Boston University

OpenBU **http://open.bu.edu**

Boston University Theses & Dissertations Boston University Theses & Dissertations

2018

Models and algorithms for multi-agent search problems

https://hdl.handle.net/2144/32075

Downloaded from OpenBU. Boston University's institutional repository.

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

MODELS AND ALGORITHMS FOR MULTI-AGENT SEARCH PROBLEMS

by

HUANYU DING

B.Eng., Zhejiang University, 2012

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2018

© 2018 by
HUANYU DING
All rights reserved

Approved by

First Reader

David A. Castañón, Ph.D.
Professor of Electrical and Computer Engineering
Professor of Systems Engineering

Second Reader

Venkatesh Saligrama, Ph.D.
Professor of Electrical and Computer Engineering
Professor of Systems Engineering
Professor of Computer Science

Third Reader

Ioannis Ch. Paschalidis, Ph.D.
Professor of Electrical and Computer Engineering
Professor of Biomedical Engineering
Professor of Systems Engineering

Fourth Reader

Bobak Nazer, Ph.D.
Associate Professor of Electrical and Computer Engineering
Associate Professor of Systems Engineering

Acknowledgments

My foremost gratitude goes to my Ph.D. advisor, Professor David Castañón. Without his enduring guidance and support, none of the results in this dissertation is possible. I am extremely fortunate to have David as my advisor and friend for the Ph.D. journey. He is familiar with every detail of my research, and is always available and willing to give advice. His first-class intelligence, passion for research, and deep insights for the problems have been great sources of inspiration. I have benefited from discussions with him much deeper than from any books. And conversations with him about life experiences have always been delightful.

I would like to thank my committee members, Professor Venkatesh Saligrama, Professor Yannis Paschalidis, and Professor Bobak Nazer, for giving me many useful comments and suggestions to improve the quality of this dissertation, and for providing much help and encouragement during my stay at BU. I would also like to thank my undergraduate advisor at Zhejiang University, Professor Jiming Chen, for encouraging me to take the path of a Ph.D.

Boston is a fun city to live and study. I have been fortunate to meet and befriend with many interesting people here. I thank all the happiness they have brought to my life. For fear of missing anyone, I choose not to list their names here.

Last but not least, I would like to thank my loving family for everything they have done for me. I am blessed to have my wife in the past four and a half years as well as the rest of my life. I am proud to be the son of my parents, who have always taught me to be a good person. I also want to thank my parents-in-law for putting their faith and trust in me.

MODELS AND ALGORITHMS FOR MULTI-AGENT SEARCH PROBLEMS

HUANYU DING

Boston University, College of Engineering, 2018

Major Professor: David A. Castañón, Ph.D.
Professor of Electrical and Computer Engineering
Professor of Systems Engineering

ABSTRACT

The problem of searching for objects of interest occurs in important applications ranging from rescue, security, transportation, to medicine. With the increasing use of autonomous vehicles as search platforms, there is a need for fast algorithms that can generate search plans for multiple agents in response to new information. In this dissertation, we develop new techniques for automated generation of search plans for different classes of search problems.

First, we study the problem of searching for a stationary object in a discrete search space with multiple agents where each agent can access only a subset of the search space. In these problems, agents can fail to detect an object when inspecting a location. We show that when the probabilities of detection only depend on the locations, this problem can be reformulated as a minimum cost network optimization problem, and develop a fast specialized algorithm for the solution. We prove that our algorithm finds the optimal solution in finite time, and has worst-case computation performance that is faster than general minimum cost flow algorithms. We then generalize it to the case where the probabilities of detection depend on the agents

and the locations, and propose a greedy algorithm that is $\frac{1}{2}$-approximate.

