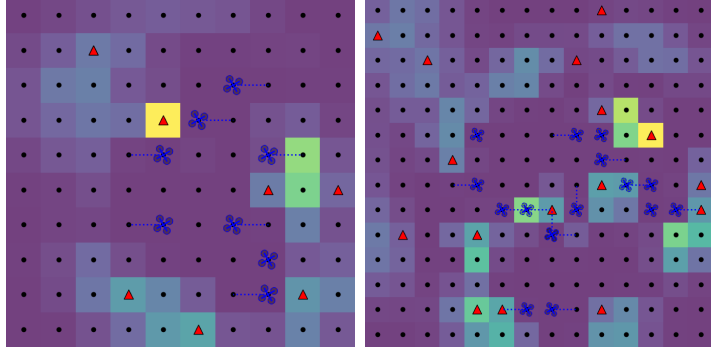
6.5.3 Summary of the proof of Theorem 10

Theorem 10 consists of two parts. The first, the effect of approximations on planning performance (6.12) follows by applying Lemma 11, on the suboptimality of general assignments, and substituting the definitions of the costs ((6.9), (6.10), and (6.11)). The second part (6.13) characterizes suboptimality due to distributed planning and follows by applying the chain rule to the definition of cost of distributed planning (6.9) and substituting (6.20), (6.21), and (6.22). Please refer to Appendix B.6 for the full proof.

6.6 Run time and scaling

Algorithm 4 requires a number of sequential planning steps that depends on the structure of the planner graph (Sect. 6.4.2) and a constant number of sequential steps for RSP planners. Yet, the sequential greedy algorithm (Alg. 2) requires one step per robot. Further, when the optimum is proportional to the sum of weights (ignoring the objective and planner costs) distributed planning *with a constant number of steps independent of the number of robots* can guarantee constant-factor suboptimality in expectation, approaching half of optimal (by extension of Theorem 7).

Spatial locality—more specifically ensuring that the robot-target channel capacities (6.7) decrease sufficiently quickly with distance—can enable robots to ignore distant robots and targets with bounded costs and provides optimization performance independent of the number of robots. Incorporating range limits in planning by ignoring robots and targets past a given distance and tracking targets using sparse filters can ensure that computation time for the single-robot planner is also constant. Likewise, regarding communication, robots send one message for each edge in the directed planner graph (Sect. 6.4.2), and ignoring distant robots reduces this to a constant number of messages per robot. Further, Appendix A.2 presents sufficient conditions for the cost of distributed planning γ^{dist} (6.9) to be bounded and so to provide constant optimization performance for any number of robots.

6.7 Results

To evaluate the approach, we provide simulation results (visualized in Fig. 6.2) for teams of robots tracking targets (one target per robot), robots moving according to planner output and targets via a random walk, all on a four-connected grid (with side length $\sqrt{12.5n_\text{r}}$). The robots estimate target locations using Bayesian filters given range observations to each target with mean $\hat{d} = \min(d, 20)$ and variance $0.25 + 0.5\hat{d}^2$ where d is the Euclidean distance to the target in grid cells. For the purpose of this chapter, all robots have access to all observations or, equivalently, have access to centralized filters. All simulation trials run for 100 time-steps; all initial states are uniformly random; and initial target locations are known. Additionally, all results ignore the first 20 steps of each trial to allow the system to converge to steady-state conditions.

Robots plan actions individually using Monte-Carlo tree search (MCTS, PLANANY-TIME) [39] with a two step horizon and collectively according to the specified planner. To ensure tractability we replace the original objective (6.5) with a sum of mutual information for each time-step

$$g^\text{sim}(X) = \sum_{i=1}^{l} \mathbb{I}(\mathbf{X}_{t+i}^\text{t}; \mathbf{Y}_{t+1:t+i}(X)|\mathbf{Y}_{0:t}, \mathbf{X}_{0:t}^\text{r}), \ X \subseteq \mathscr{U}. \tag{6.23}$$

This objective is equivalent to [156, (18)] and can be though of as minimizing uncertainty at the time of each planning step. Being a sum, (6.23) remains submodular, monotonic, and normalized so Theorem 10 still applies. Like Ryan and Hedrick [156], we evaluate this objective by simulating the system and computing the sample mean of the filter entropy. Because MCTS itself is sample-based, the planner estimates the objective implicitly by simulating the system once per MCTS rollout; by sampling the more valuable actions more often, MCTS produces increasingly accurate estimates for nearly optimal trajectories. Planners with sixteen or more robots use sparse filters with a threshold of $1e - 3$.

The experiments compare methods for multi-robot coordination including: sequential planning (Alg. 2, which has *one step per robot*); the proposed distributed planner (Alg. 4) with each robot assigned randomly to one of n_d sequential steps according to the RSP methods from Chapter 5; myopic planning (MCTS without coordination, *one step*); and random selection of actions. Additionally, we provide some results for *distributed planning with range limits* (R-lRSP) where robots plan while ignoring targets further than 12 units away (in terms of mean position) and other robots further than 20 units. Given the use of sparse filters, this latter planner runs in *constant time*.

We evaluate the distributed planning approach in terms of *task performance* (average target entropy) for various numbers of robots (Fig. 6.3), *objective values* on a common set of subproblems (6.3), and the *redundancy per robot* (6.22) for increasing numbers of robots which is proportional to $1/n$ times the bound on cost of distributed planning (6.13) for randomized assignment in n rounds (see Chapter 5). The results for average target entropy—which captures the uncertainty in target locations [58]—are based on 20 simulations of target tracking for each configuration. Results for objective values and redundancy use planning subproblems (6.3) taken from the simulation trials for distributed planning

96
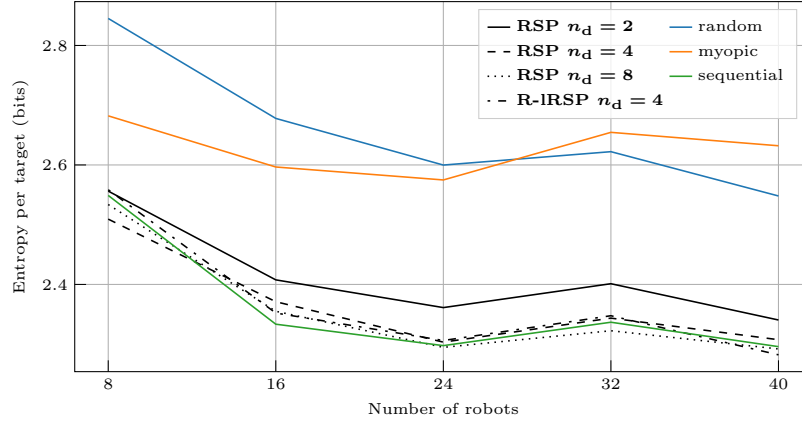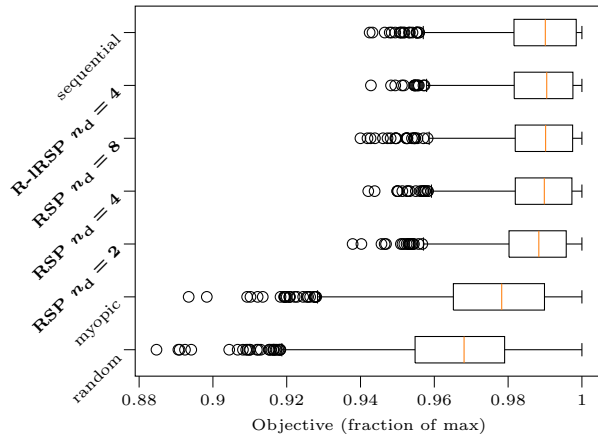
(a) Target entropy

(b) Objective values
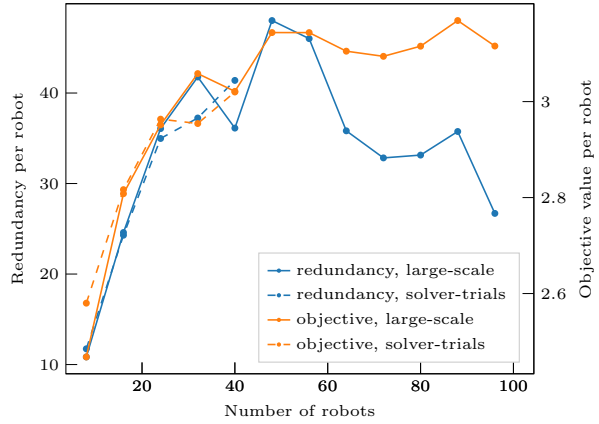
(c) Redundancy (6.22)

Figure 6.3: (a) For target entropy (lower is better) which is the task performance criterion distributed planning with RSP consistently improves upon myopic planning and approaches sequential planning given many times fewer sequential steps. (b) Objective values on common 16-robot subproblems reflect a similar trend, and results for distributed planning (bold) are effectively equivalent to sequential planning given only four planning rounds. (6.3). (c) Average objective value and total redundancy weight (6.22) per robot for RSP with $n_{\mathrm{d}} = 4$. Traces depict values for trials from (b) and five additional trials with R-lRSP reaching up to 96 robots. Values of each initially increase but appear to approach an asymptote which indicates that the distributed planner approaches constant factor suboptimality at large scales.

in four rounds. The results for objective values by solver are normalized according to the maximum values across solvers for each planning problem and reflect trials with 16 robots. For the results on redundancy, an additional five trials for a four-round distributed planner with range limits demonstrate behavior for up to 96 robots.[5]

Proposed distributed planners provide consistent improvements in target tracking performance (average target entropy) (Fig. 6.3a) compared to myopic planning; distributed planning in eight rounds roughly matches sequential planning despite requiring as much as five times fewer planning steps and produces 5–13% better (lower) target entropy than when planing myopically. The objective values (Fig. 6.3b) exhibit a similar trend, and all distributed planners closely match sequential planning, even in terms of the distributions of results.

Although the redundancy per robot (Fig. 6.3c) initially increases with the number of robots, that redundancy eventually levels off. This is consistent with the analysis of scaling performance in Appendix B.5 which indicates that the suboptimality approaches a constant factor bound. Overall, the results indicate that a small amount of coordination that does not scale with the number of robots is sufficient to produce performance comparable to sequential planning in receding-horizon settings.

6.8   Conclusions and future work

This chapter has presented a distributed planner for mutual information-based target tracking that mitigates the effects of the sequential structure of existing methods for submodular maximization. The analysis provides bounds on suboptimality for distributed planners that can be designed to run with fixed numbers of planning steps. By explicitly accounting for suboptimal local planning (e.g. anytime planning) and approximation of the objective, we affirm that the proposed approach is applicable to practical tracking systems. The results demonstrate that distributed planning improves tracking performance (in terms of target entropy) compared to planners with no coordination and that distributed planning with little coordination can even match fully sequential planning given a constant number of planning rounds.

Although we focus on target tracking, the analysis applies to general multi-objective sensing problems and generalizes the results in Chapter 5 on coverage. In this sense, we present a first positive result for mutual information objectives where the second discrete derivative is only nearly monotonic.

---

[5]Planning at this scale is intractable for other planner configurations.

Chapter 7

Time-Sensitive Exploration of Unknown Environments

The line of work that led to this thesis began with multi-robot exploration of unknown environments (Chapter 4), and we will now revisit this topic in greater depth. Obtaining improvements in performance for exploration—in terms of completion time—via submodular maximization had proven to be a challenge as the earlier results exhibited little variation in task performance across different methods for submodular maximization. The methods for analysis of redundancy that we have developed since then also do not immediately apply to mutual information objectives for exploration (see discussion in Fig. 5.2). We address the former through more thorough development and evaluation of the exploration system. Toward the same end, we also consider the design of objectives for robotic exploration. In doing so, we also address the theoretic limitations related to mutual information in order to apply our existing analysis for RSP planning to multi-robot exploration.

Unlike the target tracking problem that we studied in the previous chapter (Chapter 6), mutual information and submodular maximization on their own do not address a number of key challenges for multi-robot exploration. First, environments for exploration do not reflect the common assumption that cells are occupied with independent probability. Such cell independence assumptions are central to existing information-theoretic objectives [37, 103, 201] which often strongly emphasize efficient computation [37, 87, 118, 201]. Although these independence assumptions can be limiting, they admit changes to the occupancy prior for unobserved space (typically decreasing the prior) [87, 88, 179] (see also Sec. 4.6.1b). Still, even though recent works [118, 201] establish that Shannon mutual information can be evaluated efficiently and accurately for an individual range observation along a ray, methods for evaluating the joint contributions of multiple rays and camera views depend on approximation by summing over rays [37]. Our analysis highlights connections between mutual information and coverage objectives and suggests that coverage can accurately represent joint contributions when the prior probability of occupancy is low. Moreover, our results demonstrate that switching to a coverage objective can improve completion times by as much as 16%.

The same connections to coverage support applying RSP planning to provide efficient, distributed submodular maximization for exploration with large numbers of robots. To-

ward this end, we investigate the design of exploration systems and the potential impacts of methods for submodular maximization on the time to explore an environment. First, we introduce a reward for reducing distance to sufficiently informative views [57] which ensures reliable completion of exploration tasks and mimics methods that obtain similar effects via navigation toward frontiers [37, 198]. We also provide simulation results for more than a thousand simulation trials which compare methods for multi-robot coordination via submodular maximization for different environments and numbers of robots. The results demonstrate that sequential (greedy) and RSP planning substantially improve suboptimality for receding-horizon planning for multi-robot exploration. However, decreasing time to complete exploration tasks remains a challenge as the improvements in suboptimality translate primarily into increases in coverage rates early in the exploration process that are not often sustained through the end. Still, we remain optimistic that incorporating methods for planning and task assignment at larger spatial scales [134, 168] can realize sustained improvements in exploration performance.

7.1 Time-sensitive exploration problem

The following problem description parallels and reiterates prior discussion in Sec. 2.1 and Sec. 2.1.3 which include additional details on the properties of exploration and related problems.

Consider a team of robots $\mathcal{R} = \{1, \ldots, n_{\mathrm{r}}\}$ seeking to map a discretized environment $E = [c_1, \ldots, c_{n_{\mathrm{m}}}]$ consisting of cells $\mathcal{C} = \{1, \ldots, n_{\mathrm{m}}\}$ that are each either *free* (0) or *occupied* (1) (that is $c_i \in \{0, 1\}$ for $i \in \mathcal{C}$) according to some probability distribution. Each robot $r \in \mathcal{R}$ moves through the environment with state $\mathbf{x}_{t,r} \in \mathbb{R}^3$ at some *discrete* time t according to the following dynamics

$$\mathbf{x}_{t,r} = f(\mathbf{x}_{t-1,r}, u_{t,r}) \tag{7.1}$$

where $u_{t,r} \in \mathcal{U}$ belongs to a finite set of control inputs $\mathcal{U}$. Robots must remain in free space which induces the constraint that they remain in a safe set

$$\mathbf{x}_{t,r} \in \mathcal{X}_{\mathrm{safe}}(E). \tag{7.2}$$

The robots observe the environment with depth cameras (or other ray-based sensors such as lidar sensors). For the purpose of this chapter we assume that measurements are deterministic, without noise; robots can infer definitely that cells in the path of each ray are free up to the first occupied cell or the maximum range along the beam. For brevity, we abstract this process of inference and state that, at each time-step, robots observe sets of cells $F^{\mathrm{cam}}(\mathbf{x}, E) \subseteq \mathcal{C}$ with depth cameras and obtain their occupancy values via observations

$$\mathbf{y}_{t,r} = h(\mathbf{x}_{t,r}, E) = \{(i, c_i) : i \in F^{\mathrm{cam}}(\mathbf{x}, E)\}. \tag{7.3}$$

Through the process of exploration, robots seek to maximize the number of cells observed

$$J(\mathbf{X}_{1:t}, \mathbf{Y}_{1:t}) = \left| \bigcup_{t' \in \{1, \ldots, t\}, r \in \mathcal{R}} F^{\mathrm{cam}}(\mathbf{x}_{t',r}, E) \right|. \tag{7.4}$$

100

We refer to this quantity as the *environment coverage*.[1] Exploration is complete once the environment coverage reaches an environment-dependent quota $B(E)$ when

$$J(\mathbf{X}_{1:T}, \mathbf{Y}_{1:T}) \geq B(E), \tag{7.5}$$

and the robots seek to minimize the amount of time T required to reach that quota and complete the exploration process.

7.2 Planning for exploration

Like previous chapters, we propose planning with a receding-horizon approach whereby robots collectively maximize an objective g over a horizon with L steps

$$\max_{u'_{1:L,1:n_{\mathrm{r}}}} \quad g(\mathbf{X}_{t:t+L,1:n_{\mathrm{r}}})$$
$$\text{s.t. } \mathbf{x}_{t+l,r} \in \mathcal{X}_{\mathrm{safe}}(\mathbf{Y}_{1:t})$$
$$\mathbf{x}_{t+l,r} = f(\mathbf{x}_{t+l,r}, u'_{l,r})$$
$$\mathbf{y}_{t+l,r} = h(\mathbf{x}_{t+l,r}, E)$$
$$\text{for all } l \in \{1 \dots L\} \text{ and } r \in \mathcal{R}, \tag{7.6}$$

where $\mathcal{X}_{\mathrm{safe}}(\mathbf{Y}_{1:t})$ refers to the subset of the state space that is *known* to be safe given available observations (unlike (7.2) which refers to the complete set of safe states). After solving (7.6), robots then execute the first control actions in the sequence $u'_{1,1:n_{\mathrm{r}}}$.

Likewise, we can interpret g as a submodular, monotonic, and normalized set function and can rewrite (7.6) as a submodular maximization problem with a simple partition matroid constraint (Problem 4). From this perspective, the ground set consists of assignments of control actions to robots $\mathscr{U} = \mathcal{R} \times \mathcal{U}^L$, and the blocks of the partition matroid (Def. 6) are assignments $\{r\} \times \mathcal{U}^L$ to robots $r \in \mathcal{R}$.

We propose an objective that consists of two components g_{view} and g_{dist} so that, given some assignment of actions $X \subseteq \mathscr{U}$, then

$$g(X) = g_{\mathrm{view}}(X) + \sum_{r \in \mathcal{R}} g_{\mathrm{dist}}(X_r), \tag{7.7}$$

where X_r is the assignment to robot $r \in \mathcal{R}$. The *view* (or information) reward seeks to capture the value of observations from camera views over the planning horizon while the *distance* component provides reward for moving toward regions of the environment (beyond the planning horizon) where valuable observations can be obtained.

Note that g is normalized ($g(\emptyset) = 0$) will retain the monotonicity properties of g_{view} so long as g_{dist} is positive and zero when trajectories are not assigned because the distance terms are additive.[2]

---

[1]Previously, (in Chapter 4) we also used entropy to evaluate exploration performance. Appendix A.3 explains the reasoning behind this change.

[2]Being additive, marginal gains for the distance reward are fixed and do not depend on other robots. As such, aside from increasing monotonically, *all* higher-order monotonicity conditions hold because *all* second and higher-order derivatives (3.16) are zero.

7.3 Spatially local volumetric reward

The volumetric reward g_{view} seeks to capture the joint value of observations (camera views) in terms of the amount of unknown space that the robots will collectively observe. Ideally, g_{view} would correspond directly to the increment in environment coverage (7.4). Except, the environment E and, in turn, future values of the environment coverage are unknown.

To mitigate this issue, prior works predominantly either compute rewards given an assumed [20, 30, 62, 168] (or possibly learned) guess at the environment instantiation or else approximate information gain (i.e. mutual information (3.5)) for observations given a Bayesian prior [36, 37, 87, 103] (Chapter 4 applies such methods) so far *always* assuming independent (Bernoulli) cell occupancy.

As well as presenting the approach for this chapter, the following discussion unifies and contrasts different kinds of exploration rewards and demonstrates that our analysis for RSP planning applies to a variety of volumetric rewards.

7.3.1 Expected coverage

We will connect different exploration rewards in terms of expected coverage. Consider some possible distribution over possible environments $\mathcal{E}_{\text{guess}}$. The only restriction is that any environment that $\mathcal{E}_{\text{guess}}$ assigns non-zero probability $E' \sim \mathcal{E}_{\text{guess}}$ must be consistent with observations up to the current time $\mathbf{Y}_{1:t}$ ($\mathcal{E}_{\text{guess}}$ may even assign zero probability to all but one possible environment). Now, for convenience define the set of *future* states that robots will visit while executing a set of actions $X \subseteq \mathcal{U}$ as $\Phi(X)$. Given non-negative weights on cells $w_{\text{cell}} : \mathcal{C} \to \mathbb{R}_{\geq 0}$ the expected weighted coverage is

$$g_{\text{cov}}(X) = \mathbb{E}_{E' \sim \mathcal{E}_{\text{guess}}} \left[\sum_{i \in C_{\text{cov}}(X, E')} w_{\text{cell}}(i) \right] \tag{7.8}$$

where C_{cov} is the hypothetical set of cells that the robots may observe:

$$C_{\text{cov}}(X, E') = \bigcup_{\mathbf{x} \in \Phi(X)} F^{\text{cam}}(\mathbf{x}, E'). \tag{7.9}$$

An obvious weighting scheme, which we apply often in this chapter, is to provide a uniform (fixed) reward for each newly observed cells

$$w_{\text{new}}(i) = \begin{cases} 0 & i \in \bigcup_{t' \in \{1,\ldots,t\}, r \in \mathcal{R}} F^{\text{cam}}(\mathbf{x}_{t',r}, E) \\ 1 & \text{otherwise (for newly observed cells)} \end{cases}. \tag{7.10}$$

However, other weighting schemes are also relevant. For example, Yoder and Scherer [199] propose a system for inspecting surfaces. Their surface frontier approach would be similar in spirit to a scheme that provides increased weight to unobserved cells that are near

known occupied cells. Later, we will see that weighting cells by entropy produces a mutual information objective.[3]

Now let us discuss the properties of the expected coverage and the relationship to mutual information.

7.3.1a Expected coverage retains monotonicity properties

The expected coverage (7.8) is normalized and satisfies alternating monotonicity conditions (including being monotonic, submodular, and 3-increasing). This follows because these conditions hold for weighted coverage (Theorem 9)[4] and because the expectation forms a convex combination which preserves monotonicity conditions[5](and being normalized) [70].

Likewise, all the analysis for RSP planning in Chapter 5 applies to receding-horizon planning for exploration with expected coverage.[6]

Note that the expected coverage is not *necessarily* adaptive submodular [79]. See Appendix A.1 for more detail.

7.3.1b Well posed volumetric rewards

The previous section described how expected coverage has favorable properties for solving optimization problems. We now comment briefly on a property related to the exploration process. Specifically, the volumetric reward g_{view} should be non-zero if and only if the actual environment coverage (7.4) (given the true environment E) will increase after taking an observation. As such the robot will never be rewarded by g_{cov} for visiting a state where it will not observe any unknown cells or obtain zero reward when it will observe unknown cells.

A condition for this to hold for depth sensors is for the weight w_{cell} to be zero if and only if a cell $i \in \mathcal{C}$ has already been observed ($i \in \bigcup_{t' \in \{1,...,t\}, r \in \mathcal{R}} F^{\text{cam}}(\mathbf{x}_{t',r}, E)$). Note that the uniform weighting scheme (7.10) satisfies this requirement by construction.

This sanity condition follows, *specifically for depth sensors*, because each ray either:

1. Terminates (or reaches the maximum range) in known space for all environments that are consistent with prior observations, or

2. Terminates in unknown space, providing the occupancy value of at least one cell (even if the first unknown cell in the path of the ray is occupied).

---

[3]If not for mutual information, we would define the weighted expected coverage (7.8), concisely, in terms of unobserved cells and the weight (7.10) as a constant for all cells. However, probabilistic occupancy does not lend itself to explicit distinctions between known and unknown values.

[4]In fact, the same argument is valid for Theorem 9 which describes the probabilistic coverage objective.

[5]Moreover, the expectation is itself weighted coverage, covering a possibly exponentially larger set. This is evident by observing that the sum over weighted coverage functions is equivalent to duplicating the set being covered for each summand and adjusting weights according to probability.

[6]Note that none of the complications that arise in Chapter 6 apply to this chapter either for the 3-increasing condition or for the behavior of bounds on suboptimality for large numbers of robots, the latter because robots have clearly defined sensor ranges and are not observing objects with uncertain positions.

This observation is intuitive and not particularly interesting on its own. However, the consequence is that seemingly pathological assumptions, such as assuming that all unknown cells are occupied, can still produce sane rewards.[7]

7.3.2 Noiseless mutual information for depth sensors

Most existing works on mutual information for occupancy mapping assume noisy measurements via either a simplified (often Gaussian) [37, 201] or more general [103] noise model. However, Zhang et al. [201] find that the choice of information metric and noise model has little impact on performance in exploration experiments. Likewise, Henderson et al. [87] observe that the sensor noise for modern lidar sensors and depth cameras is typically small compared to the maximum range. For these reasons, we assume that sensor noise is negligible for the purpose of evaluating mutual information for exploration[8] and ignore sensor noise in this chapter. Additionally, prior works on mutual information for mapping [37, 103, 201] typically assume cell occupancy is independent according to the prior $\mathcal{E}_{\text{guess}}$ on the environment.

The combination of cell independence and lack of sensor noise produces a special case of an expected coverage objective:

Theorem 12 (Noiseless mutual information with independent cells is 3-increasing). *The mutual information $\mathbb{I}(E; \mathbf{Y}(X))$ between an environment E with uncertain occupancy and hypothetical future observations $\mathbf{Y}(X)$ can be written as*

$$\mathbb{I}(E; \mathbf{Y}(X)) = \mathbb{E}_{E' \sim \mathcal{E}_{guess}} \left[\sum_{i \in C_{cov}(X, E')} \mathbb{H}(c_i) \right]. \tag{7.11}$$

(7.8) given that:

1. *Cell occupancy probabilities $\mathcal{E}_{guess}$ are independent, and*
2. *There is no sensor noise.*

This expression (7.11) has the form of expected weighted coverage (7.8) and is therefore 3-increasing.

The proof is included in Appendix B.7. Note that Theorem 12 implies that our results for RSP planning for 3-increasing functions apply to noiseless mutual information just as for expected coverage. This holds even though mutual information is not 3-increasing in general (see Fig. 5.2).

Whether the sanity condition that Sec. 7.3.1b describes applies to mutual information depends on the updates to cell occupancy probabilities as observed cells would have to be marked as occupied with probabilities of either zero or one to ensure zero entropy and reward. However, this can be attributed to pedagogy as this chapter puts relatively limited weight on probabilistic models. Instead, we note that Julian et al. [103, Theorem 2.5] proves similar properties for mutual information objectives in general.

---

[7]This does not exclude the limiting case for weights w_{cell} approaching zero which will occur for mutual information objectives if an unobserved cell is marked as free or occupied with probability approaching one.

[8]Alternatively, note that sensor noise is likely more relevant to perception tasks such as surface reconstruction.

7.3.2a Limitations of existing approximations for computing mutual information

Our presentation of mutual information (7.11) (and also expected coverage (7.8)) does not yet address computational challenges. While either could be evaluated via sampling, doing so would significantly increase computational costs for systems that need to react quickly to operate effectively [63, 76]. For this reason, many works on information-based exploration [37, 87, 118, 201] emphasize computational contributions and approximate evaluation. However, relatively little is know about how design decisions and approximations can affect decision-making and exploration performance.

Moreover, Chapter 4 applied a CSQMI objective [37] while this chapter will apply a coverage-based objective instead. Our results for a small study indicate that this choice significantly improved exploration performance (by as much as 16%). However, we caution that we do not intend to establish conclusively that one objective is better than another and note that the optimistic coverage objective we compare to is itself a limiting case for mutual information objectives (see Sec. 7.3.3b). We believe that these connections between mutual information and coverage are useful for evaluating and improving mutual information objectives, and we highlight some of these points and provide additional results in Appendix A.4.

7.3.3 Remarks on specializing exploration objectives

Let us now expand on the prior two sections (Sec. 7.3.1 and Sec. 7.3.2) which define volumetric exploration objectives and their properties and discuss the ramifications of these observations for objective design.

7.3.3a Optimistic coverage

For the purpose of this chapter, we select a degenerate prior over environments by optimistically assuming that unobserved space is empty along with the uniform weighting scheme (7.10). This choice is similar in effect to numerous other works on exploration [30, 168] and is compatible with our definition and discussion of expected weighted coverage, despite failing to produce a meaningful distribution. Moreover, accurate evaluation of optimistic coverage objective is trivial, even for joint observations by teams of robots.[9]

7.3.3b Occupancy priors for mutual information and optimism

Mapping applications and even numerous papers on exploration [37, 103, 201] frequently assume that unobserved cells are independent and occupied with a probability of 0.5. Given a prior on occupancy of p, a beam will terminate after traversing $1/p$ cells which works out to *two* for a prior of 0.5. However, the environments that robots explore may, more

---

[9]While writing this thesis, optimistic coverage served an important role to characterize the performance of submodular maximization for exploration. Because the objective values for optimistic coverage are not approximate and provide oracle rewards in some cases, we can better attribute potential causes for deficiencies in exploration performance.

predominantly, consist of open space. For this reason, Chapter 4 and our prior work [179] both select priors with relatively lower occupancy probability.[10] Similarly, Henderson [88] provides detailed discussion and illustrations that demonstrate how the occupancy prior can affect decisions.

In the limit, a prior with low occupancy probability assumes that all unobserved cells are unoccupied like the optimistic coverage objective. In fact, *optimistic coverage is equivalent to mutual information* in this case after applying a scaling factor to normalize entropies of the unknown cells in (7.11) (and assuming entropies of observed cells are zero). This is a useful observation, because it provides a special case of the mutual information that can be evaluated exactly and efficiently.

7.3.3c   Learned models and oracle objectives

Another useful property is for the objective to provide rewards for observations given access to the environment E (i.e. ground truth) and thereby the true increment in the environment coverage (7.4). Comparing to an oracle is a useful tool for characterizing limits on performance as having an oracle provides the planner with additional information about the environment.[11,12] Prior works on problems related to exploration sometimes apply oracles similarly [158] while Choudhury et al. [48] use such oracles in a learning process to train an exploration policy.

Notably, *the optimistic coverage objective is an oracle for empty environments*, and we provide results for one such environment later in this chapter (Fig. 7.2).[13]

Likewise, a learned model may emulate an oracle [158, 190] and serve in a similar role as part of an expected coverage objective (with a degenerate prior) or a mutual information objective (while introducing a small amount of uncertainty). Our results where the objective is an oracle also provide insight into how well an exploration system with a learned model could perform.

7.3.3d   General distributions over environments

Finally, the prior $\mathcal{E}_{\text{guess}}$ might also encode a distribution over environments such as produced by a generative model where cells are not independent. Choudhury et al. [48] study expected coverage in a similar setting.

For such general priors, *mutual information and expected coverage may differ* because cell occupancy may not be independent. This leads to a philosophical issue for robotic exploration: *Should robots seek to observe portions of the environment which they are certain of but have not seen?* In this case, expected coverage encodes the affirmative (all

---

[10]Inspection reveals that some of this work [179] did not address the topic of priors in as much detail as remembered.

[11]Note that we still constrain the robot to operating in the known safe set $\mathcal{X}_{\text{safe}}(\mathbf{Y}_{1:t})$ despite having access to oracle rewards.

[12]An optimal policy for time-sensitive exploration (Sec. 7.1) would obtain strictly better performance with access to an oracle. Given that we do not have access to an optimal policy, actual results may vary.

[13]Our implementation of the optimistic coverage objective includes access to the bounding box for exploration so it is a true oracle for our Empty environment.
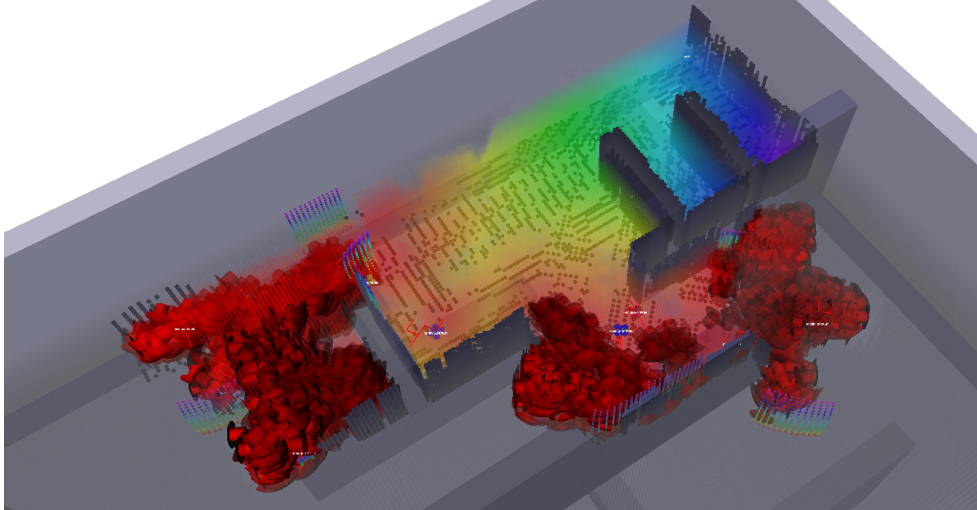
Figure 7.1: This visualization of sampled informative views in $\mathcal{X}_{\text{goal}}$ (7.12) (red arrows) and view distance (rainbow with red corresponding to least distance), demonstrates how sufficiently informative views, near the boundary of the unobserved space, (light gray) can play a similar role as frontiers for exploration.

unobserved cells are valuable), and mutual information the negative (only uncertain cells are valuable). Of course, this question is rhetorical, and answers will depend on the setting. However, we note that our analysis for RSP planning applies to the former, and the latter is a matter for future work.

7.4 Spatially global distance reward

In addition to being able to evaluate the value of nearby views over the duration of a planning horizon, robots should be able to reason about the value of visiting view points. A common approach is to navigate toward the nearest boundary of unoccupied and unknown space [198] (the frontier, Sec. 3.1.4) which is much like approaches that have robots navigate toward view points at or near frontiers [30, 36, 168].

The second term of the exploration reward (7.3) seeks to address this issue. Our distance reward g_{dist} is based on methods developed in our prior work [57]. This approach defines informative regions of the environment in terms of view value via a level set

$$\mathcal{X}_{\text{goal}} = \{\mathbf{x} \mid g_{\text{view}}(\mathbf{x}) \geq \epsilon_{\text{view}}, \, \mathbf{x} \in \mathcal{X}_{\text{safe}}(\mathbf{Y}_{1:t})\}. \tag{7.12}$$

where $\epsilon_{\text{view}} > 0$ is a threshold on view value. Rather than evaluating (7.12) exactly, we approximate this set by sampling and updating states in $\mathcal{X}_{\text{safe}}$ (which we call informative views) similarly as described by [57].[14]

Next, solving for shortest path distances between *each* point in $\mathcal{X}_{\text{safe}}$ to *any* point in $\mathcal{X}_{\text{goal}}$ produces a distance field $d_{\text{goal}} : \mathcal{X}_{\text{safe}} \rightarrow \mathbb{R}_{\geq 0}$. Figure 7.1 provides an example of such

---

[14]Some notable changes are that we replace the novelty threshold with thresholds on yaw and translation distance and only count new views toward the sample limit. We also compute distances over the entire grid to obtain more idealized results unlike our prior work which approximated values over a local sub-map [57].

(a) 80 robot-iters. **(b)** 448 robot-iters. **(c)** 816 robot-iters. **(d)** 1184 robot-iters.
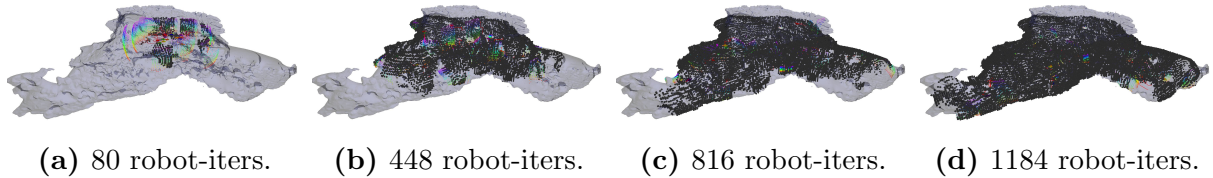
Figure 7.2: The images above visualize an example of the process of exploration of the Skylight environment with 16 robots and RSP planning with $n_{\mathrm{d}} = 6$. Additionally, a video providing examples of the exploration process for each environment and number of robots is available at: https://youtu.be/B9j8LVIs384

a distance field along with the sampled informative views. Reducing distances in d_{goal} then brings robots closer to valuable views in the goal region $\mathcal{X}_{\mathrm{goal}}$. This distance field can be computed efficiently (for low-dimensional problems as we do on a three-dimensional grid) using the fast marching method [183] which generalizes Dijkstra's algorithm.

The distance reward at time t is proportional to the greatest reduction in the path distance over the course of the trajectory

$$g_{\mathrm{dist}}(X_r) = \alpha \cdot \max_{l \in \{1,\dots,L\}} \big(d_{\mathrm{goal}}(\mathbf{x}_t) - d_{\mathrm{goal}}(\mathbf{x}_{t+l}) \big). \tag{7.13}$$

Note that this value is non-negative. Selecting the maximum along the trajectory is intended to provide sane rewards for when robots are near the informative region (7.12) and may exit the region by the end of the planning horizon.

Also, just as for frontier methods, this distance term provides the system with a weak notion of completeness. So long as there are states with observations worth a given value, robots will navigate to those states once local rewards from g_{view} have decreased sufficiently.

Finally, observe that the distance to the nearest view in (7.13) could be replaced with the distance to a specific goal location. This provides an avenue for incorporating methods for goal assignment (e.g. by solving a cardinality-constrained problem, Prob. 2) or scheduling which is common in other works related to multi-robot exploration [46, 134, 168]

7.5   Experiment design

The following section describes the design of the exploration experiments for this chapter. We provide results for 10 trials per each configuration and environment and for various numbers of robots (4, 8, 16, and 32). For intuition, Fig. 7.2 visualizes an example of the exploration process and provides a link to a video providing examples for all environments and numbers of robots.

7.5.1   Robot and sensor model

The robot dynamics and sensor model are the same as in the kinematic model in Sec. 4.6.1b. The set of control actions consists of $0.3\,\mathrm{m}$ translations in the cardinal directions with respect to the body frame as well as $\pi/2\,\mathrm{rad}$ yawing motions. Robots obtain observations from depth cameras with a range of $2.4\,\mathrm{m}$, a resolution of 19×12, and a field of view of

Planner	MCTS samples	c_p	Horizon (L)	View Value Threshold (ϵ_{view})	View Distance Factor (α)	Discount Factor
Sequential	200	1500	10	900	500	0.7
Myopic	200	1500	10	300	700	1.0

Table 7.1: Planner parameters for receding-horizon exploration. The myopic and sequential planners were tuned separately to maximize performance for 16 robots in the Boxes and Empty environments (Table. 7.2). All RSP planners use parameters for sequential planning. The parameter c_p belongs to the MCTS planner [27] and is set to roughly half the typical value of the objective for a single robot.

$43.6° \times 34.6°$, facing forward, oriented with the long axis vertical. Unlike Chapter 4, we *do not* downsample rays when evaluating mutual information and coverage objectives.

7.5.2 Single- and Multi-robot planning

Like prior chapters, robots plan by collectively solving receding-horizon planning problems (7.6)—here with the optimistic coverage objective Sec. 7.3.3a—and plan individually via Monte-Carlo tree search (MCTS) and collectively via some method for submodular maximization. For submodular maximization, we compare sequential planning (Alg. 2), myopic planning (wherein robots plan via MCTS and ignore others' decisions), and RSP planning[15] with 1, 3, and 6 rounds (n_{d}) except for the larger Office environment (see Sec. 7.5.3) for which we only provide results for $n_{\text{d}} = 6$.

We selected parameters by iteratively varying values of individual parameters in simulation trials in the Boxes and Empty environments. Parameters were selected separately for myopic and sequential planners with RSP inheriting parameters for sequential planning.[16] Table 7.1 lists the parameters for planning.

7.5.2a Discounting rewards

We also discount rewards and treat each robot as if having an independent probability of failure after each time-step. This discounting strategy produces a distribution over the states which robots will visit that is independent of the realization of the environment and is compatible with the theory for the objectives we study Sec. 7.3.1 and 7.3.2. Although we will not go into detail, evaluation of the optimistic coverage objective also remains straightforward.

Ideally, discounting prevents pathological behaviors where robots indefinitely put off rewards to a future time-step and would weight one robots' early horizon view above another's overlapping late horizon view (which is ostensibly more uncertain). However, preliminary experiments demonstrated relatively minor impacts on performance.

---

[15]Appendix A.5 compares suboptimality for RSP to DSGA (proposed for exploration in Chapter 4). Although DSGA generally outperforms RSP for a given number of planning rounds, RSP remains a more practical choice for distributed settings.

[16]Note that RSP$_1$ is equivalent to myopic planning but will use the same parameters as sequential planning so that any adverse impacts of parameter selection for the myopic planner will be evident.

7.5.3 Environments and simulation scenarios

The simulation results evaluate performance across a variety of environments (listed in Table 7.2). In each case, robots start with random yaw and slightly perturbed positions near a fixed starting location.[17] We determined maximum coverage values and the lengths of the simulation trials (iterations per robot) through longer preliminary experiments with a low view value threshold ($\epsilon_{\text{view}} = 100$) to encourage more complete exploration. Additionally, all maps use a $0.1\,\text{m}$ discretization so that a volume of $1\,\text{m}^3$ contains 1000 grid cells.

The environments that we study include synthetic environments that encourage different kinds of motions and behaviors, (e.g. upward motion is often slow because robots cannot easily observe space above them while moving upward) an empty environment which seeks to characterize maximal steady-state performance (and where optimistic coverage provides oracle rewards (Sec. 7.3.3c)), and more complex office-like and subterranean environments.

7.6 Methods for evaluation of results

The following describes how we evaluate the performance of the submodular maximization solvers (e.g. RSP) and overall exploration performance in terms of completion time.

7.6.1 Online bounds on solution quality

While most of this thesis focuses on obtaining bounds on solution quality for broad classes of problems, submodularity also produces certain online bounds (Sec. 3.5.3e, [132]) which can provide tighter guarantees for individual solutions [79, 109, 117]. These bounds also apply to any feasible solution which makes them suitable for comparing different kinds of planners.

Most works that we are aware of study these online bounds in the context of cardinality-constrained problems. Adapting these bounds to simple partition matroids produces some slight differences compared to the cited works which we present below. Specifically, maximization steps apply to blocks of the partition matroid instead of the ground set. Consider any—possibly incomplete—feasible solution $X \in \mathscr{I}$ to an instance of Problem 4 and an optimal solution $X^\star$. The following holds, respectively, by monotonicity, submodularity, and greedy choice:

$$g(X^\star) \leq g(X, X^\star) \leq g(X) + \sum_{x^\star \in X^\star} g(x^\star | X)$$

$$\leq g(X) + \sum_{r \in \mathcal{R}} \max_{x \in \mathscr{B}_r} g(x | X). \tag{7.14}$$

We apply two instances of the above bound. For the first, X is the full solution returned by the planner (assigning actions to all robots) which we refer to simply as the *online* bound

---

[17]Aside from the initial configuration, the planners also introduce stochasticity into the results.
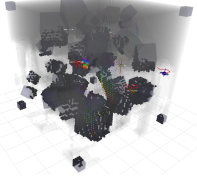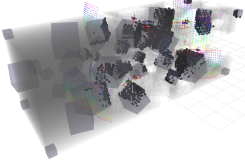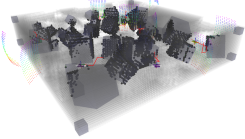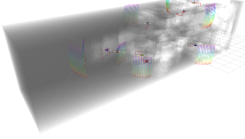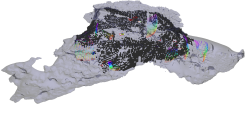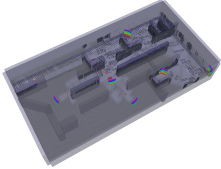
Image	Name	Bounding Box Volume	Exploration Volume	Description
	Boxes	$216\,\mathrm{m}^3$	$199\,\mathrm{m}^3$	Scattered boxes cause occlusions in a 6 m cube. Robots start offset at bottom and move *upward*.
	Hallway-Boxes	$217\,\mathrm{m}^3$	$202\,\mathrm{m}^3$	A rearrangement of the boxes environment into a 12 m square prism. Robots start at one end and move toward the other.
	Plane-Boxes	$227\,\mathrm{m}^3$	$212\,\mathrm{m}^3$	Rearrangement of the boxes environment into 2 m tall square planar configuration, and robots start in the center. This highlights performance in common, primarily two-dimensional environments.
	Empty	$500\,\mathrm{m}^3$	$500\,\mathrm{m}^3$	Robots start on one end of a 20 m hallway-like square prism that is completely devoid of obstacles which highlight steady-state performance in open space.
	Skylight	N/A	$220\,\mathrm{m}^3$	Mesh based on survey data [99, 100, 179, 196] from the Indian Tunnel skylight at Craters of the Moon National Park, scaled down to ∼35% of actual size. Robots start at the top of the mouth (now 7 m in diameter).
	Office	$1300\,\mathrm{m}^3$	$1180\,\mathrm{m}^3$	A simulated $36\,\mathrm{m} \times 18\,\mathrm{m} \times 2\,\mathrm{m}$ office environment. This environment almost maze-like in design, and the height effectively restricts robots to planar motion. Robots start in the upper right corner.

Table 7.2: This table lists details and descriptions for the different test environments. The *bounding box volume* provides the fixed volume of the bounding box for the experiment while the *exploration volume* lists the approximate maximum environment coverage volume for exploration. Images portray partially explored environments with unknown space (excluding subterranean environments) in gray and occupied in black.
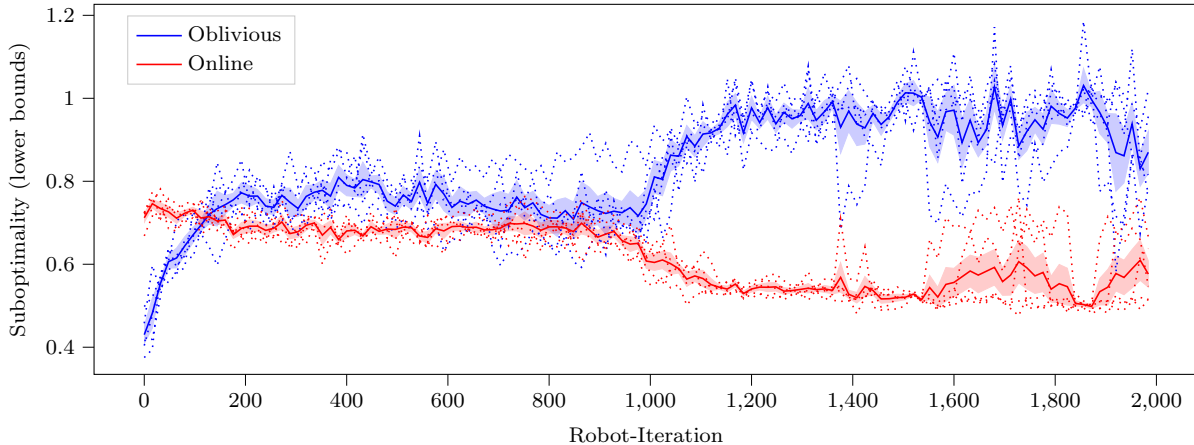
Figure 7.3: The above illustrates a representative example for the online and oblivious bounds on suboptimality (solution value over bound) with five trials for sequential planning and 16 robots, all starting near the *same position*. Note that the online bound is tighter early when robots are close together, and the oblivious bound becomes tighter later, as robots spread out, and more so even later, after robots have observed most of the environment (environment coverage nears its maximum at about 1000 robot-iterations for these trials). Note that this bound is not exact as we approximate maximization steps in (7.14) with MCTS. Shaded regions show the standard error, and dotted lines show individual trials.

because it uses the current solution to bound the optimal solution. Next, we call the case where $X = \emptyset$ is the empty set the *oblivious* bound[18] because this case can bound the optimal solution without planning. This oblivious bound reduces to $g(X^\star) \le \sum_{r \in \mathcal{R}} \max_{x \in \mathcal{B}_r} g(x)$ which would produce an optimal solution if g were additive (modular).

Typically, the online case is tightest for solutions where robots observe all nearby cells such as when robots are operating in close proximity (illustrated in Fig. 7.3).[19] On the other hand, the oblivious case is generally loose when robots are close together with overlapping fields of view and tightest when robots are spread out and observing disparate regions of the environment. Regardless, observe that the tightest bound typically exceeds 70% which is significantly tighter than the a-priori bound of 1/2 for sequential planning.

Later, we will use these bounds to characterize solution quality across trials with different planner configurations in lieu of comparison on common subproblems.

7.6.2 Exploration coverage rates, progress, and completion time

We evaluate task completion in terms of time (simulation time-steps which we refer to as iterations) to reach quotas for environment coverage (see Sec. 7.1). We provide results for quotas at 90% (completion the exploration task) and 30% (early progress in exploration) of the maximum coverage for each environment (Table 7.2). The latter seeks to provide a

---

[18]Note that (7.14) provides an upper bound on $g(X^\star)$ so we can characterize the suboptimality of other solutions by comparing to the right-hand-side.

[19]Intuitively, the online bound might also be tight late when there is little left to explore. However, Fig. 7.3 demonstrates that such behavior can be uncommon in practice, primarily (but not exclusively) due to the distance term in the objective (7.7).

measure that is more sensitive to variations in suboptimality of receding-horizon planning and submodular maximization and is less affected by challenges related to large-scale assignment and routing (which we address less thoroughly, Sec. 7.4). Note that all results for completion times provide statistics based on completion times for trials individually.

To facilitate comparisons across environments and for different numbers of robots, we also present results for completion time in terms of the coverage rates per robot, per iteration.

Of course, evaluating fractions of the maximum exploration volume to determine when the exploration task is complete would not be possible in practice for truly unknown environments. Instead, defining task completion for exploration as when the collection of sampled informative views (see Sec. 7.4) becomes empty—indicating that the robots are not aware of any more information-rich regions of the environment—may work well in practice.

7.7   Results

Figure 7.4 summarizes the exploration process from the results for the simulations in terms of environment coverage (7.4) and highlights the maximum coverage values (Table 7.2) and completion thresholds (Sec. 7.6.2). Although, there is not always much variation across planner configurations, these plots also illustrate consistency in coverage rates, graceful degradation in performance, and reliably complete exploration. The latter, reliable task completion, is evident in the asymptotic convergence and decreasing variance at the end of each trial. This is a product of the distance reward (Sec. 7.4), and we note that Corah et al. [57] provide additional results related to this feature. The rest of this section will go into more detail on the former points regarding time to completion.