Second, we study the problem of searching for a moving object in a discrete search space with multiple agents where each agent can access only a subset of a discrete search space at any time and agents can fail to detect objects when searching a location at a given time. We provide necessary conditions for an optimal search plan, extending prior results in search theory. For the case where the probabilities of detection depend on the locations and the time periods, we develop a forward-backward iterative algorithm based on coordinate descent techniques to obtain solutions. To avoid local optimum, we derive a convex relaxation of the dynamic search problem and show this can be solved optimally using coordinate descent techniques. The solutions of the relaxed problem are used to provide random starting conditions for the iterative algorithm. We also address the problem where the probabilities of detection depend on the agents as well as the locations and the time periods, and show that a greedy-style algorithm is $\frac{1}{2}$-approximate.

Third, we study problems when multiple objects of interest being searched are physically scattered among locations on a graph and the agents are subject to motion constraints captured by the graph edges as well as budget constraints. We model such problem as an orienteering problem, when searching with a single agent, or a team orienteering problem, when searching with multiple agents. We develop novel real-time efficient algorithms for both problems.

Fourth, we investigate classes of continuous-region multi-agent adaptive search problems as stochastic control problems with imperfect information. We allow the agent measurement errors to be either correlated or independent across agents. The structure of these problems, with objectives related to information entropy, allows for a complete characterization of the optimal strategies and the optimal cost. We derive a lower bound on the performance of the minimum mean-square error estimator,

and provide upper bounds on the estimation error for special cases. For agents with independent errors, we show that the optimal sensing strategies can be obtained in terms of the solution of decoupled scalar convex optimization problems, followed by a joint region selection procedure. We further consider search of multiple objects and provide an explicit construction for adaptively determining the sensing actions.

Contents

List of Tables

List of Figures

List of Abbreviations

ADP	.....	Approximate Dynamic Programming
DP	.....	Dynamic Programming
FAB	.....	Forward-and-Backward
MST	.....	Minimum Spanning Tree
SEM	.....	Scanning Electron Microscope
SLSQP	.....	Sequential Least Squares Programming
SOP	.....	Single Orienteering Problem
TOP	.....	Team Orienteering Problem
TSP	.....	Traveling Salesperson Problem
UAV	.....	Unmanned Aerial Vehicle
$\mathbb{R}^+$	.....	Non-negative real line
$\mathbb{R}^d$	.....	d-dimensional Euclidean space

Chapter 1

Introduction

1.1 Problem Description

The proliferation of intelligent agents such as robots and UAVs in diverse applications from rescue, security, transportation, to medicine has created a need for automated processing and exploitation of information. An important class of problems in these intelligent systems is the detection and locating of objects of interest using intelligent search agents, known as search problems in the operations research, control, and computer science communities.

Search theory has a long history, dating back to its early application for objects at sea in the 1940s [Koopman, 1946]. Some renowned successful applications of search theory include the search of the lost submarine *Scorpion* in 1968 [Richardson and Stone, 1971], a treasure ship named *Central America* which sank in 1857 with treasure worth hundreds of millions of dollars [Stone, 1992], and the disappeared Air France Flight 447 over the South Atlantic in 2009 [Stone et al., 2014]. Most recently, search theory has been applied to searching for Malaysian Air Flight 370 which went lost in the Indian Ocean in 2014 [Davey et al., 2016].

Other applications of search problems include locating leaked gas, discovering archaeological sites, exploring oil or mineral fields, diagnosing faulty components in a failed system, finding specific cells or tissues in scanning electron microscope (SEM) images, etc.

There are different classes of search problems. In terms of the objects of interests

being searched, there may be a *single object* or *multiple objects*. The objects may be *stationary* or *moving*.

In terms of the search agents, there are *single-agent* or *multi-agent* search problems. A variety of constraints on the agents can be considered. A search agent usually has a limited amount of resources or search effort that it can spend, so there is a *budget* constraint. Also, it may have limited area of influence or sensing range, so *visibility* constraints which only allow the agent to search a subset of the search space can be considered. We refer to search problems with visibility constraints as *sparse search* problems. There may also be *motion* constraints which restrict the agent to searching only some locations right after searching one location. In addition, when the agent moves from one location to another, a *switching cost* that is not negligible comparing to the search cost may be incurred.