7.7.1   Comparison of early and final completion by environment

Figure 7.5 illustrates how coverage rates vary as robots reach the early progress and completion thresholds for different environments and numbers of robots.[20] All environments exhibit some similar trends as would be expected: slowing exploration over time and with increasing numbers of robots. Across all this data, coverage rates vary from 120 to 630 (cells per robot per iteration) a factor of about $5\times$, and the Empty environment alone exhibits a variation of $3\times$. While not all of this slow-down is likely to be avoidable, there may be room to improve performance significantly through the end of each trial via more effective goal assignment and long-term planning. On the other hand, coverage rates for completing the exploration task are somewhat more consistent, excluding the Empty environment, and per robot performance varies by about $2.4\times$. Still, Plane-Boxes—where robots can spread out quickly to cover the volume—is the only environment where new robots consistently maintain coverage rates (and contribute to improving completion times). On the other

---

[20]This figure depicts results for RSP_6 which performs reliably and avoids the case for Sequential in the Hallway-Boxes environment in which not all trials completed the exploration task.
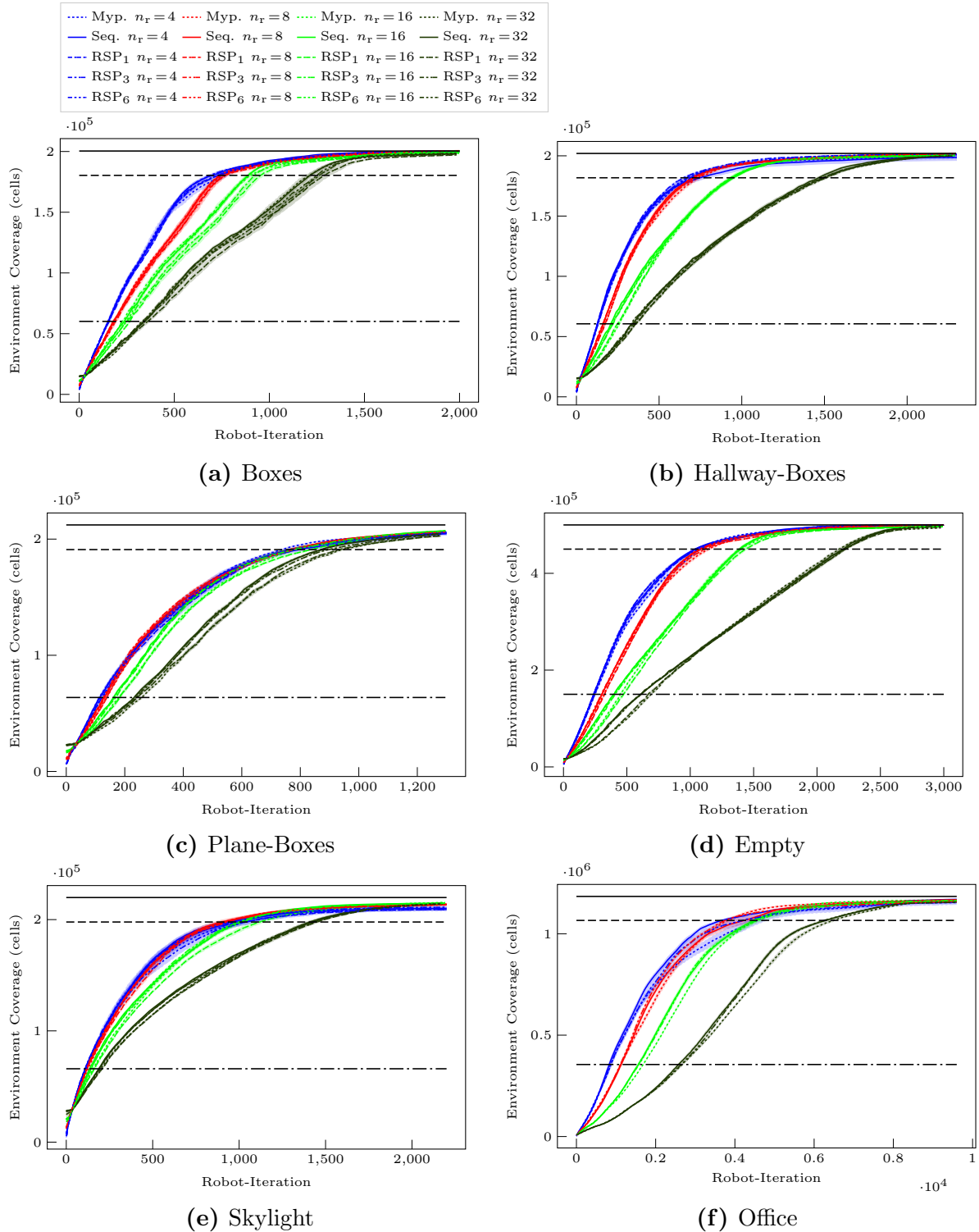
Figure 7.4: Environment coverage results for each environment. Shaded regions delineate the standard error. Black lines demarcate (from top to bottom) the maximum environment coverage, the completion threshold, and the early progress threshold. *These plots highlight general performance trends. Other plots better highlight differences between planners.*

(a) Early Progress Threshold

(b) Complete Exploration
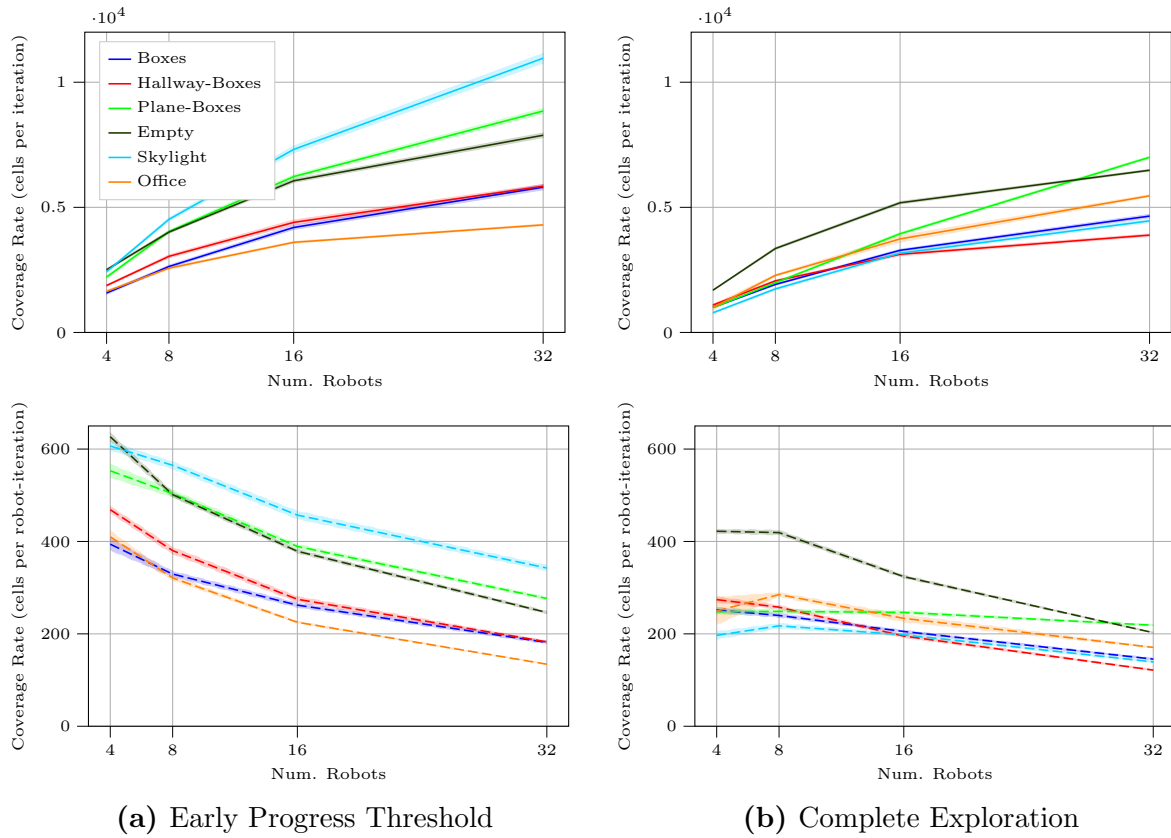
Figure 7.5: Coverage rates across environments up to (a) the threshold for early progress and (b) up to completing the exploration task both *per iteration* (top, solid) and *per robot per iteration* (bottom, dashed). In general, the exploration process slows after the beginning of each trial, and contributions per-robot decrease gradually as more are added.
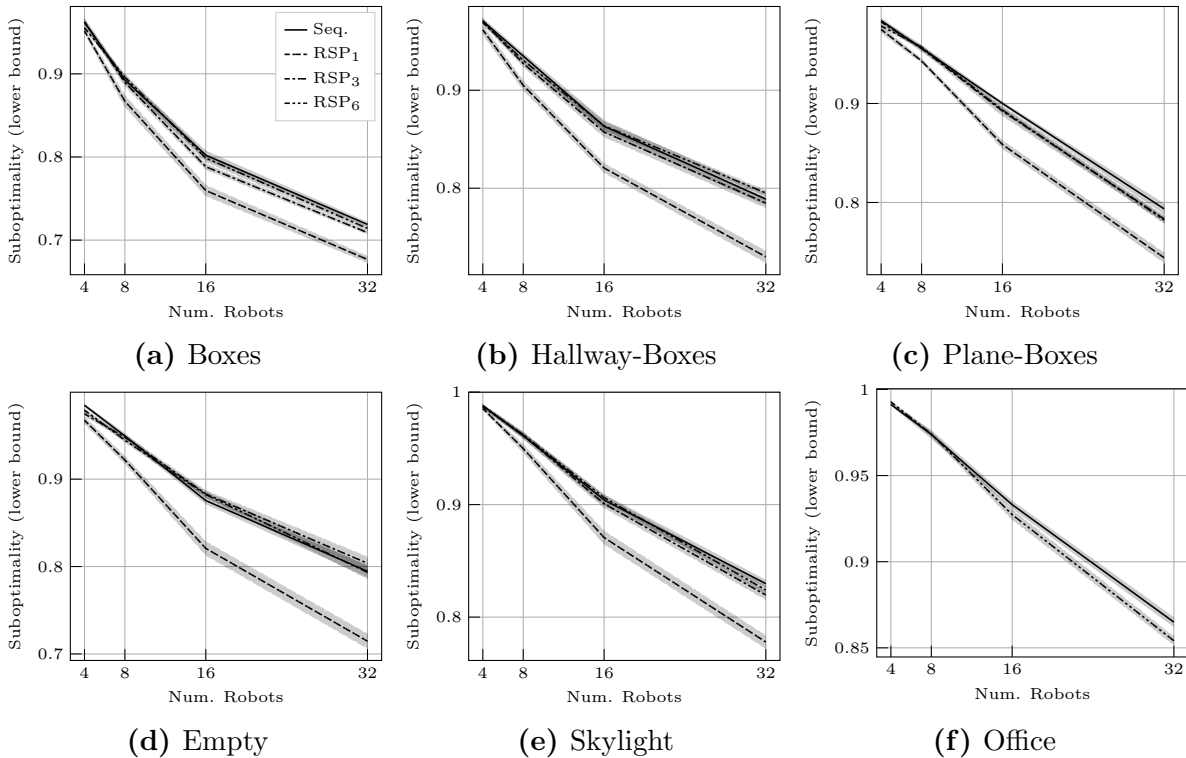
Figure 7.6: Lower bounds on suboptimality for receding-horizon planning for exploration in each environment. Plots provide mean values and standard error (*w.r.t.* the standard deviation of trial means) for data up to the completion time of each trial. These results demonstrate how RSP planning performance approaches that of sequential planning with increasing numbers of planning rounds from $n_d = 1$ to $n_d = 6$. We exclude the Myopic planner as the suboptimality bounds are not directly comparable to those of the other planners because the choice of parameters (Table 4.1) changes the objective values. Instead, RSP_1 also plans myopically but has parameters that are comparable to other planner configurations, and Table 7.3 includes data for all planners.

hand, robots slow down significantly after beginning exploration in the Skylight environment, possibly because the robots can spread out quickly in the central volume but slow down while covering the passages on either side.

Later, Sec. 7.7.3 (which also includes completion times in Table 4.1) will go into more detail on exploration times as a function of the method of planning for multi-robot coordination.

7.7.2 Planner suboptimality

Figure 7.6 presents results on mean values of the lower bound[21] on suboptimality (the greater of the two bounds in Sec. 7.6.1), and Table 7.3 lists this data as well. These plots clearly demonstrate how the suboptimality of RSP planning approaches that of sequential planning (Alg. 2) with increasing numbers of planning rounds (n_d) as the performance

---

[21]Bounds are computed approximately via MCTS (for single-robot planning) and are only representative of suboptimality in multi-robot coordination.

Num. Robot	Myopic		Sequential		RSP$_1$		RSP$_3$		RSP$_6$		Num. Robot	Myopic		Sequential		RSP$_1$		RSP$_3$		RSP$_6$	
	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.		Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
Boxes											**Empty**										
4	0.90	0.015	0.96	0.010	0.95	0.008	0.96	0.012	0.96	0.005	4	0.94	0.013	0.98	0.003	0.97	0.012	0.97	0.008	0.98	0.006
8	0.77	0.021	0.89	0.012	0.87	0.021	0.89	0.013	0.90	0.011	8	0.87	0.022	0.95	0.008	0.92	0.012	0.95	0.008	0.94	0.007
16	0.69	0.008	0.80	0.016	0.76	0.020	0.79	0.010	0.80	0.015	16	0.75	0.014	0.88	0.012	0.82	0.024	0.88	0.012	0.88	0.010
32	0.70	0.013	0.72	0.008	0.68	0.012	0.71	0.006	0.71	0.008	32	0.68	0.013	0.79	0.020	0.71	0.026	0.80	0.023	0.79	0.022
Hallway-Boxes											**Skylight**										
4	0.92	0.022	0.97	0.008	0.96	0.014	0.97	0.008	0.97	0.007	4	0.97	0.006	0.99	0.004	0.99	0.004	0.99	0.004	0.99	0.005
8	0.83	0.019	0.94	0.008	0.90	0.014	0.93	0.010	0.93	0.011	8	0.90	0.016	0.96	0.008	0.95	0.011	0.96	0.008	0.96	0.006
16	0.72	0.022	0.86	0.018	0.82	0.014	0.86	0.015	0.86	0.013	16	0.79	0.023	0.90	0.014	0.87	0.018	0.90	0.011	0.91	0.007
32	0.70	0.011	0.79	0.018	0.73	0.018	0.78	0.015	0.80	0.008	32	0.71	0.021	0.83	0.009	0.78	0.018	0.82	0.012	0.82	0.012
Plane-Boxes											**Office**										
4	0.96	0.007	0.98	0.006	0.97	0.010	0.98	0.004	0.98	0.007	4	0.98	0.006	0.99	0.001	–	–	–	–	0.99	0.002
8	0.88	0.011	0.96	0.009	0.94	0.006	0.96	0.007	0.96	0.004	8	0.95	0.007	0.97	0.007	–	–	–	–	0.97	0.005
16	0.76	0.016	0.90	0.004	0.86	0.012	0.89	0.015	0.89	0.007	16	0.88	0.011	0.93	0.006	–	–	–	–	0.93	0.009
32	0.69	0.008	0.79	0.016	0.74	0.015	0.78	0.011	0.78	0.010	32	0.78	0.016	0.86	0.008	–	–	–	–	0.85	0.007

Table 7.3: Lower bounds on suboptimality for receding-horizon planning for exploration in each environment. See Fig. 7.6 for more detail.

bounds for these planners would suggest[22] (Theorem 7). Moreover, we see that actual suboptimality is consistently better than the worst case bounds (even approaching one for fewer robots) which is consistent with observations from related works [79, 109, 117]. Likewise, the performance gaps widen with increasing suboptimality and larger numbers of robots (with the difference reaching 8% for in Empty). The decrease in these bounds with increasing numbers of robots is representative of robots operating in closer proximity and with overlapping observations over the planning horizon (see (7.6)).

7.7.3 Early progress and completion times by planner

The picture for planning performance becomes more complex when we look at the exploration process which is affected by factors such as the design of the exploration objective (7.7), (including the view and distance components) and the use of a receding-horizon with fixed trajectories (7.6). This is evident from Table 7.4 which lists statistics for time to completion and time to reach the early progress threshold (see Sec. 7.6.2). Most planners for a given environment and number of robots perform comparably both at the end of each trial and near the beginning.

The exception for complete trials is the Plane-Boxes environment where both the Myopic and RSP$_1$ planners perform worse for 32 robots (by about 6%). This is likely a product of how the small size and limited vertical mobility of the Plane-Boxes environment produces more frequent and complex interactions between robots.

Figure 7.7 compares performance across planners in terms of the rate of increase in environment coverage (via the data in Table 4.1). Although the variations in coverage rates are less significant than for suboptimality, the trends that the data does represent are consistent with expectations: the Myopic and RSP$_1$ planner configurations (which each involve planning myopically, robots ignoring others' decisions) perform worst whenever there are significant differences across planners (e.g. by 14% for 16 robots in the Hallway-Boxes environment).

The differences in these coverage rates also narrow somewhat going from 16 to 32 robots, unlike the trends for suboptimality (Fig. 7.6) where the performance gaps widen. The pro-

---

[22]Strictly, the bounds for RSP planning only establish convergence to the same worst case suboptimality as sequential planning (1/2), but we expect comparable suboptimality in practice.

(a) Boxes   **(b)** Hallway-Boxes   **(c)** Plane-Boxes

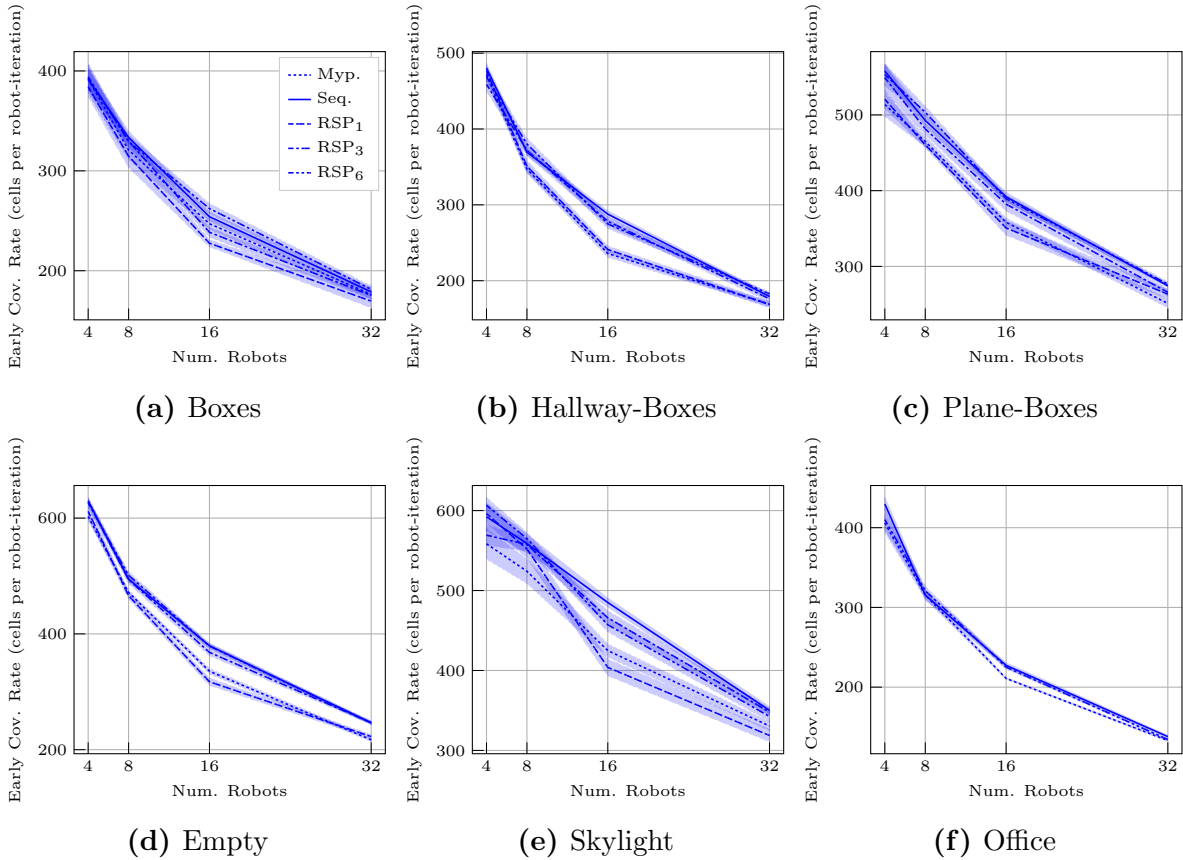**(d)** Empty   **(e)** Skylight   **(f)** Office

Figure 7.7: Coverage rates per robot up to the early progress threshold are generally similar across planners without many significant differences. Still, the differences that do exist consistently highlight deficiencies in the Myopic and RSP_1 configurations (each plans myopically with different parameters). Shaded regions depict standard error.

(a) Early progress times

Num. Robot	Myopic Avg.	Std.	Sequential Avg.	Std.	RSP$_1$ Avg.	Std.	RSP$_3$ Avg.	Std.	RSP$_6$ Avg.	Std.
Boxes										
4	155	19	155	13	157	12	154	13	154	19
8	189	16	181	11	193	20	183	9	183	12
16	244	17	237	12	265	12	253	15	230	14
32	343	30	338	24	359	41	344	21	332	18
Hallway-Boxes										
4	129	7	126	4	127	6	132	8	130	7
8	175	9	164	5	173	12	163	10	160	10
16	258	17	211	10	252	14	218	13	221	14
32	359	20	339	16	361	27	345	21	332	13
Plane-Boxes										
4	125	12	114	5	123	9	117	12	116	10
8	137	6	129	6	138	3	133	7	127	6
16	178	8	163	7	183	15	167	11	164	6
32	255	18	232	6	243	16	240	12	230	8
Empty										
4	249	9	238	4	246	17	239	7	240	11
8	318	7	304	8	322	11	304	10	300	11
16	448	19	396	13	474	29	408	19	396	14
32	692	29	609	18	675	31	609	19	610	22
Skylight										
4	119	13	112	10	111	7	116	8	109	5
8	127	12	118	8	120	8	118	6	117	5
16	156	9	136	5	164	13	142	8	145	9
32	200	11	189	7	208	15	190	8	193	11
Office										
4	881	74	830	50	–	–	–	–	874	90
8	1,128	55	1,130	60	–	–	–	–	1,110	60
16	1,685	25	1,566	70	–	–	–	–	1,580	49
32	2,672	88	2,578	87	–	–	–	–	2,646	63

(b) Completion times

Num. Robot	Myopic Avg.	Std.	Sequential Avg.	Std.	RSP$_1$ Avg.	Std.	RSP$_3$ Avg.	Std.	RSP$_6$ Avg.	Std.
Boxes										
4	751	85	698	51	733	38	689	52	721	55
8	761	33	742	30	767	73	736	16	754	42
16	900	29	886	24	946	58	921	39	880	36
32	1,198	56	1,214	46	1,276	86	1,254	65	1,243	56
Hallway-Boxes										
4	685	66	∞	–	676	58	648	37	666	47
8	737	58	701	48	709	34	725	56	707	59
16	942	38	945	43	941	40	935	46	934	53
32	1,495	55	1,457	55	1,491	67	1,507	53	1,497	58
Plane-Boxes										
4	737	38	805	119	784	65	777	87	774	62
8	762	38	766	46	768	42	777	32	771	47
16	785	32	758	30	834	33	775	41	776	35
32	949	31	872	23	932	35	900	30	873	19
Empty										
4	1,078	24	1,039	29	1,038	32	1,036	33	1,067	36
8	1,138	56	1,091	49	1,111	27	1,072	32	1,075	42
16	1,399	55	1,388	43	1,436	47	1,389	36	1,391	43
32	2,174	47	2,202	72	2,211	61	2,217	44	2,222	38
Skylight										
4	960	157	929	140	1,008	192	948	155	1,024	153
8	907	72	896	158	928	90	915	74		
16	1,014	58	1,012	42	1,124	104	1,012	80	1,002	53
32	1,418	90	1,415	60	1,452	52	1,434	41	1,423	66
Office										
4	4,675	1,162	3,696	312	–	–	–	–	∞	–
8	3,831	236	4,056	466	–	–	–	–	3,747	178
16	4,538	276	4,482	222	–	–	–	–	4,623	616
32	6,477	275	6,276	206	–	–	–	–	6,250	151

Table 7.4: Completion times (in robot-iterations) and times for reaching the early progress threshold for each environment and planner (see Sec. 7.6.2).

cess for parameter selection (Table 4.1) which focused on completion times for 16 robots partially explains this phenomenon. However, this may suggest that the receding-horizon optimization problems we formulate become less representative of actual performance with increasing numbers and crowding of robots. One possible cause is that the regions robots are planning to observe toward the end of their planning horizons are frequently being observed sooner by other robots. This produces uncertainty in future actions which assuming fixed trajectories does not account for but which might be alleviated by selecting a smaller discount factor.

7.8 Conclusion and future work

The system design and results from this chapter significantly improved on the earlier work in Chapter 4. Adding a term for distance to informative regions to the objective of the environment, ensured reliable task completion which would not be true for our prior results (Fig. 4.4). Likewise, employing a coverage-based objective also improved performance (by as much as 16%, see Appendix A.4). Incidentally, more mundane changes also contributed. Running experiments on a 16-core AMD Ryzen processor enabled us to dismiss approximations such as down-sampling rays and to perform more effective parameter tuning.

This chapter also overcame an important challenge that arose from this thesis by resolving limitations on applying RSP planning to exploration. This came about by demonstrating that mutual information on occupancy grids without noise is 3-increasing. Moreover, this ensures that the suboptimality guarantees for RSP apply to receding-horizon planning for exploration: constant-factor suboptimality, approaching 1/2, for a fixed number of plan-

ning rounds (n_d) and any number of robots subject to the extent of inter-robot interaction (via the pairwise weights). To reiterate prior chapters, this provides an $O(n_\mathrm{r})$ speed-up compared to sequential planning (Alg. 2) while approaching the same suboptimality guarantee. Additionally, while this chapter did not focus on distributed algorithms, the next will present a distributed implementation of RSP and application to similar exploration tasks.

Still, realizing significant improvements in performance via submodular maximization and RSP planning proves challenging. Our results narrow this gap by establishing significant improvements in suboptimality on the receding-horizon subproblems as well as improvements in coverage rates early in the exploration process.

7.8.1 Improving exploration performance

There are a few clear avenues for further reducing completion time and sustaining improvements in coverage rates.

Regarding solution quality, the online lower bounds suggest that, although solutions to the receding-horizon optimization subproblems predominantly outperform worst case bounds (1/2), they may still be far from optimal (as low as 70%). Iteratively improving solutions [7, 184] or developing an applicable variant of the continuous greedy algorithm [31] could each improve suboptimality. However, there is no guarantee that doing so would significantly improve exploration performance—this is evident from the large gaps in suboptimality for trials with 32 robots which do not translate into commensurate improvements in completion times.

The exploration process also often slows after the beginning of a trial. In some of these cases, better routing and goal assignment [134, 168] could alleviate this issue. Such approaches could be implemented within our framework by swapping the distance reward goal regions (7.12) with distances to specific goals.