In terms of the agent measurement error model, there are typically two categories. In the first category, there are only missed detection type of errors but no false alarm type of errors, i.e., there is a probability for the agent to output a detection signal if the agent happens to search a location containing an object of interest, but there will be no detection signal if the agent searches a location with no object of interest. We refer to measurement error models in this category as *simple error models*. In the second category, both missed detection and false alarm types of errors are assumed. We refer to measurement error models in this category as *complex error models*.

In terms of the objectives, a typical one is to maximize the probability of detection subject to the constraints on the agent such as budget, known as *detection search*. Another objective is to minimize the total search effort needed to detect the object. Other objectives include maximizing the total collected rewards where the information of a location containing an object of interest is measured by a reward at the location.

1.1.1 Search of a Stationary Object with Simple Error Models

A large body of the literature in classical search theory [Koopman, 1946, Koopman, 1953, Koopman, 1956a, Koopman, 1956b, Koopman, 1956c, Gluss, 1959, Staroverov, 1962, Everett, 1963, Zahl, 1963, Arkin, 1964a, Arkin, 1964b, Matula, 1964, Black, 1965, Chew, 1967, Kadane, 1968, Tognetti, 1968, Kadane, 1971, Wagner and Stone, 1974, Kadane and Simon, 1977, Wegener, 1980, Wegener, 1981, Wegener, 1982] deals with search of a stationary object using a single agent. An important common feature of these problems is that they assumed the agent has simple error models with only missed detections but no false alarms, i.e., there is a probability that the agent may not detect the object when querying the location that contains the object, but the agent never mistakes a false object for the real one when searching locations that do not contain the object. Under this assumption, once a detection signal is received, the search operation will terminate. Therefore, when designing a search strategy, one should always expect non-detections. No real feedback is needed and the resulting optimal search strategies are *open-loop*. That is, the optimal search strategies are a sequence of predetermined search actions, not a sequence of search policies that map observation outcomes into search actions. Many of the optimality results were obtained through the Lagrange multiplier method.

Multi-agent search of a stationary object with simple error models was studied by Song and Teneketzis [Song and Teneketzis, 2004]. In their problem, the agents are not subject to visibility or motion constraints, and each agent can freely select any location to search at any time. The agents are homogeneous, thus the probabilities of detection only depend on the locations but not on the agents. They developed an algorithm for constructing an optimal multi-agent search allocation which sorts all the possible marginal probabilities of detection. But the algorithm does not exploit the implicit ordering of the marginal probabilities of detection at each location and

therefore is slow for a large number of locations.

1.1.2 Search of a Moving Object with Simple Error Models

The study of search of a moving object in classical search theory started much later after its stationary counterpart [Pollock, 1970, Dobbie, 1974, Brown, 1980, Stewart, 1980, Washburn, 1983, Eagle, 1984, Eagle and Yee, 1990]. In moving object search, the probabilistic characterization of the object trajectories is known to the search agents. For example, one may assume that the probabilities of all possible object trajectories are known, or assume that the object moves according to a Markov chain whose transition probability matrices at each time period are known.

As in its stationary counterpart, the agent in moving object search is assumed to have simple error models; thus, the optimal search strategies are open-loop.

Most works in classic search theory deal with a single agent. Pollock [Pollock, 1970] and Dobbie [Dobbie, 1974] solved problems where there are just two possible locations. Brown [Brown, 1980] established necessary and sufficient conditions of optimal search plans in discrete time and space, with continuous search effort that can be arbitrarily divided among locations. He also developed an iterative algorithm for constructing search plans. Brown's algorithm was further generalized by Washburn [Washburn, 1983].

Multi-agent search of a moving object with simple error models was studied by Royset and Sato [Royset and Sato, 2010]. The agents in their problem are subject to motion constraints. They developed a cutting plane algorithm which is based on the integer nonlinear programming formulation. However, it does not exploit the special structure of the problem and is slow to converge.

1.1.3 Search of Multiple Objects with Motion and Switching Cost Constraints

Consider the problem of searching among a set of geographically distributed locations where every location may contain an object of interest. The information of a location containing an object of interest is evaluated by a reward value associated with the location. Once a search agent searches a location, it will receive deterministic measurements about the location and thus collect the information reward. However, the agents cannot move freely from one location to another; their motions are constrained. Also, moving from one location to another incurs a switching cost that is not negligible. We are interested in collecting as many rewards as possible without producing too much cost.

This class of problems is known as *orienteering problems*, which are NP-hard, in the operations research and computer science literature [Tsiligirides, 1984, Golden et al., 1987, Chao et al., 1996b]. Exact solutions such as column generation and approximate algorithms with performance guarantees are slow. A variety of heuristics such as in [Tsiligirides, 1984, Golden et al., 1987, Chao et al., 1996a, Silberholz and Golden, 2010, Chao et al., 1996b] and metaheuristics such as tabu search have been proposed.

1.1.4 Adaptive Search with Complex Error Models

The search problems discussed in Sections 1.1.1 and 1.1.2 assume simple error models that include only missed detections but no false alarms. The resulting search strategies are open-loop, which continue until an object is detected.

However, sometimes the agents may mistake a false object for the one we are searching for. For example, when searching for a lost submarine under ocean, a large rock of the similar size and dimensions as the submarine may produce a signal that is similar to a real submarine [Stone, 1975]. Thus, for imperfect agents under noisy

environment, a detection signal may also be false. We refer to such measurement error models where both missed detection and false alarm types of errors exist as *complex error models*.

For search problems with complex error models, open-loop strategies are not optimal. In most cases, the optimal strategies are to determine the search action based on past actions and measurements. In other words, the optimal strategies are *closed-loop* or *adaptive*. Thus, we refer to search problems with complex error models as *adaptive search* problems.

Adaptive search problems with complex error models in discrete search space were studied by Castañón [Castañón, 1995]. He showed that the optimal search strategies are to search either of the two most likely locations given the past information at each step if the agent errors satisfy a simple symmetry condition. He also provided counterexamples of when the symmetry condition is not satisfied, optimal search strategies cannot be easily characterized.

In [Jedynak et al., 2011], Jedynak et al. considered adaptive search in a continuous search space using a single agent that selects a region to query, resulting in a noisy measurement concerning the presence of the object in the query region. The measurement is then used to update the posterior distribution of the object location. The objective is to minimize the posterior differential entropy of the object location after a fixed finite number of measurements. They showed that greedy policies which maximize the one-stage expected entropy reduction are optimal.

The work in [Jedynak et al., 2011] has been generalized in several directions. Sznitman et al. [Sznitman et al., 2013] considered the case where the agent can choose different precision modes with different costs. Rajan et al. [Rajan et al., 2015] considered search of multiple objects where the agent measurements depend on the number of objects in the query region. Tsiligkaridis et al. [Tsiligkaridis et al., 2014] studied

multi-agent adaptive search of a single object where the agent measurement errors are independent but not correlated among agents. They developed characterizations of optimal strategies. For the special case where the agents have binary symmetric error models, they established the equivalence of simultaneous query strategies and sequential query strategies by the team of agents.

1.2 Contributions

Most works in classic search theory [Stone, 1975,Ahlswede and Wegener, 1987,Benkoski et al., 1991] address search problems with a single agent. The main scope of this dissertation is to develop search theory for planning, scheduling and coordinating multiple search agents. Most of the problems studied in this dissertation are in the context of multiple agents.