Chapter 8

Implementation of Distributed, Receding-Horizon Planning for Exploration

While the last chapter focused on task performance in exploration, this chapter focuses on the implementation of a distributed, receding-horizon planner. The planner is synchronous and runs in real time in an anytime fashion. Moreover, the implementation is simple and does not depend on information about other robots, aside from the sensor data that those robots use to construct maps of the environment.

A number of works propose distributed or decentralized algorithms for online[1] multi-robot informative path planning problems [7, 18, 151, 160, 174] much like those that arise in this thesis. Fewer present results for planning in real time in simulation or with real robots [38, 160, 174] as we do in Chapter 4 and typically with no more than three robots. Furthermore, actual implementations may still be centralized, again in line with our earlier work [160]. On the other hand, Sukkar et al. [174] provide results for two robots in an agricultural active perception task with a distributed implementation of Dec-MCTS. With this chapter, we seek to provide an implementation of a distributed sensor planning system that is suitable for large numbers of robots.

Auction algorithms can also solve submodular maximization problems [194] and are perhaps more mature as Ponda [146] provides results for a task assignment problem with six robots with wireless communication for both aerial and ground robots and as many as six robots at once. Those results demonstrate a version of the Consensus-Based Bundle Algorithm (CBBA) [46].

At a high level, agents in auction algorithms like CBBA compute marginal gains (bids) and iteratively communicate sets of assignments to neighbors, seeking to come to consensus on the maximization steps of Alg. 1. We provide a comparison to analogous auction algorithms in terms of solution quality and messaging costs. While auctions converge quickly given full access to objective information, our results indicate that constraints on

---

[1]As opposed to offline (equivalently non-adaptive) problems where a plan is computed in entirety before execution time [169].

information access (so that agents cannot accurately compute rewards for others' actions) can slow convergence or harm solution quality for sensing and coverage problems. This is unlike the setting that Choi et al. [46] study which is more amenable to constraints on information access.

8.1 Background

Let us begin with a brief background on distributed algorithms and communication networks.

8.1.1 Distributed models of computation: synchronous and asynchronous algorithms

This chapter describes a partially synchronous implementation of R-lRSP based on message timing and with distributed memory [124]. For contrast, the directed graph structure of RSP also lends itself to asynchronous implementation. For asynchronous models, computation times, message arrival times, and order all may vary. An asynchronous version of RSP could be implemented by providing access to neighbors in the directed graph structure of the planner and making each robot wait to start planning until receiving decisions from all in-neighbors.

8.1.2 Communication typologies

Given a network of agents, there are many ways by which those agents may communicate. One the most common is *unicast* which we also refer to as *point-to-point* communication. For unicast communication, individual agents send messages to other individuals. On the other extreme, *broadcast* refers to sending messages from one agent to all others. The actual implementation we present effectively implements broadcasts via ROS. Communication neighborhoods for R-lRSP are also naturally amenable to multicast communication as the robots send identical messages to each of their neighbors.

8.1.3 Existing networking and communication systems

This chapter presents a distributed algorithm for implementation on teams of mobile robots. However, operation in subterranean and urban environments can limit communication between robots [138]. Designers may then enable communication between robots that are not immediately connected by establishing a mesh network over the team of robots.

One complication with this approach is that connectivity between robots can change as they move about the environment. Networks that address this challenge by allowing for links to change and reconstructing routes online are called mobile ad hoc networks (MANETs). One example of a MANET protocol that may be appropriate for use alongside the algorithm we propose is Better Approach to Mobile Adhoc Networking (BATMAN) [115]. However, we note that support for multicast communication appears incom-

plete, and implementations of RSP may be limited to unicast communication, depending on the underlying mesh network.

8.2 Distributed Planning for Exploration

This section describes a distributed implementation of RSP with a focus on application to exploration problems much like those in prior chapters. Toward this end, consider a team of robots $\mathcal{R} = \{1, \ldots, n_{\mathrm{r}}\}$ exploring some environment. The robots plan in a distributed manner and collectively solve receding-horizon sensing problems (Prob. 4) via methods for submodular maximization while simultaneously maintaining distributed representations of the environment. The following describes that distributed planner and its operation.

8.2.1 Distributed computation model and assumptions

The distributed algorithm in this chapter is based on a partially synchronous, distributed memory model. The implementation itself is timing-based [124], and robots solve receding-horizon submodular maximization problems (Prob. 4) in *epochs* with fixed start and end times. For the purpose of presentation, we assume that robots transmit messages reliably and instantaneously, potentially on some mesh network. Likewise, we assume robots have access to synchronized clocks; although this work does not address clock-synchronization, the time scales in this work (seconds) are much slower than the accuracy (milliseconds) that common methods such as the Network Time Protocol (NTP) can provide [131].

Regarding information access, robots have access to local models of the environment as in Sec. 3.6.1 and are only able to accurately approximate marginal gains for their own actions such as due to latency in updates from distant robots.

8.2.2 Maintaining the environment model

Robots have access to some local approximation of the environment θ_i e.g. an occupancy map. Maintaining models of the environment involves some sort of distributed perception system. If robots simply share sensor data, the number of messages sent is quadratic with the number of robots. Compressed representations [57] can alleviate this burden somewhat as well as modifying the system to avoid the need for a complete models.

8.2.3 Communication and neighborhoods

On the lines of the definition of the R-lRSP planner in Sec. 5.5.4, we assume robots communicate with a local neighborhood $\mathcal{N}_i^c$ for each robot $i \in \mathcal{R}$. This neighborhood may be based on metric distance or a number of hops in a communication graph.

Then, aside from maintaining the local environment model, the only access to (or knowledge of) the multi-robot team and mesh network that we assume is the ability to

communicate with these neighbors $\mathcal{N}_i^{\text{c}}$. In our analysis, we assume point-to-point messaging on the mesh network, with messages possibly traveling for multiple hops. However, the design is amenable to other modalities such as multicast.

8.2.4 Long term goals

The exploration system in this chapter also includes a view distance reward like the one Sec. 7.4 describes. To avoid growth in computation time, we compute the distance field only on a sub-map around the robot as in our prior work [57]. More generally, a distributed road mapping strategy would be appropriate and in line with recent works on robotic exploration [186, 195].

8.2.5 Implementation of a synchronous, distributed, receding-horizon RSP planner

Algorithm 5 describes the implementation of RSP. Robots solve instances of receding-horizon planning problems over the course of each epoch, once every Δ^{d} seconds, and planning for individual robots runs for $\Delta^{\text{p}} = \frac{\Delta^{\text{d}}}{n_{\text{d}}}$ seconds (less some time for slack in practice), recalling that n_{d} refers to the number of sequential planning rounds for the RSP planner.

At the beginning of each epoch, robots sample their assignments to planning rounds, uniformly at random (lines 8–9). Each robot $i \in \mathcal{R}$ then waits for the beginning of its respective round (line 10) and, meanwhile, listens for planned actions from robots in its neighborhood $\mathcal{N}_i^{\text{c}}$ and receives $X_{\mathcal{N}_i^{\text{in}}}^{\text{d}}$ (line 11). The robot then plans sensing actions (line 12) over a receding horizon in an anytime fashion—in our case via MCTS—given those received actions, its state $\mathbf{x}_i$, and its local representation of the environment θ_i. Finally, robots send their planned actions to their neighbors and execute their plans (lines 13–14).

The following sub-sections expound on the implementation and behavior of this distributed algorithm.

8.2.5a Message content

The messages that Alg. 5 uses to send and receive decisions consist of:
  1. The unique id of the sender
  2. The number of the current planning epoch
  3. A representation of the finite-horizon plan (e.g. the initial state and time and a sequence of action indices)
A typical decision message consists of about 130 bytes.

8.2.5b Minimalist design

The implementation of Alg. 5 differs from other distributed algorithms in this thesis (aside from being the only actual distributed implementation) because it avoids depending on

---

Algorithm 5 Synchronous, distributed implementation of Range-limited Randomized Sequential Partitions (R-lRSP) from the perspective of robot i

---

1: $n_e \leftarrow$ number of the current planning epoch
2: $n_d \leftarrow$ number of planning rounds
3: $\Delta^d \leftarrow$ duration for execution of the distributed planer
4: $\Delta^p \leftarrow \Delta^d / n_d$ (planning time per robot)
5: $\mathcal{N}_i^c \leftarrow$ Robots (neighbors) within communication range
6: $\theta_i \leftarrow$ belief (e.g. map) available to robot i
7: $\mathbf{x}_i \leftarrow$ state (e.g. position) of robot i

8: Wait for the start of the current planning epoch at time $n_e \Delta^d$
9: $d_i \sim \{0, \ldots, n_d - 1\}$ (randomly select one of n_d planning rounds)

10: At time $n_e \Delta^d + d \Delta^p$
11: RECEIVE: $X_{\mathcal{N}_i^{in}}^d$ (listen for action selections from nearby robots $\mathcal{N}_i^{in} \subseteq \mathcal{N}_i^c$)
12: $x_i^d \leftarrow$ PLANANYTIME$(\theta_i, \mathbf{x}_i, X_{\mathcal{N}_i^{in}}^d, \Delta^p)$ (plan given available time)
13: SEND: x_i^d to $\mathcal{N}_i^c$ (send plan with epoch n_e for neighbors in later rounds)
14: EXECUTE: x_i^d

---

objects related to the planning process. Instead, the implementation focuses on handling messages and scheduling planning times. As such, the planning step (PLANANYTIME) is simply a callback with messages and planning time as inputs and outputs.

We chose this design to simplify the process of augmenting existing single-robot sensor planning systems with distributed reasoning. The main limitation of this approach is that the user becomes responsible for tracking any statistics related to planning such as objective value or solution quality.

8.2.5c   Robustness to communication failure

Additionally, the *realization of the directed graph structure for the planning process* (as described in Sec. 5.3) *is implicit* given the set of decisions each robot receives. When a robot begins planning for itself, it simply uses the messages that are available at that scheduled planning time. Messages that arrive late or fail to arrive (such as due to a crashed robot or loss of communication in a subterranean environment) do not prevent or delay the planning process. Instead, failures in messaging contribute to increasing suboptimality according to the realization of the directed planner graph (as in Theorem 6 or Theorem 10). This non-blocking behavior is valuable, considering that a naive implementation of a sequential planner that waits to receive messages from previous robots will not produce a complete plan on a disconnected network or if there are communication failures.

8.2.5d   Toward preventing collisions between robots

The planner that we present in this chapter is also not aware of possible collisions between robots unlike the planner in Chapter 4. Moreover, the approach to preventing inter-robot collisions in Chapter 4 depends on sequential reasoning to determine which robots execute planned trajectories and which execute fallbacks. However, a planner could be designed that allows robots to update their decisions in parallel because:

- Any robot can commit to a new plan if doing so avoids collisions with other robots' old and new plans.

- Local and global maxima (with respect collision-neighbors)[2] given any ordering can always commit to new plans, using the maxima to break symmetry.

Such an approach would preserve most of the guarantees from Chapter 4 (e.g. liveness) with only local rules.

Alternatively, other collision-avoidance tools such as barrier methods [189] (which have also been specialized for aerial robots [121, 188]) can prevent collisions by augmenting planners that are not necessarily collision-aware. Barriers methods benefit from being minimally invasive (changing robots' plans only when necessary). At the same time, such changes may harm sensing performance along a robot's trajectory so further study on the topic of inter-robot collisions would be beneficial.

8.2.6   Characterizing timing and synchronization

Because messages contain the number of the planning epoch, robots are able to determine whether to accept messages (if they belong to the current epoch) or reject them (because they did not arrive in time for a previous epoch). This also provides a mechanism for evaluating whether timing mechanisms are working. Due to random assignment to planning rounds (line 9) and because robots should receive messages from prior rounds on time, the nominal message acceptance rate is[3]

$$\mathbb{E}\left[\frac{n_{\text{accept}}}{n_{\text{accept}} + n_{\text{reject}}}\right] = \frac{1}{2}\left(1 - \frac{1}{n_{\text{d}}}\right). \tag{8.1}$$

Tracking whether the empirical acceptance rate matches expectations then enables designers to determine whether the planning system is respecting timing constraints and whether latencies in messaging are preventing messages from being received in time.

Note that this is distinct from rejecting messages that are outside the communication range. Hypothetically, a robot might also receive and ignore messages from a distant robot but at later times due to multi-hop communication. Such messages would then have to be excluded from these statistics on acceptance rates (based on timing).

---

[2]A *collision-neighbor* is a robot that is near enough that some pair of trajectories over the finite horizon could be in collision.

[3]This calculation includes unnecessary messages sent during the last round. Omitting those messages would require adaptation of this formula.

8.3 Asymptotic behavior for messaging

To begin, the number of sequential message transmissions—which we refer to as the communication span—for Alg. 5 is constant ($n_\mathrm{d} - 1$ or n_d if including messages which are sent during the last round and so not used). The total number of messages sent is in $O(n_\mathrm{r}n_\mathrm{n})$ where $n_\mathrm{n} = \max_{i \in \mathcal{R}} |\mathcal{N}_i^\mathrm{c}|$ is the largest number of communication neighbors over all robots. The total number of message hops in a mesh network depends on the length of the longest shortest-path d_n from any robot i to a communication neighbor $j \in \mathcal{N}_i^\mathrm{c}$ and comes to $O(n_\mathrm{r}n_\mathrm{n}d_\mathrm{n})$. The total communication volume, considering the number of decisions being sent, is the same. The numbers of messages, message hops, and communication volume are then constant per-robot if n_n and d_n are bounded.

In general, distributed perception is more expensive. In the worst case, robots may share all sensor data (or summaries thereof) and exchange $O(n_\mathrm{r}^2)$ messages per unit time. Then, if the diameter of the mesh network (the length of the longest shortest-path between any two robots) is d_m, the number of message hops and communication volume come to $O(d_\mathrm{m}n_\mathrm{r}^2)$

Mechanisms such as compressed representations [53] can reduce these communication costs. Further improvement may be possible by maintaining spatially local models, but some sort of global modeling is necessary in the framework we present such as to compute distances to long-term goals or views (Sec. 8.2.4).

8.4 Results

The results for this chapter include two studies. The first set of results (Sec. 8.4.1) characterize communication costs in relation to other possible distributed solvers in a setting based on the coverage experiments in Chapter 5. This supports the second set of results (Sec. 8.4.2) which demonstrate the synchronous, distributed implementation of RSP (Alg. 5) via a simulation of exploration with dynamic aerial robots (as in Chapter 4) which runs in real time but in a setting where communication costs are trivial.

8.4.1 Communication networks and messaging study

This study seeks to characterize communication costs for distributed submodular maximization on a mesh network in comparison to other relevant approaches. The submodular maximization problems are based on the coverage problems in Sec. 5.7.2. The objective of these problems is to maximize area coverage over a unit square. Actions cover circles with radius r_s and are distributed around agent centers within a radius of r_a. The radii are set so as to normalize the sum of the areas of all sensing actions for a given number of agents, as described in Table 5.1. The simulations vary the number of agents from 10 to 100 in increments of 10, and we provide results for 50 trials for each configuration.

Agents communicate decisions on an undirected communication graph with edges between any pair of agents within a radius of $r_\mathrm{c} = 3r_\mathrm{a}$. Due to the choice of sensor and agent radii, this ensures that there is an edge between any two agents that may have non-zero

redundancy. Unlike the experiments in Chapter 5, we require agents to form a connected communication graph (to avoid complications with comparisons to other solvers). To generate such agent positions, the position of the first agent center is sampled randomly and the rest by sampling a random agent and a position within range of that agent and inside the unit square.

The solver configurations compare R-lRSP to sequential planning (Alg. 2) and two auction algorithms (which converge to results equivalent to Alg. 1). The results for R-lRSP encompass planning with different numbers of rounds $n_d \in \{4, 8, 16\}$. To reduce suboptimality and simplify comparisons, the communication range for R-lRSP is set to one step (or r_c). As such, the cost of ignoring decisions outside the communication range (5.18) is zero, and agents only communicate with their immediate neighbors during each round of the planning process. For these results, we assume that agents have access to their communication-neighbors' planning rounds[4] d_j for $j \in \mathcal{N}_i^c$ and only send messages to agents within range and *in later rounds* (whereas Alg. 5 includes communication with *all* agents within range). For the sequential planner, agents plan in the order they were generated. Messages travel along shortest paths, and we take advantage of the structure of the assignment process to consolidate decisions into individual messages sent between adjacent agents in the planning sequence. The auction algorithms are adapted from CBBA by Choi et al. [46] in order to be applicable to general submodular objectives. These two auction algorithms—detailed in Appendix A.6—differ in terms of information access: the *global auction* algorithm (Alg. 6) requires full access to the objective g and the ability to re-evaluate marginal gains for other agents' actions; and for the *local auction* algorithm (Alg. 7) agents only compute values of their own actions, and they communicate assignments in lists along with the corresponding marginal gains. While the global auction algorithm will converge faster, this comes at the cost of a strict requirement that all agents have complete and consistent access to the objective[5] *which violates the assumptions on information access* (Sec. 8.2.1). The local auction converges more slowly but respects constraints on information access for local models. These auction algorithms represent two extremes. Specialized implementations might interpolate between the two such as by tracking whether assignments change within a given distance of each agent, somewhat like the specialized update rules for CBBA. In the spirit of online planning, we also provide results for early stopping after a given numbers of steps (matching the numbers of rounds for R-lRSP) as well as for planning to convergence.[6]

Figure 8.1 plots objective values and communication costs for varying numbers of agents. The numbers of messages include each hop on the communication graph. However, only the sequential algorithm sends messages that travel for multiple hops. The communication volume then weights messages and hops by numbers of decisions being sent (ignoring the marginal gains that the local auction algorithm also sends). The span is the number

---

[4]This could be implemented by sharing random seeds used to generate the assignments to planning rounds.

[5]Issues related to inconsistencies in the objective can prevent convergence which is an important topic for related works on auction algorithms [98].

[6]We state that an auction algorithm has converged once all agents have a complete set of matching assignments.

of sequential message transmissions or hops so that R-lRSP$_4$ has a span of 3, and the span for auction algorithms with early stopping is at most as such.

While these results reflect a large variation in the number of agents, the smallest communication radius is still nearly half the length of the unit square. This may explain the slow growth in spans (Fig. 8.1b) for the global auction algorithm. Likewise, the large communication radii cause messaging costs to increase quickly compared to asymptotic rates for R-lRSP and both auction algorithms (due to increasing numbers of neighbors) and more slowly for sequential planning (as paths between sequential agents are short). Overall, R-lRSP maintains constant span, small communication volume,[7] and consistent performance in terms of objective values and requires roughly eight rounds to approach objective values for the global auction solver. Interestingly, the convergence times for global auctions grow slowly, but this may be a product of small network diameters[8] in our results. The global auction solver may then be appropriate for sensor planning in similar settings if designers can accept greater communication costs and strict constraints on information access. On the other hand, the local auction solver better reflects our assumptions on information access, but early stopping significantly harms objective values.[9] These results also exclusively address communication; even when either auction algorithm converges more quickly, the robot whose action is assigned last completes maximization steps for each prior assignment and incurs a computational cost equivalent to sequential planning for the entire team (Alg. 2). Conversely, for R-lRSP each robot completes only a single maximization step.

8.4.2 Anytime distributed planning for exploration

Having, addressed the question of communication costs, let us now evaluate the distributed planner in the context of simulated multi-robot exploration (with Fig. 8.2 visualizing the exploration process for these experiments). For these results, the robot and sensor models are identical to the distributed model in Sec. 4.6.2. Likewise, individual robots plan with Monte-Carlo tree search [27, 39] with the same motion primitive library. Aside from the distributed implementation, the primary differences in the exploration system, compared to Chapter 4, are that planners do not check for inter-robot collisions, inclusion of a view distance term (see Secs. 8.2.4 and 7.4), and use of the optimistic coverage objective (Sec. 7.3.3a). The distributed, receding-horizon planner (Alg. 5) runs in real time in an anytime fashion with an epoch duration of 3 seconds and a horizon of 4 seconds. Because the epoch duration is constant, robots planning with greater numbers of rounds (n_d) have decreasing amounts of time to plan for themselves. The simulation results include 30 trials with 8 robots and planning with distributed RSP for 1, 2, and 3 planning rounds. The

---

[7]The auction algorithms produce large communication volumes and numbers of messages because all agents plan and communicate during each communication round and because agents communicate sets of decisions rather than individual decisions like RSP.

[8]The diameter is the length of the longest shortest-path between any two agents.

[9]The distributed exploration system exhibited in Sec. 8.4.2 would also violate the requirements of the global auction solver, simply because the robots query the maps at different times so that the models for planning are not entirely consistent with each other.
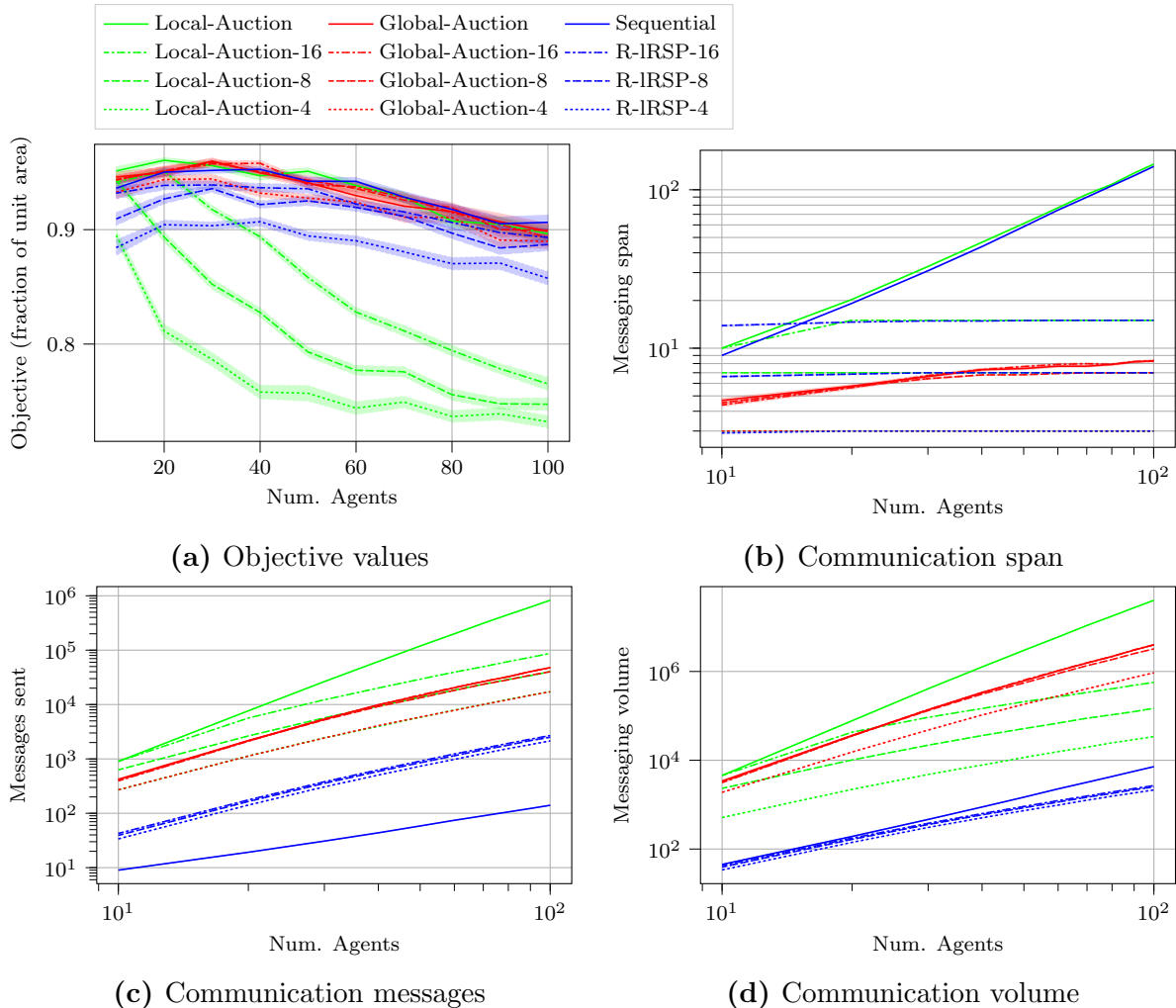
(a) Objective values

(b) Communication span

(c) Communication messages

(d) Communication volume

Figure 8.1: Objective values and messaging statistics for various distributed submodular maximization algorithms. Shaded regions (although small) show standard error. (a) Objective values (out of one) remain consistent with increasing numbers of agents, except for the local auction solver which degrades significantly when querying solutions before convergence. RSP requires eight or more communication rounds to obtain performance comparable to the sequential solver or global-information auction. (b) When considering the span (number of sequential message hops), only RSP obtains constant values, but the global-information auction solvers still retain a small spans, peaking at slightly over nine. (c) Numbers of messages (including hops) and (c) total message volume (by numbers of decisions and hops traveled) all grow super-linearly as the smallest communication radius (0.48) covers nearly half the length of the environment. While sequential planning benefits from being able to consolidate messages, RSP obtains the smallest communication volumes and consistently outperforms all auction planners by about at least an order of magnitude on both metrics.