In this dissertation, we propose new multi-agent search models and develop algorithms to address search problems under these models. Details of the contributions made in this dissertation are:

- We solved the problem of multi-agent sparse search of a stationary object with simple error models. We showed that when the agents are homogeneous, the problem can be reformulated as a minimum cost network flow problem, and developed a fast specialized primal-dual algorithm for the solution. We proved that our algorithm finds the optimal solution in finite time and has worst case computation performance that is much faster than general minimum cost network flow algorithms. We also addressed the problem for the case of heterogeneous agents where detection performance depends on both location and agent, which is known to be NP-hard [Lloyd and Witsenhausen, 1986]. We reduced the problem to a submodular maximization problem over a matroid, and developed an approximate algorithm with guaranteed performance. We also developed a

block coordinate ascent algorithm, and provided an upper bound on the optimal objective value (total probability of detection).

- We studied the problem of multi-agent sparse search of a Markovian moving object with simple error models. We developed necessary conditions for an optimal search plan. Using these necessary conditions, we developed a forward-backward algorithm based on coordinate descent techniques to obtain solutions. For the case where the agents are homogeneous, each iteration of the coordinate descent can be reduced to the solution of a min-cost flow problem. To avoid local minima, we derived a convex relaxation of the dynamic search problem and showed this can be solved optimally using coordinate descent techniques. The solutions of the relaxed problem are used to provide random initial allocations for the iterative algorithm. We also addressed the problem where the probabilities of detection depend on agents, time periods and locations. We reduced the problem to a submodular maximization problem over a matroid, and developed two greedy algorithms with performance guarantees. We also provided a lower bound on the optimal objective value (total probability of non-detection).

- We studied the multi-agent multi-object search problem where the potential objects of interest being searched are physically scattered among locations abstracted as nodes on an undirected graph and the agents are subject to motion and switching cost constraints captured by the graph edges as well as budget constraints. We modeled such problem as a single orienteering problem (SOP), when searching with a single agent, or a team orienteering problem (TOP), when searching with multiple agents. We developed a new class of fast algorithms, based on a decomposition of the orienteering problem into a knapsack assignment problem and a subsequent traveling salesperson problem. We combined greedy algorithms for knapsack problems with the use of spanning trees to esti-

mate traveling salesperson tour lengths to obtain new approximate algorithms for orienteering problems.

- We solved the problem of multi-agent adaptive search with complex error models, formed as a stochastic control problem with imperfect information. The structure of the problem, with objectives related to information entropy, allows for a complete characterization of the optimal strategies and the optimal cost-to-go for the resulting finite-horizon stochastic control problem. We focused on the case where multiple agents can each select its own search region and obtain noisy measurements regarding the presence of the object in the region. We allowed the agent measurement errors to be correlated across agents. We obtained a complete characterization of optimal policies for this dynamic search problem, and provided a constructive algorithm for computing optimal policies in real time based on convex optimization. When the measurement errors of the agents are independent, we showed that the computation of optimal policies can be decoupled into individual scalar convex optimization problems, followed by a joint region selection procedure. We provided simple symmetry conditions where the solutions can be determined analytically. We also considered the problem where individual agents can select the accuracy of their sensing modes with different costs, and derived optimal strategies for this problem. In addition, we developed new results for the case where multiple objects of interest are present.

1.3 Organization of the Dissertation

In Chapter 2, we provide the related background of the search problems studied in this dissertation. In Chapter 3, we present results for problems of multi-agent sparse search of a stationary object with simple error models. In Chapter 4, we present

results for problems of multi-agent sparse search of a Markovian moving object with simple error models. In Chapter 5, we present results for problems of multi-agent multi-object search with motion and switching cost constraints. In Chapter 6, we present results for problems of multi-agent adaptive search with complex error models. We conclude the dissertation in Chapter 7 and provide directions for future work.