130

(a) 27 seconds   **(b)** 418 seconds   **(c)** 809 seconds   **(d)** 1200 seconds
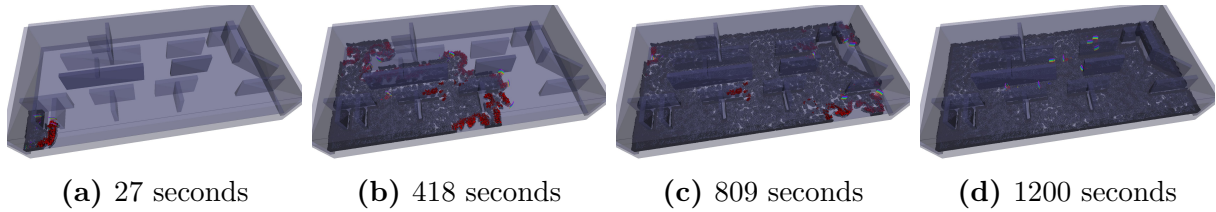
Figure 8.2: The above images visualize exploration of an office environment with eight robots and RSP planning with $n_d = 3$. Time stamps are approximate. A video of the exploration process can be found at: https://youtu.be/MMr9NxT_J8c
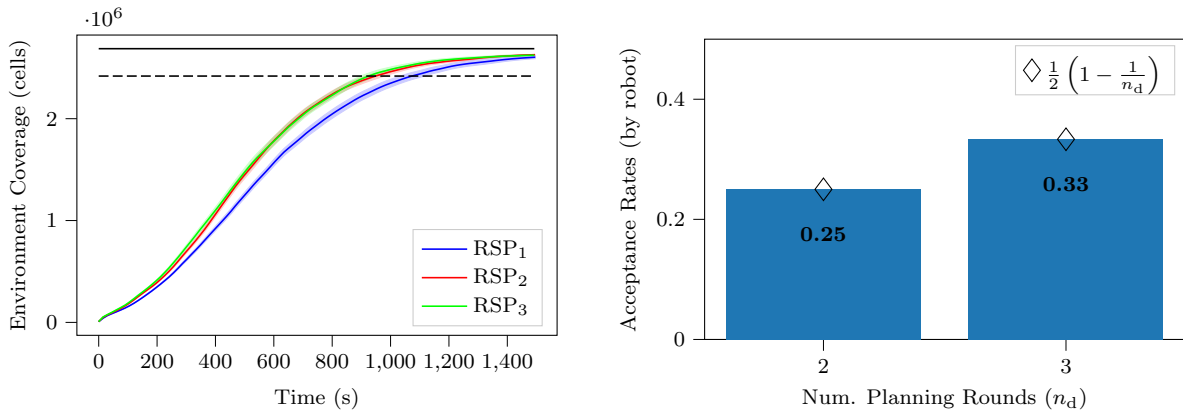
(a) Environment coverage   **(b)** Acceptance rates
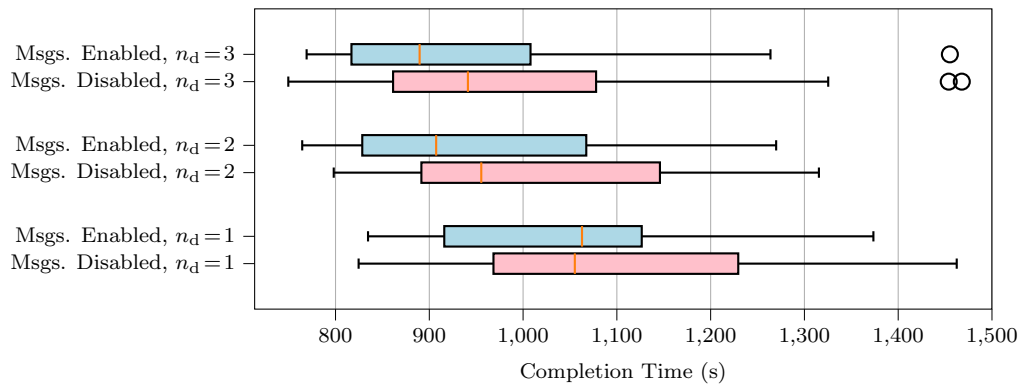
(c) Completion times

Figure 8.3: Results for exploration with the distributed RSP implementation. (a) Mean environment coverage and standard error (shaded regions) for different numbers of robots. Black lines demarcate the maximum environment coverage and a 90% threshold for task completion. (b) Message acceptance rates closely match predictions (8.1) which indicates that the synchronous execution of anytime planning rounds is functioning properly. (c) Planning with two or three rounds (n_d) improves median completion times by about 5% compared to the same planners after disabling communication of control actions. Note that the cases for $n_d = 1$ effectively represent the same configuration.

131

simulations in the Office environment pictured in Table 7.2. The mesh is scaled to 166% of the size in Chapter 7 so that the longer dimension is now 60m, and the exploration volume[10] is 2687m^3.

The implementation of the distributed planner takes advantage of ROS [148] for messaging, and planners for each robot run in separate processes (nodes). The simulations themselves run on a single desktop computer with a 16-core Ryzen 2950X processor. While the processor provides ample capacity to run computation in parallel, communication costs are not well-represented. Providing projected communication costs can ameliorate this issue somewhat. Regarding messaging span and latencies, the largest span (for $n_d = 3$) produces a span of two. Even significant latency, say 0.1s, produces a time cost of 0.2s which is small compared to the total planning time (three seconds). The implementation of the distributed planner does not include range limits on communication so that robots effectively communicate via broadcast. However, there is no barrier to including range limits, particularly at larger scales. Recalling the discussion of message content in Sec. 8.2.5a, the total cost of a communication round (assuming point-to-point communication between all robots) for eight robots is 7.1KiB or, conservatively, at most 7.1KiB s^{-1}. The simulated robots also maintain maps by assimilating camera data from the entire team. Still, all planning runs on spatially local sub-maps, and our approach is compatible with methods for distributed mapping that can significantly reduce communication costs such as via Gaussian mixture models [57, 178, 180]. Extrapolating from results for Gaussian mixture mapping [57, Fig. 7],[11] maintaining distributed maps for eight robots, again by point-to-point communication, would require a total of about 400KiB s^{-1} of communication volume.

Figure 8.3 plots results for environment coverage, message acceptance, and task completion. The synchronous planner functions as designed (Fig. 8.3b) with message acceptance rates matching predictions (8.1). This is consistent with the requirement that robots have access to decisions from prior rounds for RSP planning and subsequently to apply suboptimality guarantees for RSP (Theorem 7). In our implementation, robots query their maps when they begin planning, at the beginning of the assigned round rather than the beginning of the epoch. Because of this the effective latency can vary, the average decreasing with increasing n_d. To address this, we ran a second set of experiments with *communication for* RSP *disabled* which isolates the impact of distributed planning. This narrows the gap in task completion time to about 5%. Still, the scope of these simulation results is narrow as the intent is to demonstrate functionality of the distributed planner rather than to characterize task performance (which is the aim of Chapter 7).

8.5   Conclusion

This chapter has demonstrated a distributed implementation of RSP for receding-horizon sensor planning in the context of multi-robot exploration. The design is simple, requires

---

[10]The discrepancy in exploration volume is a product of differences in handling height. Here, robots have a height limit of 0.6m without a specific bounding box while Chapter 7 provided a 2m tall bounding box.

[11]In these results [57], three robots produce about 0.02MiB s^{-1} of data or 6.8KiB s^{-1} per robot.

minimal knowledge of the network structure, and is naturally robust to communication failures. Doing so realizes one of the main goals of this thesis, making sensor planning for large numbers of robots tractable by reducing the growth in planning time for sequential planners (or conversely the increasing division of fixed planning time) to a constant number of rounds with fixed duration. As the exploration experiments focused on computation in a setting where communication is relatively trivial, we also included a numerical study that inspects communication costs. This study demonstrated that R-lRSP can provide a significant reduction in communication compared to likely competitors, auction algorithms. Moreover, our approach can also provide good solution quality with only locally consistent models while auctions struggle to do so under time constraints. Again, our results demonstrate distributed, receding-horizon planning in real time in simulation with eight robots, significantly more than for comparable works, and we hope that this work can serve to enable further development of sensing systems with large numbers of robots.

Chapter 9

Conclusions and Future Work

This thesis has developed and advanced methods for receding-horizon sensor planning for teams of robots. Specifically, receding-horizon planning for problems such as exploration involves reasoning about likely observations at different camera views, reasoning about overlaps and redundancy between views, and collectively optimizing trajectories and views for teams of robots.

Existing works are able to address some of the challenges related to solving these problems via greedy algorithms for maximizing submodular objectives—submodularity being a common property amongst many sensing objectives [69, 169]. However, these algorithms are not particularly amenable to planning in real time for large numbers of robots, particularly in distributed settings: numbers of computation and communication rounds both grow linearly with the number of robots.

Seeking to reduce computation time for submodular maximization problems leads to results indicating that constant-factor computation time (more specifically, adaptivity, Sec. 3.6.2) and solution quality cannot co-exist in general [11, 75, 82]. This led us to develop Randomized Sequential Partitions (RSP) and methods for analysis of pairwise redundancy for multi-robot sensing problems. In Chapter 5, we identified 3-increasing functions—which include general coverage objectives—as the class of functions where redundancy between robots' actions decreases monotonically given others' decisions. Later, we applied this result for 3-increasing function to a case of mutual information for exploration in Chapter 7 via interpretation of the mutual information objective as expected coverage. Further, in Chapter 6, developing similar methods for redundancy analysis for sums of submodular functions enabled application of RSP for target tracking problems, demonstrating consistent task performance for as many as 96 robots in simulation results. Finally, we described a distributed implementation of RSP in Chapter 8 and provided simulation results for exploration in real time with eight robots.

9.1 Future work

This thesis work opens up numerous directions for future work which can be divided roughly between submodular function maximization and active sensing for robotics.

9.1.1 Submodular function maximization

This thesis also did not thoroughly characterize classes of 3-increasing objectives or more general classes that satisfy bounds on redundancy such as sums of submodular functions. For example, we did not identify any meaningful objectives that are 3-increasing but do not satisfy alternating monotonicity conditions as coverage objectives do.

Additionally, although we focused on greedy algorithms, the continuous greedy algorithm [31] could improve solution quality. While applying the continuous greedy algorithm to sensing problems, particular informative path planning problems with large spaces of actions, may prove challenging, there is interest in applying such methods in robotics [154]. Our methods for redundancy analysis could possibly be used to develop versions of such algorithms that converge with small numbers of integration steps.

Continuous analogues of submodular functions have also been a recent topic of interest [19, 154, 197]. Reformulating our results for 3-increasing functions in continuous domains could produce interesting results. However, the nature of possible impacts in this area are unclear.

Future works may also view this text as providing existence proofs that characterize certain sensing and submodular maximization problems. For example, a corollary to our results is that the adaptive complexity (Sec. 3.6.3) of a wide variety of receding-horizon sensing problems is $O(1)$. Likewise, our results could be interpreted as providing upper bounds on the amount of information access that is necessary to achieve a given bound on solution quality; doing so could inform design of more complex optimization systems or machine learning methods.

9.1.2 Active sensing

To begin, this thesis studied two sensing problems: exploration and target tracking. However, RSP methods are applicable to a broad variety of sensing problems. An interesting property of the problems we studied is that equilibrium conditions generally perform well without explicit coordination (i.e. with myopic planning): so long as robots communicate observations to each other, they tend to distribute themselves evenly across the environment, automatically. Other sensing problems, one being target coverage, may not exhibit such equilibrium behavior and may benefit more significantly from coordinated sensor planning.

Online adaptation of the planner structure was discussed in Chapter 5 but not revived afterward. Moreover, selecting the number of rounds produces a potentially challenging tradeoff between planning time and solution quality. Development of impactful methods for adaptation of RSP planning, particularly for planning in real time as in Chapter 8 is attractive. Ideal numbers of planning rounds may also vary spatially, and this may pose a challenging problem, particularly for planning in real time.

While Chapter 6 applied results for sums of submodular functions to target tracking, the methods can also apply more generally to multi-objective sensing problems. The analysis for redundancy could capture changes in dependency between robots through the course of a complex task. For example, robots searching for the source of a gas leak in a building

may experience large amount of redundancy that decreases after they localize the leak and focusing, instead, on a mapping task with more local interactions.

The applications in this thesis also focused on receding-horizon planning *with fixed trajectories*. However, the problems we solve could be addressed more directly as variants of POMDPs. Satsangi et al. [159] recursively apply greedy algorithms to obtain guarantees for certain submodular sensing processes. Similar results could be obtained for partition matroids in multi-robot problems, and the contributions of this thesis could be applied as well, possibly to develop efficient, distributed solvers for active perception processes.

In Chapter 7, we provided some incidental contributions toward objective design for exploration, and in kind with Henderson et al. [87], we identified some limitations of prior works on mutual information objectives. In one such direction, the interpretation of mutual information without noise as expected coverage could produce more effective approximations of joint mutual information (see Appendix A.4). The comparison to CSQMI in same appendix also strongly suggests that improving approximations for joint mutual information could be fruitful in terms of task performance. Such approximations should also be compared to recent work on approximating conditional mutual information [87].

As a whole, the methods for planning in this thesis are limited by the focus on receding-horizon planning. Further, realizing consistent improvements in task performance via RSP may require better and more complete planning methodologies. Chapter 7 cited spatially global routing and assignment—such as via distributed auctions—as likely directions for improving performance. Likewise, development of methods for planning with distributed road maps would be an important step toward improving our methods for computing view distance rewards.

Appendix A

Additional Technical Discussion

A.1 Expected coverage and mutual information for exploration with independent cells are not necessarily adaptive submodular

Although this thesis does not take advantage of adaptive submodularity, the work was developed in the wake of a number of works that apply adaptive submodularity to active sensing problems in robotics [45, 48, 96]. For this reason, considering whether and how adaptive submodularity might apply to problems such as exploration is important to this thesis.

In simple terms, adaptive submodularity [79] seeks to apply the concept of submodularity (and subsequent suboptimality guarantees) to adaptive settings—wherein agents obtain realizations of actions or observations (e.g. whether a measurement succeeded or the realization of a actual camera view) at each step of a greedy decision process (as in Alg. 1). In our work, the receding-horizon planning sub-problems which we formulate (Sec. 2.1.2) are *non-adaptive*, and the theory we develop primarily applies to the sub-problem solutions. However, *the decision process by which the robots replan and explore is adaptive*, and we do not provide guarantees for this process as a whole.

Methods based on adaptive submodularity provide the prospect of general suboptimality guarantees for the entire exploration process. Moreover, Choudhury et al. [48] study an exploration problem with an expected coverage objective equivalent to (7.8) and cite routing constraints (e.g. path planning) as the primary theoretic limitation for applying adaptive submodularity to exploration (versus existing results for cardinality constraints [79]). However, we note that they only *assume* that the objective is (or is approximately) adaptive submodular. Although this is true for some special cases [79, Sec. 7.1], such exploration objectives are not adaptive submodular in general which we will prove by counterexample after introducing adaptive submodularity more formally.

Consider a function $g : 2^{\mathscr{U}} \times \mathcal{O}^{\mathscr{U}}$ where $\mathcal{O}$ is a set of observations or outcomes. Here, $X \subseteq \mathscr{U}$ encodes the set of selections and $\Phi \in \mathcal{O}^{\mathscr{U}}$ encodes the outcomes for each observation. Further, agents receive an observation whenever they execute an action, and there is
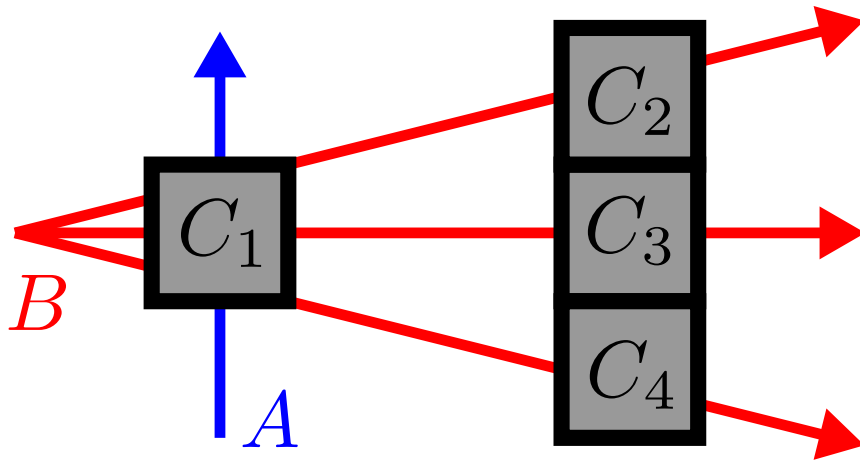
Figure A.1: The above illustrates a counter-example which demonstrates that mutual information for ranging observations with independent cells is not necessarily adaptive submodular. The target variable, the environment, consists of four cells $E = [C_1, \ldots, C_4]$ which are each independent and *free* (0) or *occupied* (1) with probability 0.5 (each cell with 1 bit of entropy). Two available observations B and A provide distances to the nearest occupied cell along a given ray, as illustrated.

some probability distribution Φ over the outcomes for each action in $\mathscr{U}$. Then, following Golovin and Krause [79], we can define the expected marginal gain as

$$\Delta(x|\psi) = \mathbb{E}\left[g(\mathrm{dom}(\psi) \cup \{x\}, \Phi) - g(\mathrm{dom}(\psi), \Phi) \mid \Phi \sim \psi\right] \tag{A.1}$$

where $x \in \mathscr{U}$, ψ is a partial realization that consists of action-outcome pairs, $\mathrm{dom}(\psi)$ is the associated set of actions in $\mathscr{U}$, and $\Phi \sim \psi$ indicates that the full realization Φ is drawn conditional on the partial realization ψ. A function g is adaptive submodular *with respect to the distribution over outcomes* $\mathcal{O}^{\mathscr{U}}$ if for all ψ, $\psi' \subseteq \psi$, and $x \in \mathscr{U}$, then

$$\Delta(x|\psi) \geq \Delta(x|\psi'). \tag{A.2}$$

In other words, the expected marginal gains for all actions must decrease *regardless of the outcome of a selection.*

Now, let us move on to the counterexamples for mutual information[1] and expected coverage.[2] Note that we now use more traditional notation for mutual information since that already includes mechanisms for encoding outcomes for observations and expected marginal gains which are equivalent to the ones Golovin and Krause [79] describe.

Theorem 13 (Noiseless mutual information for depth sensors with independent cells is not necessarily adaptive submodular)**.** *Consider an environment E which consists of Bernoulli cells that are each free or occupied with some probability and possible observations $\mathscr{U}$ from a depth sensor which provide ranges to the nearest occupied cell along collections of rays according to descriptions in Chapter 7 and Sec. 7.3.2.*

The mutual information $\mathbf{I}(E; X)$ for $X \subseteq \mathscr{U}$ is not adaptive submodular.

[1]The fact that mutual information is not adaptive submodular in general is well-known [80]. We prove that the same holds more narrowly for exploration and as part of the proof for expected coverage.

[2]Note that this does not preclude some cases being exactly or approximately adaptive submodular as in the work of Choudhury et al. [48].

Proof. Consider the scenario that Fig. A.1 illustrates and the resulting distribution over environments E and ground set $\mathscr{U} = \{A, B\}$.

The mutual information between B and E will violate adaptive submodularity. Now, note that B either observes the value of only one cell (if C_1 is occupied) or else all four. As such, the mutual information between the two is initially

$$\mathbb{I}(B; E) = 1 + 0.5 \cdot 3 = 2.5 \tag{A.3}$$

given the expected information gain (7.11) and the entropies of the cells. However, the mutual information for B increases if we observe A and determine that C_1 is free

$$\mathbb{I}(B; E | C_1 = 0) = 3. \tag{A.4}$$

This increase in mutual information violates adaptive submodularity (A.2) which completes the proof. ∎

Corollary 13.1 (Expected coverage for exploration is not necessarily adaptive submodular). *Expected coverage as described in Sec. 7.3.1 is not necessarily adaptive submodular.*

Proof. This follows from Theorem 13 because noiseless mutual information with independent cells is an expected coverage objective (Theorem 12). ∎

Still, there is still room to apply adaptive submodularity or similar properties to exploration policies. In fact, theory for objectives based on the size of the hypothesis space still applies [45, 79, 96, 97]. Further, Gupta et al. [85] provide results for a routing problem in a similar setting.

However, reducing the size of the hypothesis space has its limitations. Consider the reduction in the hypothesis space h and the following hypothetical bound for a greedy maximization process:

$$h^{\mathrm{g}} \geq \alpha h^{\star} \tag{A.5}$$

For a uniform prior over n hypotheses, the relationship between a reduction in the hypothesis space h and the information gain $\mathbb{I}$ is

$$\mathbb{I} = \log_2 \left(\frac{n}{n-h} \right) = \log_2(n) - \log_2(n - h), \tag{A.6}$$

and

$$h = n - 2^{\log_2(n) - \mathbb{I}}. \tag{A.7}$$

Substituting into the bound

$$\mathbb{I}^{\mathrm{g}} \geq \log_2 \left(\frac{n}{n - \alpha(h^{\star})} \right) \tag{A.8}$$

141

Consider if $h^\star = n$ (such as for mapping an entire environment) so that

$$\mathbb{I}^{\mathrm{g}} \geq \log_2\left(\frac{1}{1-\alpha}\right) \tag{A.9}$$

which works out to *one bit* for $\alpha = 1/2$ and *1.44 bits* for $\alpha = 1 - 1/e$.

In general, we can expect bounds on the reduction of the hypothesis space to be most useful for small hypothesis spaces (such as to distinguish between a small number of likely environments predicted by a learner or localizing objects [96]). However, such bounds would be less impactful for the exploration problems that we study which exhibit large information gain and exponentially hypothesis spaces.

A.2 Analysis for scaling target tracking to large numbers of robots

The analysis in this section establishes sufficient conditions for the cost of distributed planning γ^{dist} (6.9) for each robot to be constant (in expectation) for planners with a fixed number of sequential steps, independent of the number of robots. Afterward, we discuss how this analysis relates to the design and analysis of target tracking systems.

Consider a distribution of robots and targets on $\mathbb{R}^n$ with at most α robots and β targets on average per unit volume. Then assume that the channel capacities (6.7) between each robot $i \in \mathcal{R}$ and target $j \in \mathcal{T}$ satisfy a non-increasing upper bound $\phi : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ (possibly in expectation) so that $C_{i,j} \leq \phi(||\mathbf{p}_i^{\mathrm{r}} - \mathbf{p}_j^{\mathrm{t}}||_2)$ where $\mathbf{p}_i^{\mathrm{r}}$ and $\mathbf{p}_j^{\mathrm{t}}$ are the robot position and target *mean* position in $\mathbb{R}^n$. Now, taking the expectation of the total weight for targets distributed on a n-ball centered around one robot and for robots on another ball centered on each target, each ball with radius R, produces an upper bound on the expectation for robots and targets within a radius of $R/2$ of the given robot. Given the bound on interaction between robots in terms of interactions between robots and targets (6.22), and designating the zero-centered ball with radius R as B this expectation has the form:

$$\mathbb{E}\left[\sum_{j\in\mathcal{R}\backslash\{i\}} \mathcal{W}(i,j)\right] = \int_B \int_B \alpha\beta \min(\phi(||\mathbf{x}||_2), \phi(||\mathbf{y}||_2))\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{y}\,. \tag{A.10}$$

By integrating over the surface of each ball (each an $(n-1)$-sphere with surface area S_{n-1})

$$= \alpha\beta \int_0^R \int_0^R S_{n-1}{}^2 r_1{}^{n-1} r_2{}^{n-1} \min(\phi(r_1), \phi(r_2))\, \mathrm{d}r_1\, \mathrm{d}r_2\,. \tag{A.11}$$

Given that ϕ is non-increasing, separating the minimum produces:

$$
\begin{aligned}
= \alpha\beta S_{n-1}{}^2 \Bigg(&\int_0^R \int_0^{r_2} r_1{}^{n-1} r_2{}^{n-1} \phi(r_2)\, \mathrm{d}r_1\, \mathrm{d}r_2 \\
&+ \int_0^R \int_{r_2}^R r_1{}^{n-1} r_2{}^{n-1} \phi(r_1)\, \mathrm{d}r_1\, \mathrm{d}r_2 \Bigg),
\end{aligned}
\tag{A.12}
$$

and by swapping the bounds of the second integral, combining, and evaluating the inner integral, we get:

$$= \alpha\beta S_{n-1}^2 \left(\int_0^R \int_0^{r_2} r_1^{n-1} r_2^{n-1} \phi(r_2)\, \mathrm{d}r_1\, \mathrm{d}r_2 \right.$$
$$\left. + \int_0^R \int_0^{r_1} r_1^{n-1} r_2^{n-1} \phi(r_1)\, \mathrm{d}r_2\, \mathrm{d}r_1 \right) \qquad (A.13)$$

$$= 2\alpha\beta S_{n-1}^2 \int_0^R \int_0^{r_1} r_1^{n-1} r_2^{n-1} \phi(r_1)\, \mathrm{d}r_2\, \mathrm{d}r_1 \qquad (A.14)$$

$$= \frac{2\alpha\beta S_{n-1}^2}{n} \int_0^R r_1^{2n-1} \phi(r_1)\, \mathrm{d}r_1 . \qquad (A.15)$$

The above integral (A.15) converges in the limit if $\phi \in O(1/x^{2n+\epsilon})$, or most relevantly, on a plane this condition comes to $\phi \in O(1/x^{4+\epsilon})$.[3] Given that sensor models purely with sensor noise proportional to distance do not fit this constraint, designers may wish to consider the effects of constraints on the total sensor range or the number of observations one robot can obtain at once.

A.2.1 Scaling and sensor models

The sensitivity to how quickly interactions between robots and targets fall off motivates attention to sensor design and modeling lest planners perform poorly for large numbers or require additional computation time. For example, additive Gaussian noise with the standard deviation proportional to distance (as is common in range sensing models [35]) is insufficient which is evident from the channel capacity of a Gaussian channel [58, Chap. 9].

At the same time, robots in realizable systems cannot obtain and process observations of unbounded numbers of targets, and features such as a maximum sensor range can model these limits. Still, the observation of whether a target is within range provides some information. Here, a narrow tails assumption on the probability distributions for the targets—such as that the *tails approach zero exponentially*—ensures that ϕ decreases sufficiently quickly. As a result of this sensitivity to the tails, the scaling behavior (A.15) may be difficult to characterize a-priori, even when known to be bounded.