Chapter 2

Background

In this chapter, we provide related background of the search problems studied in this dissertation. Section 2.1 reviews previous results on search of a stationary object with simple error models. Section 2.2 reviews previous results on search of a moving object with simple error models. Section 2.3 reviews previous results on search of multiple objects with motion and switching cost constraints. Section 2.4 reviews previous results on adaptive search with complex error models.

2.1 Search of a Stationary Object with Simple Error Models

In classical search theory [Stone, 1975], the search effort was assumed to be either continuous (e.g., one can allocate "half a search" to a location) or only allowed to be discrete (usually number of searches at each location). It was assumed to be applied continuously or over discrete stages. For the purpose of this dissertation, we focus on reviewing results for discrete search effort applied over discrete stages in this section.

A typical problem of *single-agent search of a stationary object with simple error models* in classical search theory [Stone, 1975] is defined as follows:

- $\mathcal{X} = \{1, \ldots, K\}$: a discrete search space of K possible locations.
- $p_{k0}, k \in \mathcal{X}$: prior probability of location k containing the object.
- $p_{kj}, j \geq 1, k \in \mathcal{X}$: probability of detecting the object at location k on the j-th search after failing the previous $j-1$ searches. Assume that p_{kj} is decreasing in

j for each k. A special case is where searches of the same location k are independent with probability of detection α_k; in this case, $p_{kj} = p_{k0}(1-\alpha_k)^{j-1}\alpha_k$.

- $c_{kj} = c_0, j \geq 1, k \in \mathcal{X}$: cost on search effort of the j-th search of location k. It is a positive constant number.
- B: budget on total search effort. It is a positive integer.
- $\boldsymbol{a} = (a_1, \ldots, a_K)$: search plan where a_k is the number of searches performed at location k.
- $C(\boldsymbol{a})$: cost on search effort using plan $\boldsymbol{a}$.
- $F(\boldsymbol{a})$: total probability of detection using plan $\boldsymbol{a}$.

Note that p_{kj} is the probability of detecting the object at location k if the object is at the location. There will be no detection signal if the object is not at the location. In such *simple error models*, there are only missed detections but no false alarms. The search operation will terminate once a detection signal is received. Therefore, feedback is not needed for such problems and the resulting search plans are open-loop.

In this formulation, the objective is to select search plan $\boldsymbol{a}$ to maximize the probability of detection $F(\boldsymbol{a})$ without exceeding the search effort budget:

$$\begin{aligned} \underset{\boldsymbol{a}}{\text{maximize}} \;\; F(\boldsymbol{a}) &= \sum_{k=1}^{K} p_{k0} \sum_{j=1}^{a_k} p_{kj} \\ \text{subject to} \;\; C(\boldsymbol{a}) &= \sum_{k=1}^{K} \sum_{j=1}^{a_k} c_{kj} \leq B \\ a_k &\in \{0, 1, \ldots, B\} \end{aligned}$$

The following result is a necessary and sufficient condition for an optimal search plan. It is based on duality, and exploits the concavity of a piecewise linear interpolation of $F(\boldsymbol{a})$ along with the constraints on $C(\boldsymbol{a})$.

Theorem 2.1.1 (Theorem 4.2.3 of [Stone, 1975]). *Assume that $c_{kj} = c_0, j \geq 1, k \in \mathcal{X}$, i.e., the cost of the j-th search of location k is a constant c_0. Assume that p_{kj} is decreasing in j for each k and search plan $\boldsymbol{a}^*$ has cost $C(\boldsymbol{a}^*)$ such that $0 < C(\boldsymbol{a}) < \infty$. Then a necessary and sufficient condition for $\boldsymbol{a}^*$ to be optimal for cost $C(\boldsymbol{a}^*)$ is that there exists $\lambda \geq 0$ such that for each $k \in \mathcal{X}$*

$$\begin{aligned} p_{k0}\frac{p_{kj}}{c_0} &\geq \lambda \quad \textit{for} \quad 1 \leq k \leq a_k^* \\ &\leq \lambda \quad \textit{for} \quad a_k^* < k < \infty \end{aligned}$$

The quantity $p_{k0}\frac{p_{kj}}{c_0}$ is the ratio of marginal increase in probability of detection to the marginal increase of cost, or the *marginal rate of return*. The variable λ is a Lagrange multiplier.