A.3 An argument in favor of coverage over entropy reduction for evaluating exploration performance

Chapter 7 applies coverage-based performance measures while, previously in Chapter 4, we used entropy reduction to evaluate exploration performance. However, the systems for exploration that we study in this thesis are subject to little or no noise and are tuned so that cell occupancy values converge quickly. Anecdotally, we have observed significant

---

[3]This requirement on interactions between *robots and targets* is stricter than the equivalent one *between robots* (see Theorem 8) as interactions between robots must decrease as $O(1/x^{n+\epsilon})$.
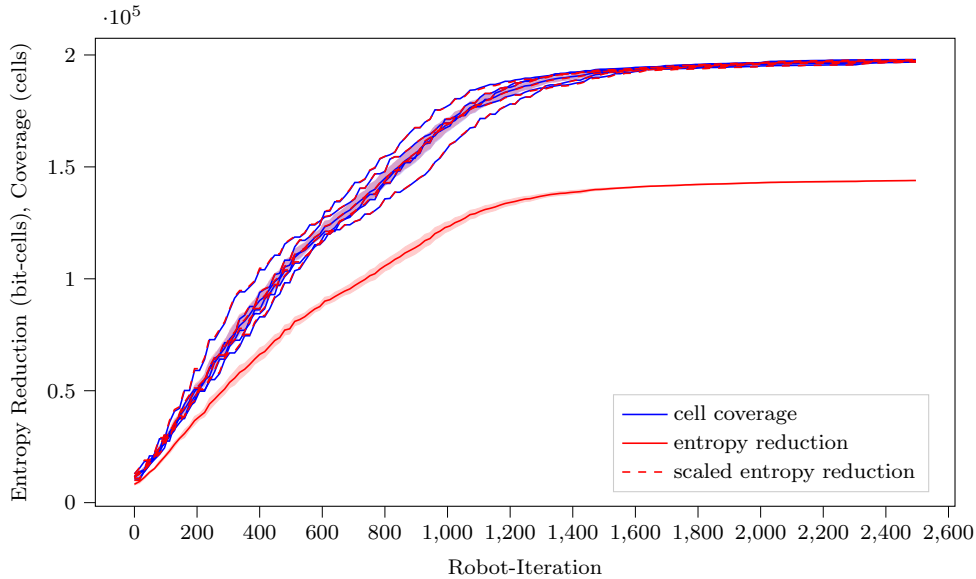
Figure A.2: Compare entropy reduction and environment coverage for five exploration trials with 16 robots in the *boxes* environment (jagged lines are individual trials). Notice that these coincide exactly when scaled to match.

changes in maximum entropy reduction for exploration trials in the same environment as systems and parameters have changed over time. Moreover, entropy results for our systems closely match environment coverage when scaled by a constant factor (Fig. A.2) which likely reflects design decisions such as limits on occupancy certainty (which are common in practice [92, (4)]) rather than any feature of performance. Instead, a coverage-based metric provides a consistent measure of the amount of volume that robots have observed (given cell volume) and is less affected by design parameters such as limits on occupancy uncertainty.

A.4 Evaluating the performance of an approximate mutual information objective

Initially, we chose the optimistic coverage objective in Chapter 7 primarily because exact evaluation is tractable and because optimistic coverage is 3-increasing.[4] As such, we wish to clarify how this decision affects performance, particularly in comparison to the CSQMI objective applied in Chapter 4. Additionally, we are not aware of any other works that compare ray-based mutual information [37, 103] to coverage-like approaches which are also common [20, 62, 168]. On the other hand, Zhang and Vorobeychik [200] found that Shannon mutual information objectives behaves similarly as CSQMI.

Figure A.3 presents results for exploration in the Boxes and Empty environments where optimistic coverage provides a 16% and 11% improvement in average completion time, re-

[4]We made this decision before identifying the connection between mutual information and expected coverage.
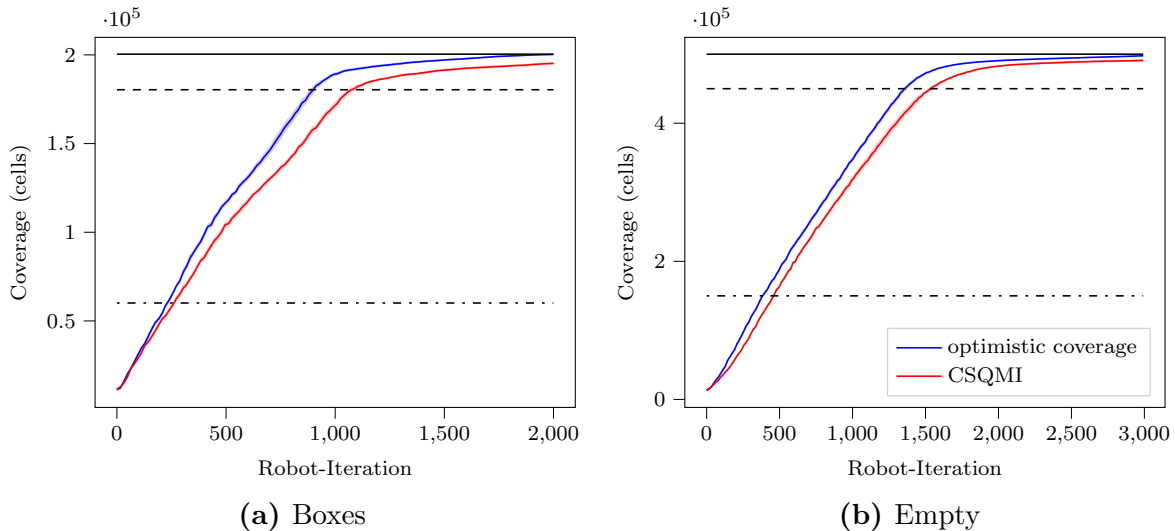
144

Figure A.3: The above plots compare exploration with optimistic coverage and CSQMI objectives in terms of environment coverage in the Boxes and Empty environments (for ten trials with sequential planning and 16 robots). Black lines demarcate (from top to bottom) the maximum environment coverage, the completion threshold, and the early progress threshold.

spectively, compared to exploration with a CSQMI objective with an occupancy prior of 0.125.[5] Note that some improvement in performance is expected for the Empty environment because the optimistic coverage objective provides oracle values.

These results are somewhat limited. First, we do not vary the parameters of the CSQMI objective such as the prior and parameters for the independence test. In particular, the results suggest that one way to improve performance may be to select a lower occupancy prior (compared to 0.125). Additionally, we did not put the exploration system with the CSQMI objective through the same parameter tuning process as we did for optimistic coverage. Still, the average completion times for CSQMI in the Boxes and Empty environments (1067 and 1526 robot-iterations, respectively) exceed the average for any configuration of the optimistic coverage objective that we tested.

That said, these results indicate several ways by which exploration performance could be improved for systems based on mutual information objectives:

To begin, selecting priors with low occupancy probabilities may improve performance (in agreement with Henderson et al. [87]) given that optimistic coverage is equivalent to mutual information with a prior approaching zero. Toward this end, normalizing the mutual information (such as by dividing by the entropy of unobserved cells) would mitigate the issue of the mutual information trending toward zero for small priors. The motivation for such normalization is self-evident from the form of (7.11), and we note that the optimistic coverage objective effectively realizes this normalization in the limit.

---

[5]Chapter 4 uses the same value for the prior.

Additionally, approximation of joint mutual information[6] via the method of Charrow et al. [37]) may significantly affect accuracy for small occupancy priors.[7] Notably, recent work by Henderson et al. [87] provides accurate values for individual camera views by treating the view as a continuous integral as well as new bounds for multiple views, and this approach could significantly improve approximations for multiple views and robots. For either case, optimistic coverage, having the advantage of being exact for a certain regime, could provide a useful point of comparison for evaluating the impact of approximation.

A.4.1   Upper and lower bounds on mutual information

As stated, approximations of mutual information for multiple rays and camera views frequently apply upper and lower bounds on joint mutual information [37, 88]. The interpretation of noiseless mutual information as expected coverage likewise leads to useful bounds on mutual information. Specifically, the expectation for the mutual information in (7.11) commutes and can be computed cell-wise. The expectations for individual cells can then be bounded via upper and lower bounds on probabilities that an observation along a given ray will infer the occupancy value for that cell.[8] The maximum probability that any of several rays will observe a cell produces a lower bound on the mutual information while summing probabilities of observation (and capping at one) produces an upper bound. Notably, both of these bounds become tight as the prior probability of occupancy approaches zero because the probability of observing all cells in range approaches one.

A.5   Comparison of suboptimality for RSP and DSGA

The proposed RSP planners overcome a number of limitations of DSGA (Chapter 4): DSGA does not provide offline suboptimality guarantees, distributed implementations would involve broadcasts and reductions over all robots during each round, and the assignments themselves are sequential. We provide the following comparison as a point of reference as DSGA may be impractical in practice.

Figure A.4 compares lower bounds on suboptimality (Sec. 7.6.1) for DSGA and RSP given different numbers of planning rounds (n_d). On average, DSGA outperforms RSP planning in these results. Even though RSP provides stricter performance guarantees, this result is not surprising. While RSP partitions robots randomly, DSGA effectively partitions robots in the midst of the planning process. This could enable DSGA to better take advantage of problem structure or random improvements in planner results across rounds (as DSGA selects a subset of results from *all* robots during each round).

---

[6]Arguably, accurate evaluation of joint mutual information is much more important to this work than others on exploration since we study the collective contributions of teams of robots.

[7]One case where the approximation of the joint by Charrow et al. [37] would perform poorly is when many rays traverse a single, previously unobserved cell but collectively observe many unobserved cells. However, the contributions of only one or few rays would be counted.

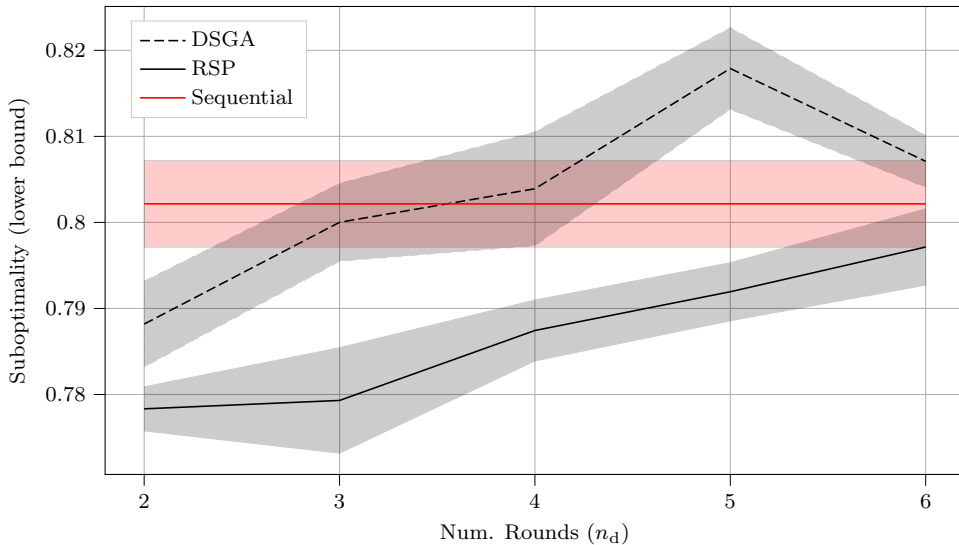[8]Such probabilities are trivial to compute [37, 103].

Figure A.4: Comparison of lower bounds on suboptimality for planning for exploration with RSP and DSGA for ten simulation trials in the Boxes environment. The red line is based suboptimality results for sequential planning in the same configuration from Table 7.3.

A.6 Description of auction methods for distributed submodular maximization

Chapter 8 compares RSP planning to auction methods much like those which are common for task assignment [46]. Such auction algorithms converge to greedy solutions (Alg. 1) and can even be extended to complex constraints [194]. However, the auction algorithms that Choi et al. [46] describe (e.g. CBBA) are specific to the assignment problems they study and do not immediately apply for general submodular objectives. Specifically, they take advantage of how the value of an assignment to one robot is independent of assignments to others to avoid recomputing marginal gains (bids) when updating assignments.

The following algorithms (Algs. 6 and 7) are similar and are suitable for the kinds of assignment problems we study in this thesis (e.g. (2.9)) The two differ based on assumptions on how agents evaluate objective values. The first requires full information—the ability to evaluate marginal gains for any agents' actions—in order for agents to reorder existing assignments. This likely produces behavior that is comparable to CBBA which can often avoid re-evaluating marginal gains. For simplicity we assume that both run synchronously on an undirected communication graph where each agent i has neighbors $\mathcal{N}_i$. Both algorithms eventually converge to the maximum value[9] across all agents at each step of the assignment process and so eventually obtain solutions equivalent to the general greedy algorithm (Alg. 1). Although we do not seek to prove this point, the arguments would be similar to those by Choi et al. [46].

Algorithm 6 requires consistent, global access to the objective g. This algorithm con-

---

[9]Like Choi et al. [46] we assume that there is a deterministic mechanism to break symmetry in comparisons (e.g. via agent and action indices).

sists of exchanging assignments with neighbors and simply executing that general greedy algorithm (Alg. 1) on a reduced ground set to update the agent's assignments, much like the approach of Luo et al. [122]. The reduced ground set consists of the agent's own block of control actions along with its and its neighbors current assignments. The output then preserves maxima at each assignment position and includes some best-guess at the assignment to that agent (i). This greedy process can enable agents to collectively identify multiple elements of the final solution during a single synchronous communication round and sometimes converges in fewer synchronous communication rounds than the number of agents. However, this approach *strictly* requires global access to the objective g because the agents evaluate the value of others' assignments during the greedy decision process and to ensure that the outputs are consistent and deterministic. As such, this algorithm does not represent a practical solution to the problems we study as asymmetry in information access is common in distributed settings [46].[10]

Algorithm 6 Auction algorithm with global information. One synchronous iteration from the perspective of agent i.

1: $X_i \leftarrow$ Current set of assignments for agent i
2: $\mathcal{B}_i \leftarrow$ Set of actions available to i
3: $\mathcal{N}_i \leftarrow$ Set of communication neighbors

4: SEND: X_i to $\mathcal{N}_i$
5: RECEIVE: X_j from j for $j \in \mathcal{N}_i$

6: $G \leftarrow \bigcup_{j \in \mathcal{N}_i \cup i} X_j \cup \mathcal{B}_i$
7: $X_i \leftarrow$ GREEDY(G)   (Executing Alg. 1 for objective g and partition matroid $\mathscr{I}$)

Algorithm 7, which performs auctions with local information, is somewhat more complex because agents are not allowed to evaluate marginal gains for others' assignments so that these values have to be communicated along with the decisions and so that updates require more care. The main body (lines 5–8) of this algorithm is similar to the last in that agents exchange assignments with neighbors before updating the local solution. Except, this time, values and assignments are exchanged in lists in assignment order. The update itself is based on the first position where the assignments differ and consists of three cases: (line 12) the current assignments dominate (and already include an assignment to i); (line 14) the other's assignments dominate and include some assignment to i (in block $\mathcal{B}_i$ of the partition matroid), obtained via the following procedure; or finally (line 16) agent i must update the other's assignments with an assignment to itself. Seeking to emulate Alg. 1, agent i checks whether its best marginal gain beats the value of the previous winning assignment $V_{o,m}$ at each point m in the decision process. Once identifying a winning assignment position (or implicitly beating a null assignment) the agent concatenates its assignment and value (using "·" to represent list concatenation) onto the sub-lists of prior

---

[10]For example, robots in Chapter 6 use different local approximations for target tracking, and for the distributed exploration implementation in Chapter 8, even though robots construct maps with sensor data from all robots, the maps are not synchronized and generally differ.

assignments and values (line 21) and returns this result. The agent thereby discards assignments at later positions in the sequence as inserting the assignment for i invalidates any following marginal gains. This behavior also implies that, unlike Alg. 6, this auction algorithm requires at least one synchronous round per each element of the final solution (i.e. one per robot) as the incremental construction of the solution in line 21 ensures that the agents also collectively produce the full solution sequentially.

Both of these algorithms can be implemented more efficiently from a computational perspective. But, given that the discussion in Chapter 8 focuses on communication costs, efficient computation will not be necessary for this text. Still, even efficient implementations of these auction algorithms will be more computation intensive than RSP (where agents together complete a single pass over each block $\mathscr{B}_i$ of the ground set). The auction algorithms both execute a complete pass over the ground set in the first synchronous round alone and may require many times more computation to converge. Likewise, auction algorithms incur greater communication costs because all agents exchange assignments during each round (rather than a subset) and because agents exchange sets of assignments rather than individual assignments as in RSP.

Algorithm 7 Auction algorithm with local information. One synchronous iteration from the perspective of agent i.

1: $X_i \leftarrow List$ of assignments for agent i (in assignment order)
2: $V_i \leftarrow$ List of marginal gains (values) for assignments X_i
3: $\mathscr{B}_i \leftarrow$ Set of actions available to i
4: $\mathcal{N}_i \leftarrow$ Set of communication neighbors

5: SEND: X_i, V_i to $\mathcal{N}_i$
6: RECEIVE: X_j, V_j from j for $j \in \mathcal{N}_i$

7: **for** $j \in \mathcal{N}_i$ **do**
8:     X_i, $V_i \leftarrow$ UPDATEASSIGNMENTS(X_i, V_i, X_j, V_j)

9: ▷ Update local assignment in light of one from an other agent
10: **procedure** UPDATEASSIGNMENTS$(X_\ell, V_\ell, X_o, V_o)$
11:     $n_{\text{diff}} \leftarrow \min_n V_{\ell,n} \neq V_{o,n}$
12:     **if** $V_{\ell,n} \geq V_{o,n}$ **then**   (Agent i's assignments dominate. Return.)
13:         **return** X_ℓ, V_ℓ
14:     **else if** $\mathscr{B}_i \cap X_o \neq \emptyset$ **then**   (Other's assignments reflect i's choices)
15:         **return** X_o, V_o
16:     **else**
17:         **for** $m \in \{n, \ldots, |X_o|\}$ **do**   (Search for a viable assignment position)
18:             $x \leftarrow \arg\max_{x' \in \mathscr{B}_i} g(x'|X_{o,1:m-1})$
19:             $v \leftarrow g(x|X_{o,1:m-1})$
20:             **if** $v > V_{o,m}$ **then**   (i's bid wins at position m)
21:                 **return** $X_{o,1:m-1} \cdot x$, $V_{o,1:m-1} \cdot v$ (values V_o valid up to $m-1$)

Appendix B

Assorted Proofs

B.1 Proof for representations of derivatives of set functions

Equation (3.16) defines the nth derivative of a set function as

$$g(Y_1; \ldots; Y_n | X) = g(Y_1; \ldots; Y_{n-1} | X, Y_n) - g(Y_1; \ldots; Y_{n-1} | X). \tag{B.1}$$

This is equivalent to the definition by Foldes and Hammer [70],

$$g(Y_1; \ldots; Y_n | X) = \sum_{Y \subseteq \{Y_1, \ldots, Y_n\}} (-1)^{n-|Y|} g(X \cup \hat{Y}), \quad \hat{Y} = \bigcup_{Y' \in Y} Y'. \tag{B.2}$$

with the exceptions that we define derivatives with respect to sets for convenience (where the extension follows from closure properties Foldes and Hammer [70]) and omit set differences by requiring disjoint sets.

Proof. The proof follows by expanding the right hand side of (B.1) and combining the resulting expressions. Define the union of a collection of sets as $\phi(X) = \bigcup_{X' \in X} X'$. Then

$$
\begin{aligned}
&g(Y_1; \ldots; Y_{n-1} | X, Y_n) - g(Y_1; \ldots; Y_{n-1} | X) \\
&= \sum_{Y \subseteq \{Y_1, \ldots, Y_{n-1}\}} (-1)^{n-1-|Y|} g(X \cup Y_n \cup \phi(Y)) - \sum_{Y \subseteq \{Y_1, \ldots, Y_{n-1}\}} (-1)^{n-1-|Y|} g(X \cup \phi(Y)) \\
&= \sum_{Y \subseteq \{Y_1, \ldots, Y_{n-1}\}} (-1)^{n-(|Y|+1)} g(X \cup Y_n \cup \phi(Y)) + \sum_{Y \subseteq \{Y_1, \ldots, Y_{n-1}\}} (-1)^{n-|Y|} g(X \cup \phi(Y)) \\
&= \sum_{Y \subseteq \{Y_1, \ldots, Y_n\}} (-1)^{n-|Y|} g(X \cup \phi(Y)) = g(Y_1; \ldots; Y_n | X).
\end{aligned}
$$

■

B.2 Proof of Lemma 1, chain rule for derivatives of set functions

Proof. The proof follows by expanding the derivative (3.16), forming a telescoping sum, and rewriting the summands as individual derivatives:

$$
\begin{aligned}
g(Y_1; \ldots; Y_n | X) &= g(Y_1; \ldots; Y_{n-1} | Y_n, X) - g(Y_1; \ldots; Y_{n-1} | X) \\
&= \sum_{i=1}^{|Y_n|} \big(g(Y_1; \ldots; Y_{n-1} | Y_{n,1:i}, X) - g(Y_1; \ldots; Y_{n-1} | Y_{n,1:i-1}, X) \big) \\
&= \sum_{i=1}^{|Y_n|} g(Y_1; \ldots; Y_{n-1}; y_{n,i} | Y_{n,1:i-1}, X).
\end{aligned}
\tag{B.3}
$$

∎

B.3 Proof of Theorem 3, post-hoc suboptimality of DSGA

Proof. The proof of the suboptimality bound relating DSGA to SGA incorporates suboptimality of the single-robot planner and is similar to [169] or [7]. We obtain the following by monotonicity and by rearranging the resulting telescoping sum

$$
\begin{aligned}
\mathbb{I}(M; Y^\star) &\leq \mathbb{I}(M; Y^\star) + \sum_{i=1}^{n_r} \mathbb{I}(M; Y_i^d | Y_{1:i-1}^d, Y_{i+1:n_r}^\star) \\
&= \mathbb{I}(M; Y^d) + \sum_{i=1}^{n_r} \mathbb{I}(M; Y_i^\star | Y_{1:i-1}^d, Y_{i+1:n_r}^\star).
\end{aligned}
\tag{B.4}
$$

By submodularity

$$
\mathbb{I}(M; Y_i^\star | Y_{1:i-1}^d, Y_{i+1:n_r}^\star) \leq \mathbb{I}(M; Y_i^\star | Y_{1:i-1}^d).
$$

Without loss of generality, assume that agent indices correspond to the selection order and rewrite in terms of the planning rounds such that $\mathbb{I}(M; Y_i^\star | Y_{1:i-1}^d) = \mathbb{I}(M; Y_{D_{j,k}}^\star | Y_{D_{j,1:k-1} \cup F_{j-1}}^d)$ and note that although $Y^\star$ is formally a set, the mapping from elements to robots can be obtained by the intersections $Y^\star \cap \mathcal{Y}_i$ given that the sets $\mathcal{Y}_i$ are disjoint. Then, by submodularity,

$$
\mathbb{I}(M; Y_i^\star | Y_{1:i-1}^d) \leq \mathbb{I}(M; Y_{D_{j,k}}^\star | Y_{F_{j-1}}^d).
$$

By (4.9) and the greedy maximization step in Alg. 3

$$
\mathbb{I}(M; Y_{D_{i,j}}^\star | Y_{F_{i-1}}^d) \leq \eta \mathbb{I}(M; Y_{D_{i,j}}^d | Y_{F_{i-1}}^d).
\tag{B.5}
$$

Substitute (B.5) and preceding inequalities into (B.4) to obtain

$$\mathbb{I}(M; Y^{\star}) \leq \mathbb{I}(M; Y^{\mathrm{d}}) + \eta \sum_{i=1}^{n_{\mathrm{d}}} \sum_{j=1}^{|D_i|} \mathbb{I}(M; Y^{\mathrm{d}}_{D_{i,j}} | Y^{\mathrm{d}}_{F_{i-1}}).$$ (B.6)

The distributed objective can be rewritten as a sum so that $\mathbb{I}(M; Y^{\mathrm{d}}) = \sum_{i=1}^{n_{\mathrm{d}}} \sum_{j=1}^{|D_i|} \mathbb{I}(M; Y^{\mathrm{d}}_{D_{i,j}} | Y^{\mathrm{d}}_{D_{i,1:j-1} \cup F_{i-1}})$ and substituted into (B.6) to obtain

$$\mathbb{I}(M; Y^{\star}) \leq (1 + \eta) \mathbb{I}(M; Y^{\mathrm{d}}) + \eta \sum_{i=1}^{n_{\mathrm{d}}} \sum_{j=1}^{|D_i|} \left(\mathbb{I}(M; Y^{\mathrm{d}}_{D_{i,j}} | Y^{\mathrm{d}}_{F_{i-1}}) - \mathbb{I}(M; Y^{\mathrm{d}}_{D_{i,j}} | Y^{\mathrm{d}}_{D_{i,1:j-1} \cup F_{i-1}}) \right)$$ (B.7)

which expresses the suboptimality in terms of decreases in the conditional mutual information from when the plans were first obtained from the planner (Alg. 3, line 7) to when they were assigned, (Alg. 3, line 9) potentially after other assignments in the same round. By rewriting mutual information in terms of entropies we can rearrange to obtain the following

$$\mathbb{I}(M; Y_1) - \mathbb{I}(M; Y_1 | Y_2) = \mathbb{I}(Y_1; Y_2) - \mathbb{I}(Y_1; Y_2 | M).$$

If Y_1 and Y_2 are conditionally independent given M, then the mutual information, $\mathbb{I}(Y_1; Y_2 | M) = 0$ and so $\mathbb{I}(M; Y_1) - \mathbb{I}(M; Y_1 | Y_2) = \mathbb{I}(Y_1; Y_2)$. By substitution into (B.7) we can obtain the slightly more concise and final expression for the suboptimality in terms of the mutual information between observations

$$\mathbb{I}(M; Y^{\star}) \leq (1 + \eta) \mathbb{I}(M; Y^{\mathrm{d}}) + \eta \sum_{i=1}^{n_{\mathrm{d}}} \sum_{j=1}^{|D_i|} \mathbb{I}(Y^{\mathrm{d}}_{D_{i,j}}; Y^{\mathrm{d}}_{D_{i,1:j-1}} | Y^{\mathrm{d}}_{F_{i-1}}).$$ (B.8)

■

B.4 Proof of Theorem 4, worst case suboptimality of DSGA

Proof. Equation (4.13) follows from Theorem 1 by Grimsman et al. [82] which proves a $k+1$ bound where k is the clique cover number of a directed acyclic graph associated with the planner structure. In this directed graph, each robot represents a vertex, and the graph has a directed edge (a, b) between robots $a, b \in \mathcal{R}$ if b maximizes its objective (4.7) conditional on the sequence of observations selected by a. Here, a clique, which is a complete subgraph, is analogous to a set of robots that plan sequentially given the choices by all prior robots in the clique. For Alg. 3, any set of robots $A \subseteq \mathcal{R}$ with at most one robot from each planning round ($|A \cap D_i| \leq 1$ for $i \in \{1, \ldots, n_{\mathrm{d}}\}$) forms a clique in the associated directed graph. A clique cover of size $\lceil n_{\mathrm{r}}/n_{\mathrm{d}} \rceil = \max_{i \in \{1, \ldots, n_{\mathrm{d}}\}} |D_i|$ can be obtained by selecting cliques with a single robot (as available) from each planning round ($D_1, \ldots, D_{n_{\mathrm{d}}}$) without replacement. Then, (4.13) follows by substitution of (4.9) to obtain a factor of η. ■

B.5 Proof of Lemma 11, suboptimality of general assignments

Proof. This result follows a similar approach as the other proofs related to sequential submodular maximization that arise throughout this thesis with slight deviation to assist in book-keeping:

$$g(X^\star) \leq g(X^d, X^\star) \tag{B.9}$$

$$= g(X^d) + \sum_{i=1}^{n_r} g(x_i^\star | X_{1:i-1}^\star, X^d) \tag{B.10}$$

$$\leq g(X^d) + \sum_{i=1}^{n_r} g(x_i^\star | X_{1:i-1}^d) \tag{B.11}$$

$$\leq g(X^d) + \sum_{i=1}^{n_r} \left(g(x_i^d | X_{1:i-1}^d) + \gamma_i(g, X_{1:i-1}^d) \right) \tag{B.12}$$

$$= 2g(X^d) + \sum_{i=1}^{n_r} \gamma_i(g, X_{1:i-1}^d). \tag{B.13}$$

Above, (B.9) follows from monotonicity; (B.10) expands a telescoping series; (B.11) follows from submodularity; (B.12) upper bounds the incremental values optimal robot decisions with the incremental value of each actual decision and its suboptimality (6.8); and (B.13) collapses the telescoping series. ∎

B.6 Proof of Theorem 10, suboptimality of distributed planning for target tracking

Proof. Theorem 10 consists of two parts, (6.12) and (6.13). We prove each in turn. Since the costs in both equations involve sums over robots, both proofs analyze costs with respect to some robot $i \in \mathcal{R}$.

Proof of Theorem 10, part 1 (6.12)

According to the standard greedy algorithm (Alg. 2), robot i would plan conditional on decisions by robots $\{1, \ldots, i-1\}$. However, in Alg. 4 that robot instead plans conditional on a subset of these robots $\mathcal{N}_i \subseteq \{1, \ldots, i-1\}$ and ignores $\hat{\mathcal{N}}_i = \{1, \ldots, i-1\} \setminus \mathcal{N}_i$. We can then write the cost of a suboptimal decision from (6.14) in terms of γ_i^{dist} as

$$\gamma_i(g, X_{1:i-1}^d) \leq \gamma_i(g, X_{\mathcal{N}_i}^d) + g(x_i^d | X_{\mathcal{N}_i}^d) - g(x_i^d | X_{1:i-1}^d)$$
$$= \gamma_i(g, X_{\mathcal{N}_i}^d) + \gamma_i^{\text{dist}}, \tag{B.14}$$

where the first step follows by substituting the cost model (6.8) and given that $\max_{x \in \mathcal{B}_i} g(x | X_{1:i-1}^d) \leq \max_{x \in \mathcal{B}_i} g(x | X_{\mathcal{N}_i}^d)$ due to submodularity, and the second follows

from the definition of the cost of distributed planning (6.9). To incorporate the cost of suboptimal planning γ_i^{plan}, observe that

$$
\begin{aligned}
\gamma_i(g, X_{\mathcal{N}_i}^{\text{d}}) &= \gamma_i(g, X_{\mathcal{N}_i}^{\text{d}}) + \gamma_i^{\text{plan}} - \gamma_i^{\text{plan}} \\
&= \gamma_i^{\text{plan}} + \gamma_i(g, X_{\mathcal{N}_i}^{\text{d}}) - \gamma_i(\widetilde{g}_i, X_{\mathcal{N}_i}^{\text{d}})
\end{aligned}
\tag{B.15}
$$

which follows from the definition of the planning cost (6.11). The cost of approximation of the objective γ_i^{obj} upper bounds the difference of the last two terms in (B.15):

$$
\begin{aligned}
\gamma_i(g, X_{\mathcal{N}_i}^{\text{d}}) - \gamma_i(\widetilde{g}_i, X_{\mathcal{N}_i}^{\text{d}}) &= \widetilde{g}_i(x_i^{\text{d}}|X_{\mathcal{N}_i}^{\text{d}}) - g(x_i^{\text{d}}|X_{\mathcal{N}_i}^{\text{d}}) \\
&\quad + \max_{x \in \mathscr{B}_i} g(x|X_{\mathcal{N}_i}^{\text{d}}) - \max_{x \in \mathscr{B}_i} \widetilde{g}_i(x|X_{\mathcal{N}_i}^{\text{d}}) \\
&\leq \widetilde{g}_i(x_i^{\text{d}}|X_{\mathcal{N}_i}^{\text{d}}) - g(x_i^{\text{d}}|X_{\mathcal{N}_i}^{\text{d}}) \\
&\quad + g(\hat{x}|X_{\mathcal{N}_i}^{\text{d}}) - \widetilde{g}_i(\hat{x}|X_{\mathcal{N}_i}^{\text{d}}), \quad \text{for } \hat{x} \in \arg\max g(\hat{x}|X_{\mathcal{N}_i}^{\text{d}}) \\
&\leq \gamma_i^{\text{obj}}
\end{aligned}
\tag{B.16}
$$

by expanding and rearranging the costs (6.8) on the left-hand-side, using an upper bound to match the arguments of the last two terms, and by using the definition of the objective cost (6.10) to bound the two differences.

Then, the expression for the costs in (6.12)

$$
\gamma_i(g, X_{1:i-1}^{\text{d}}) \leq \gamma_i^{\text{obj}} + \gamma_i^{\text{plan}} + \gamma_i^{\text{dist}}
\tag{B.17}
$$

follows by substituting the prior three equations into each other: (B.15) into (B.14) and (B.16) into the result. Finally, substituting this inequality (B.17) into (6.14) from Lemma 11 on the suboptimality of general assignments yields the expression for (6.12) which completes the first part of this proof.

Proof of Theorem 10, part 2 (6.13)

The second part of Theorem 10 (6.13) follows by referring to definition of γ_i^{dist} in (6.9), applying the chain rule (3.17), and substituting the bound on second derivatives (6.20) and the definitions of the weights (6.21) and (6.22) in turn:

$$
\gamma_i^{\text{dist}} = -g(x_i^{\text{d}}; X_{\hat{\mathcal{N}}_i}^{\text{d}}|X_{\mathcal{N}_i}^{\text{d}})
\tag{B.18}
$$

$$
= -\sum_{j \in \hat{\mathcal{N}}_i} g\left(x_i^{\text{d}}; x_j^{\text{d}}|X_{\mathcal{N}_i}^{\text{d}}, X_{\hat{\mathcal{N}}_i \cap \{1:j-1\}}^{\text{d}}\right)
\tag{B.19}
$$

$$
\leq \sum_{j \in \hat{\mathcal{N}}_i} \mathcal{W}(i, j) \leq \sum_{j \in \hat{\mathcal{N}}_i} \widehat{\mathcal{W}}(i, j).
\tag{B.20}
$$

Then, (6.13) follows by summing over $\mathcal{R}$. This completes this second and last part of the proof of Theorem 10. ∎

B.7 Proof of Theorem 12, noiseless mutual information with independent cells is 3-increasing

The following proof takes advantage of cell independence liberally to write mutual information in terms of the expected entropy of the cells that the robots will observe.

Proof. We can write the mutual information (3.6) between the environment E and future observations $\mathbf{Y}(X)$ in terms of entropies:

$$\mathbb{I}(E; \mathbf{Y}(X)) = \mathbb{H}(E) - \mathbb{H}(E|\mathbf{Y}(X)). \tag{B.21}$$

The conditional entropy can then be rewritten in terms of the expected entropy (3.4) given the direct observations of cell occupancy (7.3) associated with a hypothetical instantiation of the environment E' while abbreviating observed cells as $C' = C_{\text{cov}}(X, E')$:

$$= \mathbb{H}(E) - \mathbb{E}_{E' \sim \mathcal{E}_{\text{guess}}} \left[\mathbb{H}(E|E_{C'} = E'_{C'}) \right], \tag{B.22}$$

and that conditional entropy is simply the entropy of the cells that have not yet been observed $D' = C \setminus C'$ due to independence:

$$= \mathbb{H}(E) - \mathbb{E}_{E' \sim \mathcal{E}_{\text{guess}}} \left[\mathbb{H}(E_{D'}) \right]. \tag{B.23}$$

Then, bringing the entropy of E into the expectation does not change its value, and separating the independent observed and unobserved cells simplifies the expression:

$$= \mathbb{E}_{E' \sim \mathcal{E}_{\text{guess}}} \left[\mathbb{H}(E_{C'}) + \mathbb{H}(E_{D'}) - \mathbb{H}(E_{D'}) \right]. \tag{B.24}$$

$$= \mathbb{E}_{E' \sim \mathcal{E}_{\text{guess}}} \left[\mathbb{H}(E_{C'}) \right]. \tag{B.25}$$

Finally, the joint entropy of the cells the robot will observe is the sum of their individual entropies

$$\mathbb{I}(E; \mathbf{Y}(X)) = \mathbb{E}_{E' \sim \mathcal{E}_{\text{guess}}} \left[\sum_{i \in C_{\text{cov}}(X, E')} \mathbb{H}(c_i) \right]. \tag{B.26}$$

This expresses a weighted expected coverage objective (7.8) where the weight $w_{\text{cell}}(i)$ of each cell $i \in C$ is equal to its entropy $\mathbb{H}(c_i)$. Observing that weighted expected coverage is 3-increasing (Sec. 7.3.1a) completes the proof. ∎

Bibliography

[1] Ian Abraham, Anastasia Mavrommati, and Todd D Murphey. Data-driven measurement models for active localization in sparse environments. In *Proc. of Robot.: Sci. and Syst.*, Pittsburgh, PA, 2018. 3.3.1

[2] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016. 2.1.1d

[3] Gunnar Andersson. An approximation algorithm for Max p-Section. In *Annual Symposium on Theoretical Aspects of Computer Science*, Trier, Germany, 1999. 5.5.2

[4] Aristotle Arapostathis, Vivek S. Borkar, Emmanuel Fernández-Gaucherand, Mrinal K. Ghosh, and Steven I. Marcus. Discrete-time controlled markov processes with average cost criterion: a survey. *SIAM Journal on Control and Optimization*, 31(2): 282–344, 1993. 2.2.1, 2.2.2

[5] Mauricio Araya, Olivier Buffet, Vincent Thomas, and Françcois Charpillet. A pomdp extension with belief-dependent rewards. In *Advances in neural information processing systems*, pages 64–72, 2010. 3.3.3

[6] Arash Asadpour and Hamid Nazerzadeh. Maximizing stochastic monotone submodular functions. *Management Science*, 62(8):2374–2391, 2015. 1, 3.3.3

[7] Nikolay A. Atanasov, Jerome Le Ny, Kostas Daniilidis, and George J. Pappas. Decentralized active information acquisition: Theory and application to multi-robot SLAM. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Seattle, WA, May 2015. 1.2, 3.1.2, 3.3.1, 3.4.1, 4.1, 6.1, 7.8.1, 8, B.3

[8] Elif Ayvali, Hadi Salman, and Howie Choset. Ergodic coverage in constrained environments using stochastic trajectory optimization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Vancouver, Canada, September 2017. 2.2.2

[9] Francis Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2-3), 2013. 5.6, 3

[10] Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988. 3.3.1

[11] Eric Balkanski and Yaron Singer. The adaptive complexity of maximizing a submodular function. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1138–1151, 2018. 3.6.3, 3.1, 9

[12] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 66–77, 2019. 3.6.3, 8, 3.1, 3.6.5

[13] Jacopo Banfi, Nicola Basilico, and Francesco Amigoni. Multirobot reconnection on graphs: Problem, complexity, and algorithms. *IEEE Trans. Robotics*, 34(5):1299–1314, 2018. 3.1.6

[14] Jacopo Banfi, Alberto Quattrini Li, Ioannis Rekleitis, Francesco Amigoni, and Nicola Basilico. Strategies for coordinated multirobot exploration with recurrent connectivity constraints. *IEEE Trans. Robotics*, 42(4):875–894, 2018. 3.1.6

[15] Rafael Barbosa, Alina Ene, Huy Nguyen, and Justin Ward. The power of randomization: Distributed submodular maximization on massive datasets. In *International Conference on Machine Learning*, pages 1236–1244, 2015. 3.6.4

[16] Rafael da Ponte Barbosa, Alina Ene, Huy L. Nguyen, and Justin Ward. A new framework for distributed submodular maximization. In *Proc. of the IEEE Annu. Symp. Found. Comput. Sci.*, New Brunswick, NJ, October 2016. 3.6.4

[17] Jose A. J. Berni, Pablo J. Zarco-Tejada, Lola Suárez, and Elias Fereres. Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle. *IEEE Trans. Robotics*, 47(3):722–738, 2009. 1

[18] Graeme Best, Oliver M. Cliff, Timothy Patten, Ramgopal R. Mettu, and Robert Fitch. Dec-MCTS: Decentralized planning for multi-robot active perception. *Intl. Journal of Robotics Research*, 38(2-3):316–337, 2019. 3.4, 4.3, 8

[19] Andrew An Bian, Baharan Mirzasoleiman, Joachim Buhmann, and Andreas Krause. Guaranteed non-convex optimization: Submodular maximization over continuous domains. In *Proc. of the Intl. Conf. on Artif. Intell. and Stat.*, Ft. Lauderdale, FL, 2017. 9.1.1

[20] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon path planning for 3D exploration and surface inspection. *Auton. Robots*, 42(2):291–306, 2018. 2.1.2, 3.1.3, 3.4.1b, 7.3, A.4

[21] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S. Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Trans. Robotics*, 33(6):1273–1291, 2017. 3.3.1

[22] Rogerio Bonatti, Yanfu Zhang, Sanjiban Choudhury, Wenshan Wang, and Sebastian Scherer. Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming. In *Proc. of the Intl. Sym. on Exp. Robot.*, 2018. 1

[23] Terry Bossomaier, Lionel Barnett, Michael Harré, and Joseph T Lizier. *An introduction to transfer entropy*. Springer, 2016. 2.2.3

[24] Adam Breuer, Eric Balkanski, and Yaron Singer. The FAST algorithm for submod-

ular maximization. *arXiv preprint arXiv:1907.06173*, 2019. 3.6.3

[25] Rodney A Brooks. Intelligence without reason. 1991. 1

[26] Robin Brown, Federico Rossi, Kiril Solovey, Michael T. Wolf, and Marco Pavone. Exploiting locality and structure for distributed optimization in multi-agent systems. In *Proc. of the Euro. Control Conf.*, St. Petersburg, Russia, 2020. 6

[27] Cameron Browne, Edward Powley, Daniel Whitehouse, Simon Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Trans. on Comput. Intell. and AI in Games*, 4(1):1–43, 2012. 4.3, 6.4.4, 7.1, 8.4.2

[28] Francesco Bullo. *Lectures on Network Systems.* CreateSpace, 2018. 3.6.5

[29] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E. Schneider. Coordinated multi-robot exploration. *IEEE Trans. Robotics*, 21(3):376–386, 2005. 3.1.3, 3.1.5

[30] Jonathan Butzke and Maxim Likhachev. Planning for multi-robot exploration with multiple objective utility functions. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3254–3259. IEEE, 2011. 3.1.5, 7.3, 7.3.3a, 7.4

[31] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011. 3.5.3b, 3.6.3, 8, 3.6.5, 7.8.1, 9.1.1

[32] Luca Carlone and Sertac Karaman. Attention and anticipation in fast visual-inertial navigation. *IEEE Trans. Robotics*, 35(1):1–20, 2018. 3.5.3a

[33] Benjamin Charrow. *Information-Theoretic Active Perception for Multi-Robot Teams.* PhD thesis, University of Pennsylvania, 2015. 4.6.1b

[34] Benjamin Charrow, Vijay Kumar, and Nathan Michael. Approximate representations for multi-robot control policies that maximize mutual information. *Auton. Robots*, 37(4):383–400, 2014. 4.1, 6.1

[35] Benjamin Charrow, Nathan Michael, and Vijay Kumar. Cooperative multi-robot estimation and control for radio source localization. *Intl. Journal of Robotics Research*, 33(4):569–580, April 2014. 3.3.1, 6.1, A.2.1

[36] Benjamin Charrow, Gregory Kahn, Sachini Patil, Sikang Liu, Ken Goldberg, Pieter Abbeel, Nathan Michael, and Vijay Kumar. Information-theoretic planning with trajectory optimization for dense 3D mapping. In *Proc. of Robot.: Sci. and Syst.*, Rome, Italy, July 2015. 1, 2.1.2, 3.1.3, 6.4.4, 7.3, 7.4

[37] Benjamin Charrow, Sikang Liu, Nathan Michael, and Vijay Kumar. Information-theoretic mapping using Cauchy-Schwarz quadratic mutual information. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Seattle, WA, May 2015. 3.1, 3.1.3, 3.3.1, 4.2.2, 4.2.2, 4.4.5, 6, 7, 7.3, 7.3.2, 7.3.2a, 7.3.3b, A.4, A.4.1, 7, 8

[38] Benjamin Charrow, Nathan Michael, and Vijay Kumar. Active control strategies for discovering and localizing devices with range-only sensors. In *Algorithmic Found. Robot.*, pages 55–71, 2015. 3.4.1b, 8

159

[39] Guillaume Chaslot. *Monte-Carlo Tree Search*. PhD thesis, Universiteit Maastricht, 2010. 3.1.3, 4.3, 6.4.4, 6.7, 8.4.2

[40] Nesrine Mahdoui Chedly. *Communicating multi-UAV system for cooperative SLAM-based exploration*. PhD thesis, Université de Technologie de Compiègne, 2018. 3.1.5

[41] Chandra Chekuri and Pal Martin. A recursive greedy algorithm for walks in directed graphs. In *Proc. of the IEEE Annu. Symp. Found. Comput. Sci.*, pages 245–253, 2005. 2, 2.1.1c, 2.1.2a, 3.3.2, 4.3, 6.1, 6.4.4

[42] Chandra Chekuri and Kent Quanrud. Parallelizing greedy for submodular set function maximization in matroids and beyond. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 78–89, 2019. 3.6.3

[43] Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms (TALG)*, 8(3):1–27, 2012. 2.1.1c, 2.1.2a

[44] Wei Chen, Qiang Li, Xiaohan Shan, Xiaoming Sun, and Jialin Zhang. Higher order monotonicity and submodularity of influence in social networks: from local to global. *arXiv preprint arXiv:1803.00666*, 2018. 3.5.1d, 5.1.1, 5.6, 5.6

[45] Yuxin Chen, Shervin Javdani, Amin Karbasi, J. Andrew Bagnell, Siddhartha S. Srinivasa, and Andreas Krause. Submodular surrogates for value of information. In *AAAI*, pages 3511–3518, 2015. 2.1.1b, 3.3.3, A.1, A.1

[46] Han-Lim Choi, Luc Brunet, and Jonathan P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Trans. Robotics*, 25(4):912–926, 2009. 3.6.5, 3.6.5a, 7.4, 8, 8.4.1, A.6, 9

[47] Sanjiban Choudhury, Ashish Kapoor, Gireeja Ranade, Sebastian Scherer, and Debadeepta Dey. Adaptive information gathering via imitation learning. In *Proc. of Robot.: Sci. and Syst.*, 2017. 2.1.1c

[48] Sanjiban Choudhury, Mohak Bhardwaj, Sankalp Arora, Ashish Kapoor, Gireeja Ranade, Sebastian Scherer, and Debadeepta Dey. Data-driven planning via imitation learning. *Intl. Journal of Robotics Research*, 37(13-14):1632–1672, 2018. 7.3.3c, 7.3.3d, A.1, 2

[49] Timothy H. Chung, Geoffrey A. Hollinger, and Volkan Isler. Search and pursuit-evasion in mobile robotics. *Auton. Robots*, 31(4):299, 2011. 1

[50] Titus Cieslewski, Elia Kaufmann, and Davide Scaramuzza. Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Vancouver, Canada, September 2017. IEEE. 3.1.4

[51] Andrew Clark and Radha Poovendran. A submodular optimization framework for leader selection in linear multi-agent systems. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 3614–3621. IEEE, 2011. 3.5.3a

[52] Oliver M. Cliff, Robert Fitch, Salah Sukkarieh, Debra L. Saunders, and Robert

Heinsohn. Online localization of radio-tagged wildlife with an autonomous aerial robot system. In *Proc. of Robot.: Sci. and Syst.*, Rome, Italy, July 2015. 6.1

[53] Micah Corah and Nathan Michael. Active estimation of mass properties for safe cooperative lifting. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Singapore, May 2017. IEEE. 3.3.1, 8.3

[54] Micah Corah and Nathan Michael. Efficient online multi-robot exploration via distributed sequential greedy assignment. In *Proc. of Robot.: Sci. and Syst.*, Cambridge, MA, 2017. 4

[55] Micah Corah and Nathan Michael. Distributed submodular maximization on partition matroids for planning on large sensor networks. In *Proc. of the IEEE Conf. on Decision and Control*, Miami, FL, December 2018. 5, 1

[56] Micah Corah and Nathan Michael. Distributed matroid-constrained submodular maximization for multi-robot exploration: theory and practice. *Auton. Robots*, 43 (2):485–501, 2019. 4

[57] Micah Corah, Cormac O'Meadhra, Kshitij Goel, and Nathan Michael. Communication-efficient planning and mapping for multi-robot exploration in large environments. *IEEE Robot. Autom. Letters*, 4(2):1715–1721, 2019. 1.2, 3.1.4, 3.1.5, 4, 7, 7.4, 7.4, 14, 7.7, 8.2.2, 8.2.4, 8.4.2, 11

[58] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, NY, 2012. 2.2.3, 3.2, 3.2.2, 5.2, 6.2, 6.2.1, 6.2.1, 6.2.2, 6.7, A.2.1

[59] Philip Dames and Vijay Kumar. Autonomous localization of an unknown number of targets without data association using teams of mobile sensors. *IEEE Transactions on Automation Science and Engineering*, 12(3):850–864, 2015. 3.3.1

[60] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. 10

[61] Robin Deits and Russ Tedrake. Efficient mixed-integer planning for uavs in cluttered environments. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 42–49. IEEE, 2015. 1

[62] Jeffrey Delmerico, Stefan Isler, Reza Sabzevari, and Davide Scaramuzza. A comparison of volumetric information gain metrics for active 3D object reconstruction. *Auton. Robots*, 42(2):197–208, 2018. 3.1.3, 3.1.4, 7.3, A.4

[63] Mihir Dharmadhikari, Tung Dang, Lukas Solanka, Johannes Loje, Huan Nguyen, Nikhil Khedekar, and Kostas Alexis. Motion primitives-based path planning for fast and agile exploration using aerial robots. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Paris, France, May 2020. 7.3.2a

[64] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer Society*, 22(6):46–57, 1989. 4.2.2

[65] FAA. FAA aerospace forecast: Fiscal years 2019–2039. https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/FY2019-39_FAA_Aerospace_Forecast.pdf, 2019. 1

[66] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998. 3.5.3a, 5.1.1

[67] Yuval Filmus and Justin Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. In *Proc. of the IEEE Annu. Symp. Found. Comput. Sci.*, New Brunswick, NJ, October 2012. 3.5.3b

[68] Conor Finn. A new framework for decomposing multivariate information. 2020. 3.5.1d

[69] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions-II. *Polyhedral Combinatorics*, 8:73–87, 1978. 1.2, 1.4, 2, 3.5.3b, 4.1, 4.5.1, 6.1, 6.5.1, 9

[70] Stephan Foldes and Peter L. Hammer. Submodularity, supermodularity, and higher-order monotonicities of pseudo-boolean functions. *Mathematics of Operations Research*, 30(2):453–461, 2005. 3.5.1c, 3.5.1d, 5.1.1, 5.1.1, 5.6, 6.5.2a, 7.3.1a, B.1, B.1

[71] David Fridovich-Keil, Jaime F. Fisac, and Claire J. Tomlin. Safe and complete real-time planning and exploration in unknown environments. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Montreal, Canada, May 2019. 3.1.1

[72] Vibhav Nagaraj Ganesh. Robust distributed 3D mapping with communication constraints. Technical Report CMU-RI-TR-17-32, Carnegie Mellon University, Pittsburgh, PA, May 2017. 1.2

[73] Stephen M. George, Wei Zhou, Harshavardhan Chenji, Myounggyu Won, Yong Oh Lee, Andria Pazarloglou, Radu Stoleru, and Prabir Barooah. DistressNet: a wireless ad hoc and sensor network architecture for situation management in disaster response. *IEEE Communications Magazine*, 48(3):128–136, 2010. 1.1

[74] Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *Proc. of the Symposium on Discrete Algorithms*, Philadelphia, PA, January 2011. 4.4.4

[75] Bahman Gharesifard and Stephen L. Smith. Distributed submodular maximization with limited information. *IEEE Trans. Control Netw. Syst.*, 2017. to appear. 1.2, 3.6.3, 3.1, 3.6.5, 3.6.5c, 3.6, 4.4.3, 5, 5.3, 5.3, 5.8, 6.4.2, 9

[76] Kshitij Goel, Micah Corah, Curtis Boirum, and Nathan Michael. Fast exploration using multirotors: Analysis, planning, and experimentation. In *Field and Service Robotics*, Tokyo, Japan, August 2019. 1, 1.2, 2.1.1a, 3.1.1, 3.1.3, 3.3.3, 4, 7.3.2a

[77] Kshitij Goel, Micah Corah, and Nathan Michael. Fast exploration using multirotors: Analysis, planning, and experimentation. Technical Report CMU-RI-TR-19-03, Carnegie Mellon University, Pittsburgh, PA, February 2019. 1, 1.2, 2.1.1d, 3.1.1, 3.1.3, 3.3.3

[78] Michel Goemans and Jan Vondrák. Stochastic covering and adaptivity. In *Latin American symposium on theoretical informatics*, pages 532–543. Springer, 2006. 3.3.3

[79] 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. 1, 2, 2.1, 2, 2.1.1b, 2.1.1c, 2.1.1f, 2.1.2, 2.1.3, 3.3.3, 3.5.3e, 7.3.1a, 7.6.1, 7.7.2, A.1, A.1, A.1

[80] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *Adv. in Neural Inf. Process. Syst.*, pages 766–774, 2010. 2.1.1b, 1

[81] Pranava R. Goundan and Andreas S. Schulz. Revisiting the greedy approach to submodular set function maximization. *Optim. online*, (1984):1–25, 2007. 2, 3.6.5b

[82] David Grimsman, Mohd Shabbir Ali, Joao P. Hespanha, and Jason R. Marden. The impact of information in greedy submodular maximization. *IEEE Transactions on Control of Network Systems*, 2018. 1.2, 3.5.3d, 3.6.3, 3.1, 3.6.5, 3.6.5c, 3.6, 4.4.3, 5, 5.1.1, 5.3, 5.8, 6.4.2, 9, B.4

[83] David Grimsman, João P. Hespanha, and Jason R. Marden. Strategic information sharing in greedy submodular maximization. In *Proc. of the IEEE Conf. on Decision and Control*, pages 2722–2727, Miami, FL, December 2018. IEEE. 1.2

[84] Anupam Gupta and Roie Levin. Fully-dynamic submodular cover with bounded recourse. *arXiv preprint arXiv:2009.00800*, 2020. 3.5.1d, 5.8

[85] Anupam Gupta, Viswanath Nagarajan, and R. Ravi. Approximation algorithms for optimal decision trees and adaptive tsp problems. *Mathematics of Operations Research*, 42(3):876–896, 2017. A.1

[86] Panu Harmo, Tapio Taipalus, Jere Knuuttila, José Vallet, and Aarne Halme. Needs and solutions-home automation and service robots for the elderly and disabled. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 3201–3206. IEEE, 2005. 1

[87] Theia Henderson, Vivienne Sze, and Sertac Karaman. An efficient and continuous approach to information-theoretic exploration. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, May 2020. 3, 7, 7.3, 7.3.2, 7.3.2a, 9.1.2, A.4

[88] Trevor Henderson. *A continuous approach to information-theoretic exploration with range sensors*. PhD thesis, Massachusetts Institute of Technology, 2019. 7, 7.3.3b, A.4.1

[89] Geoffrey Hollinger, Sanjiv Singh, Joseph Djugash, and Athanasios Kehagias. Efficient multi-robot search for a moving target. *Intl. Journal of Robotics Research*, 28(2):201–219, 2009. 1.2, 3.4.1, 6.1

[90] Geoffrey A. Hollinger and Gaurav S. Sukhatme. Sampling-based robotic information gathering algorithms. *Intl. Journal of Robotics Research*, 33(9):1271–1287, 2014. 2.1.2a, 3.3.2, 6.1, 6.4.4

[91] Geoffrey A. Hollinger, Brendan Englot, Franz Hover, Urbashi Mitra, and Gaurav S. Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *Intl. Journal of Robotics Research*, 32(1):3–18, 2013. 1, 3.3.3

[92] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on oc-

trees. *Auton. Robots*, 34(3):189–206, 2013. A.3

[93] Vadim Indelman, Luca Carlone, and Frank Dellaert. Planning under uncertainty in the continuous domain: a generalized belief space approach. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 6763–6770. IEEE, 2014. 1.2, 3.1.2, 3.3.1, 6.4.4

[94] Rishabh Iyer, Ninad Khargoankar, Jeff Bilmes, and Himanshu Asanani. Submodular combinatorial information measures with applications in machine learning. *arXiv preprint arXiv:2006.15412*, 2020. 3.5.1d

[95] Lucas Janson, Tommy Hu, and Marco Pavone. Safe motion planning in unknown environments: Optimality benchmarks and tractable policies. In *Proc. of Robot.: Sci. and Syst.*, Pittsburgh, PA, 2018. 2.1.1d, 3.1.1

[96] Shervin Javdani, Matthew Klingensmith, J. Andrew Bagnell, Nancy S. Pollard, and Siddhartha S. Srinivasa. Efficient touch based localization through submodularity. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Karlsruhe, Germany, May 2013. 3, 3.3.3, A.1, A.1, A.1

[97] Shervin Javdani, Yuxin Chen, Amin Karbasi, Andreas Krause, Drew Bagnell, and Siddhartha Srinivasa. Near optimal bayesian active learning for decision making. In *Artificial Intelligence and Statistics*, pages 430–438, 2014. A.1

[98] Luke B. Johnson, Han-Lim Choi, Sameera S. Ponda, and Jonathan P. How. Decentralized task allocation using local information consistency assumptions. *J. Aero. Inf. Syst.*, 14(2):103–122, 2017. 3.6.5a, 5

[99] Heather Jones. *Using Planned View Trajectories to Build Good Models of Planetary Features under Transient Illumination*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, May 2016. 7.2

[100] Heather Jones, Wennie Tabib, and William L. Whittaker. Planning views to model planetary pits under transient illumination. In *IEEE Aerospace Conference*. IEEE, 2015. 7.2

[101] Stefan Jorgensen, Robert H. Chen, Mark B. Milam, and Marco Pavone. The team surviving orienteers problem: Routing robots in uncertain environments with survival constraints. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 227–234. IEEE, 2017. 3.5.3a

[102] Stefan Jorgensen, Robert H. Chen, Mark B. Milam, and Marco Pavone. The matroid team surviving orienteers problem: Constrained routing of heterogeneous teams with risky traversal. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Vancouver, Canada, September 2017. 1.2, 3.4.1, 5, 6.1

[103] Brian J. Julian, Sertac Karaman, and Daniela Rus. On mutual information-based control of range sensing robots for mapping applications. *Intl. Journal of Robotics Research*, 33(10):1357–1392, 2014. 3.1, 3.1.3, 3.3.1, 4.2.2, 6, 7, 7.3, 7.3.2, 7.3.2, 7.3.3b, A.4, 8

[104] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning

and acting in partially observable stochastic domains. *Artificial intelligence*, 101 (1-2):99–134, 1998. 2.1.1f

[105] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972. 5.5.1

[106] Nitish Korula, Vahab Mirrokni, and Morteza Zadimoghaddam. Online submodular welfare maximization: Greedy beats 1/2 in random order. *SIAM Journal on Computing*, 47(3):1056–1086, 2018. 3.5.1d, 5.1.1, 5.8

[107] Andreas Krause and Carlos E. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proc. of the Conf. on Uncertainty in Artif. Intell.*, Edinburgh, Scotland, 2005. 3.5.1a, 3.5.3a, 6.2.1

[108] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 2–10. ACM, 2006. 6

[109] Andreas Krause, Jure Leskovec, Carlos Guestrin, Jeanne VanBriesen, and Christos Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6): 516–526, 2008. 3.5.3e, 7.6.1, 7.7.2

[110] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.*, 9:235–284, 2008. 4.1

[111] Miroslav Kulich, Jan Faigl, and Libor Přeučil. On distance utility in the exploration task. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Shanghai, China, May 2011. 3.1.4

[112] Rajiv Ranjan Kumar, Pradeep Varakantham, and Akshat Kumar. Decentralized planning in stochastic environments with submodular rewards. In *AAAI*, pages 3021–3028, 2017. 3.3.3, 3.6.5b

[113] Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast greedy algorithms in mapreduce and streaming. *ACM Transactions on Parallel Computing (TOPC)*, 2(3):1–22, 2015. 3.6.4

[114] Richard E. Ladner and Michael J. Fischer. Parallel prefix computation. *Journal of the ACM (JACM)*, 27(4):831–838, 1980. 4.4.2, 4.4.5

[115] Jean-Philippe Lang. Open-mesh. https://www.open-mesh.org/projects/open-mesh/wiki, 2020. Accessed: 08-07-2020. 8.1.3

[116] Mikko Lauri and Risto Ritala. Planning for robotic exploration based on forward simulation. *Robot. Auton. Syst.*, 83:15–31, 2016. 2.1.2, 3.1.1, 3.1.3, 3.3.3, 3.4.1b, 4.2.3, 4.3, 6.1, 6.4.4

[117] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery*

and data mining, pages 420–429, 2007. 7.6.1, 7.7.2

[118] Peter Zhi Xuan Li, Zhengdong Zhang, Sertac Karaman, and Vivienne Sze. High-throughput computation of Shannon mutual information on chip. In *Proc. of Robot.: Sci. and Syst.*, Freiburg, Germany, 2019. 7, 7.3.2a

[119] Wenzheng Li, Paul Liu, and Jan Vondrak. A polynomial lower bound on adaptive complexity of submodular maximization. *arXiv preprint arXiv:2002.09130*, 2020. 3.6.3

[120] Sikang Liu, Michael Watterson, Sarah Tang, and Vijay Kumar. High speed navigation for quadrotors with limited onboard sensing. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Stockholm, Sweden, May 2016. 1, 2.1.1d

[121] Wenhao Luo and Ashish Kapoor. Airborne collision avoidance systems with probabilistic safety barrier certificates. In *Proceedings of Workshop on Safety and Robustness in Decision-making at the Thirty-third Conference on Neural Information Processing Systems (NeurIPS 2019)*, December 2019. 2.1.1d, 8.2.5d

[122] Wenhao Luo, Shehzaman S. Khatib, Sasanka Nagavalli, Nilanjan Chakraborty, and Katia Sycara. Distributed knowledge leader selection for multi-robot environmental sampling under bandwidth constraints. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 5751–5757, Daejon, Korea, October 2016. IEEE. A.6

[123] Wenhao Luo, Changjoo Nam, George Kantor, and Katia Sycara. Distributed environmental modeling and adaptive sampling for multi-robot sensor coverage. In *Proc. of the International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1488–1496, 2019. 1.2

[124] Nancy A. Lynch. *Distributed algorithms*. Elsevier, 1996. 8.1.1, 8.2.1

[125] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robot. Autom. Mag.*, 19(3):20–32, September 2012. 4.6.2a

[126] Jason R. Marden. The role of information in multiagent coordination. In *Proc. of the IEEE Conf. on Decision and Control*, 2015. 3.6.5b

[127] George Mathew and Igor Mezić. Metrics for ergodicity and design of ergodic dynamics for multi-agent systems. *Physica D: Nonlinear Phenomena*, 240(4-5):432–442, 2011. 2.2.2

[128] Nathan Michael, Michael M. Zavlanos, Vijay Kumar, and George J. Pappas. Maintaining connectivity in mobile robot networks. In *Experimental Robotics*, pages 117–126. Springer, 2009. 3.1.6

[129] Lauren M. Miller, Yonatan Silverman, Malcolm A. MacIver, and Todd D. Murphey. Ergodic exploration of distributed information. *IEEE Trans. Robotics*, 32(1):36–52, 2015. 2.2.2

[130] Stephen Miller, Jur Van Den Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and Pieter Abbeel. A geometric approach to robotic laundry folding. *Intl. Journal of Robotics Research*, 31(2):249–267, 2012. 1

166

[131] David L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on communications*, 39(10):1482–1493, 1991. 8.2.1

[132] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pages 234–243. Springer, 1978. 3.5.3e, 7.6.1

[133] Baharan Mirzasoleiman, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Adv. in Neural Inf. Process. Syst.*, Stateline, Nevada, December 2013. 3.6.4, 4.4.3

[134] Derek Mitchell and Nathan Michael. Persistent multi-robot mapping in an uncertain environment. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Montreal, Canada, May 2019. 3.1.5, 7, 7.4, 7.8.1

[135] Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Decentralized submodular maximization: Bridging discrete and continuous settings. In *Proc. of the Intl. Conf. on Machine Learning*, 2018. 3.6.5

[136] Robin R. Murphy and Alexander Kleiner. A community-driven roadmap for the adoption of safety security and rescue robots. In *IEEE Intl. Symp. on Safety, Security, and Rescue Robotics*, pages 1–5. IEEE, 2013. 1.1

[137] Robin R. Murphy, Satoshi Tadokoro, Daniele Nardi, Adam Jacoff, Paolo Fiorini, Howie Choset, and Aydan M. Erkmen. Search and rescue robotics. *Springer handbook of robotics*, pages 1151–1173, 2008. 1.1, 3.1.2

[138] Robin R. Murphy, Satoshi Tadokoro, and Alexander Kleiner. Disaster robotics. *Springer Handbook of Robotics*, pages 1577–1604, 2016. 1, 1.1, 3.1.6, 8.1.3

[139] Erik Nelson, Micah Corah, and Nathan Michael. Environment model adaptation for mobile robot exploration. *Auton. Robots*, 42(2):257–272, 2018. 3.1.3

[140] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions-I. *Math. Program.*, 14(1):265–294, 1978. 1.2, 3.5.3a

[141] George L. Nemhauser and Laurence A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3 (3):177–188, 1978. 3.5.3a

[142] Gennaro Notomista, Sebastian F. Ruf, and Magnus Egerstedt. Persistification of robotic tasks using control barrier functions. *IEEE Robot. Autom. Letters*, 3(2): 758–763, 2018. 2.1.1d

[143] Reza Olfati-Saber and Richard M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on automatic control*, 49(9):1520–1533, 2004. 3.6.5

[144] Timothy Patten. *Active Object Classification from 3D Range Data with Mobile Robots*. PhD thesis, The University of Sydney, 2017. 4.3

[145] Chiara Piacentini, Sara Bernardini, and J. Christopher Beck. Autonomous target search with multiple coordinated UAVs. *Journal of Artificial Intelligence Research*, 65:519–568, 2019. 6.1

[146] Sameera S. Ponda. *Robust distributed planning strategies for autonomous multi-agent teams.* PhD thesis, Massachusetts Institute of Technology, 2012. 3.6.5a, 8

[147] Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *International Workshop on Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004. 2.1.1d

[148] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, Kobe, Japan, May 2009. 4.6.1b, 8.4.2

[149] Srikumar Ramalingam, Chris Russell, L'ubor Ladickỳ, and Philip H.S. Torr. Efficient minimization of higher order submodular functions using monotonic boolean functions. *Discrete Applied Mathematics*, 220:1–19, 2017. 5.1.1, 5.6

[150] James B. Rawlings and Kenneth R. Muske. The stability of constrained receding horizon control. *IEEE Trans. Autom. Control*, 38(10):1512–1516, 1993. 2.1.1d, 2.1.2, 3.1.1, 4.5.2

[151] Tal Regev and Vadim Indelman. Multi-robot decentralized belief space planning in unknown environments via efficient re-evaluation of impacted paths. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Daejeon, Korea, October 2016. 1.2, 3.4.1, 4.1, 8

[152] Charles Richter and Nicholas Roy. Safe visual navigation via deep learning and novelty detection. In *Proc. of Robot.: Sci. and Syst.*, Cambridge, MA, 2017. 3.1.1

[153] Mike Roberts, Debadeepta Dey, Anh Truong, Sudipta Sinha, Shital Shah, Ashish Kapoor, Pat Hanrahan, and Neel Joshi. Submodular trajectory optimization for aerial 3d scanning. *arXiv preprint arXiv:1705.00703*, 2017. 3.3.1, 5

[154] Alexander Robey, Arman Adibi, Brent Schlotfeldt, George J. Pappas, and Hamed Hassani. Optimal algorithms for submodular maximization with distributed constraints. *arXiv preprint arXiv:1909.13676*, 2019. 3.6.5, 9.1.1

[155] Tim Roughgarden. Intrinsic robustness of the price of anarchy. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 513–522. ACM, 2009. 3.6.5b

[156] Allison Ryan and J. Karl Hedrick. Particle filter based information-theoretic active sensing. *Robot. Auton. Syst.*, 58(5):574–584, 2010. 2.2.4, 6.7

[157] Mahyar Salek, Shahin Shayandeh, and David Kempe. You share, I share: Network effects and economic incentives in P2P file-sharing systems. In *International Workshop on Internet and Network Economics*, pages 354–365. Springer, 2010. 5.6, 5.8

[158] Manish Saroya, Graeme Best, and Geoffrey A. Hollinger. Online exploration of tunnel networks leveraging topological cnn-based world predictions. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Las Vegas, NV, October 2020. 7.3.3c

[159] Yash Satsangi, Shimon Whiteson, Frans A. Oliehoek, and Matthijs T. J. Spaan. Ex-

ploiting submodular value functions for scaling up active perception. *Auton. Robots*, 42(2):209–233, 2018. 3.3.3, 9.1.2

[160] Brent Schlotfeldt, Dinesh Thakur, Nikolay Atanasov, Vijay Kumar, and George J. Pappas. Anytime planning for decentralized multi-robot active information gathering. *IEEE Robot. Autom. Letters*, 3766(c):1–8, 2018. 2.1.2, 8

[161] Brent Schlotfeldt, Vasileios Tzoumas, Dinesh Thakur, and George J. Pappas. Resilient active information gathering with mobile robots. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 4309–4316. IEEE, 2018. 1.2

[162] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003. 3.5.2, 3.5.2

[163] Mac Schwager, Philip Dames, Daniela Rus, and Vijay Kumar. A multi-robot control policy for information gathering in the presence of unknown hazards. In *Proc. of the Intl. Sym. of Robot. Research*, Flagstaff, AZ, August 2011. 6

[164] Pau Segui-Gasco, Hyo-Sang Shin, Antonios Tsourdos, and V. J. Segui. Decentralised submodular multi-robot task allocation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 2829–2834, Hamburg, Germany, September 2015. IEEE. 3.6.5

[165] Kunal Shah and Mac Schwager. GRAPE: Geometric risk-aware pursuit-evasion. *Robot. Auton. Syst.*, 2019. 6.1, 1

[166] Hyo-Sang Shin, Teng Li, and Pau Segui-Gasco. Sample greedy based task allocation for multiple robot systems. *arXiv preprint arXiv:1901.03258*, 2019. 3.6.5a

[167] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Adv. in Neural Inf. Process. Syst.*, pages 1–9, 2010. 3.3.3

[168] Reid Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrun, and Håkan Younes. Coordination for multi-robot exploration and mapping. In *Assoc. for Adv. of Artif. Intell.*, pages 852–858, 2000. 7, 7.3, 7.3.3a, 7.4, 7.4, 7.8.1, A.4

[169] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William J. Kaiser. Efficient informative sensing using multiple robots. *J. Artif. Intell. Res.*, 34:707–755, 2009. 1.2, 2, 2.1.2a, 3.3.2, 3.4.1, 3.4.1, 2, 3.4.1a, 3.4.1b, 4.1, 4.3, 4.4, 4.4.1, 2, 5, 6.1, 6.4.1, 6.4.4, 6.5.1, 1, 9, B.3

[170] Agusti Solanas and Miguel Angel Garcia. Coordinated multi-robot exploration through unsupervised clustering of unknown space. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, volume 1, pages 717–721. IEEE, 2004. 3.1.5

[171] Matthijs T. J. Spaan, Tiago S. Veiga, and Pedro U. Lima. Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *Autonomous Agents and Multi-Agent Systems*, 29(6):1157–1185, 2015. 3.3.3

[172] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using Rao-Blackwellized particle filters. In *Proc. of Robot.: Sci. and Syst.*, 2005. 3.1.1, 3.1.2

[173] Soumya Sudhakar, Sertac Karaman, and Vivienne Sze. Balancing actuation and computing energy in motion planning. 2

[174] Fouad Sukkar, Graeme Best, Chanyeol Yoo, and Robert Fitch. Multi-robot region-of-interest reconstruction with Dec-MCTS. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 9101–9107. IEEE, 2019. 8

[175] Haoyuan Sun, David Grimsman, and Jason R Marden. Distributed submodular maximization with parallel execution. In *Proc. of the Amer. Control Conf.*, Denver, CO, 2020. 3.6.5c

[176] Yoonchang Sung, Ashish Kumar Budhiraja, Ryan K. Williams, and Pratap Tokekar. Distributed simultaneous action and target assignment for multi-robot multi-target tracking. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Brisbane, Australia, May 2018. IEEE. 3.4

[177] Yoonchang Sung, Ashish Kumar Budhiraja, Ryan K. Williams, and Pratap Tokekar. Distributed assignment with limited communication for multi-robot multi-target tracking. *Auton. Robots*, 2019. 6.1

[178] Wennie Tabib and Nathan Michael. Simultaneous localization and mapping of subterranean voids with Gaussian mixture models. In *Field and Service Robotics*, 2019. 8.4.2

[179] Wennie Tabib, Micah Corah, Nathan Michael, and Red Whittaker. Computationally efficient information-theoretic exploration of pits and caves. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Daejeon, Korea, October 2016. 2, 3.1.3, 4.6.1b, 4.6.2a, 7, 7.3.3b, 10, 7.2

[180] Wennie Tabib, Kshitij Goel, John Yao, Curtis Boirum, and Nathan Michael. Autonomous cave surveying with an aerial robot. *arXiv preprint arXiv:2003.13883*, 2020. 8.4.2

[181] Michael Tatum. Communications coverage in unknown underground environments. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, June 2020. 3.1.6

[182] Pratap Tokekar, Volkan Isler, and Antonio Franchi. Multi-target visual tracking with aerial robots. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Chicago, IL, September 2014. 6.1

[183] Alberto Valero-Gómez, Javier V. Gómez, Santiago Garrido, and Luis Moreno. Fast marching methods in path planning. *IEEE Robot. Autom. Mag.*, 2013. 7.4

[184] Adrian Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 416–425. IEEE, 2002. 3.6.5b, 7.8.1

[185] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74. ACM, 2008. 3.5.3b, 11

[186] Chaoqun Wang, Delong Zhu, Teng Li, Max Q.-H. Meng, and Clarence W. de Silva.

Efficient autonomous robotic exploration with semantic road map in indoor environments. *IEEE Robot. Autom. Letters*, 4(3):2989–2996, 2019. 8.2.4

[187] Li Wang, Aaron D. Ames, and Magnus Egerstedt. Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots. In *Proc. of the IEEE Conf. on Decision and Control*, pages 2659–2664, Las Vegas, Nevada, December 2016. IEEE. 3.1.6

[188] Li Wang, Aaron D. Ames, and Magnus Egerstedt. Safe certificate-based maneuvers for teams of quadrotors using differential flatness. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 3293–3298, Singapore, May 2017. IEEE. 8.2.5d

[189] Li Wang, Aaron D. Ames, and Magnus Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Trans. Robotics*, 33(3):661–674, 2017. 8.2.5d

[190] Yiming Wang, Stuart James, Elisavet Konstantina Stathopoulou, Carlos Beltrán-González, Yoshinori Konishi, and Alessio Del Bue. Autonomous 3-D reconstruction, mapping, and exploration of indoor environments with a robotic arm. *IEEE Robot. Autom. Letters*, 4(4):3340–3347, 2019. 7.3.3c

[191] Zengfu Wang, Bill Moran, Xuezhi Wang, and Quan Pan. An accelerated continuous greedy algorithm for maximizing strong submodular functions. *Journal of Combinatorial Optimization*, 30(4):1107–1124, 2015. 3.5.1d, 1, 5.4, 5.6, 5.8

[192] Michael Watterson and Vijay Kumar. Safe receding horizon control for aggressive mav flight with limited range sensing. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3235–3240. IEEE, 2015. 2.1.1d

[193] Jason L. Williams. *Information Theoretic Sensor Management*. PhD thesis, Massachusetts Institute of Technology, 2007. 2

[194] Ryan K. Williams, Andrea Gasparri, and Giovanni Ulivi. Decentralized matroid optimization for topology constraints in multi-robot allocation problems. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Singapore, May 2017. 3.4.1, 3.6.5, 3.6.5a, 8, A.6

[195] Christian Witting, Marius Fehr, Rik Bähnemann, Helen Oleynikova, and Roland Siegwart. History-aware autonomous exploration in confined environments using mavs. In *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Madrid, Spain, October 2018. IEEE. 8.2.4

[196] Uland Wong, Warren Whittaker, Heather Jones, and Red Whittaker. NASA planetary pits and caves analog dataset. https://ti.arc.nasa.gov/dataset/caves/, December 2014. Accessed: 07-01-2020. 7.2

[197] Jiahao Xie, Chao Zhang, Zebang Shen, Chao Mi, and Hui Qian. Decentralized gradient tracking for continuous DR-submodular maximization. In *Proc. of the Intl. Conf. on Artif. Intell. and Stat.*, pages 2897–2906, 2019. 3.6.5, 9.1.1

[198] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. of the Intl. Sym. on Comput. Intell. in Robot. and Autom.*, Monterey, CA, July 1997. 2.1, 2.1.1e, 3.1.1, 3.1.4, 7, 7.4

[199] Luke Yoder and Sebastian Scherer. Autonomous exploration for infrastructure modeling with a micro aerial vehicle. In *Field and Service Robotics*, pages 427–440. Springer, 2016. 7.3.1

[200] Haifeng Zhang and Yevgeniy Vorobeychik. Submodular optimization with routing constraints. In *Assoc. for Adv. of Artif. Intell.*, 2016. 2.1.2a, 3.3.2, 6.4.4, A.4

[201] Zhengdong Zhang, Theia Henderson, Sertac Karaman, and Vivienne Sze. FSMI: Fast computation of shannon mutual information for information-theoretic mapping. *Intl. Journal of Robotics Research*, 2020. 3.1.3, 4.2.2, 7, 7.3.2, 7.3.2a, 7.3.3b

[202] Lifeng Zhou and Pratap Tokekar. Sensor assignment algorithms to improve observability while tracking targets. *IEEE Trans. Robotics*, 35(5):1206–1219, 2019. 6.1

[203] Lifeng Zhou, Vasileios Tzoumas, George J. Pappas, and Pratap Tokekar. Resilient active target tracking with multiple robots. *IEEE Robot. Autom. Letters*, 4(1):129–136, 2019. 1.2