To the best of our knowledge, problems of multi-agent search with simple error models were first considered by Song and Teneketzis [Song and Teneketzis, 2004]. The problem is formulated as follows:

- M: number of agents.
- $\mathcal{X} = \{1, \ldots, K\}$: a discrete search space of K possible locations.
- $p_{k0}, k \in \mathcal{X}$: prior probability of location k containing the object.
- α_k: probability of detecting the object at location k if the object is there.
- N: searches are performed over N units of time. Each agent can search one location at each time. Each location can be searched by at most one agent at a given time.
- $\boldsymbol{a} = (a_1, \ldots, a_K)$: search plan where a_k is the number of searches performed at location k.
- $F(\boldsymbol{a})$: total probability of detection using plan $\boldsymbol{a}$.

If the object is at location k, the number of searches needed until detecting the object satisfies the geometric distribution. Thus, the probability of detecting the object at location k on the j-th search after failing the previous $j-1$ searches is (Lemma 2.1 of [Song and Teneketzis, 2004]):

$$p_{kj} = p_{k0}(1-\alpha_k)^{j-1}\alpha_k$$

The objective is to select search plan $\boldsymbol{a}$ to maximize the probability of detection $F(\boldsymbol{a})$ after N rounds of searches:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{maximize}} \;\; & F(\boldsymbol{a}) = \sum_{k=1}^{K} p_{k0} \sum_{j=1}^{a_k} p_{kj} \\ \text{subject to} \;\; & \sum_{k=1}^{K} a_k \leq MN \\ & a_k \in \{0, \ldots, N\}, \;\; \forall k \end{aligned}$$

From the formulation, it is natural to assume that $M < K$, i.e., the number of agents is less than the number of locations. The following result states that the optimal search plan can be obtained from the MN largest numbers out of all KN numbers p_{kj}.

Theorem 2.1.2 (Theorem 2.1 of [Song and Teneketzis, 2004])**.** *Let $\mathcal{L}$ denote the set of MN largest numbers p_{kj}, $k \in \mathcal{X}$, $j = 1, \ldots, N$. Then there exists a search plan $\boldsymbol{a}^* = (a_1^*, \ldots, a_K^*)$ such that*

$$a_k^* = \sum_{j=1}^{N} \mathbb{1}_{\mathcal{L}}(p_{kj}) \tag{2.1}$$

where

$$\mathbb{1}_{\mathcal{L}}(p_{kj}) = \begin{cases} 1, & \text{if } p_{kj} \in \mathcal{L} \\ 0, & \text{otherwise} \end{cases}$$

that is optimal.

To obtain the MN largest marginal probabilities p_{kj}, one needs to sort the KN numbers.

2.2 Search of a Moving Object with Simple Error Models

Since many results in classical search theory [Washburn et al., 2016] were obtained assuming continuous search effort, we will review results of continuous search effort. We will also review results of discrete search effort.

A typical problem of *single-agent search of a moving object with simple error models* in classical search theory [Washburn et al., 2016] is defined as follows:

- $\mathcal{X} = \{1, \ldots, K\}$: a discrete search space of K possible locations.
- T: finite time horizon.
- $\boldsymbol{x} = (x_1, \ldots, x_T) \in \mathcal{X}^T$: object trajectory. At time period t, the object is at location x_t.
- Ω: space of all possible object trajectories.
- $p(\boldsymbol{x})$: probability that the object is taking trajectory $\boldsymbol{x}$. It is independent of the search actions. $\sum_{\boldsymbol{x}\in\Omega} p(\boldsymbol{x}) = 1$.
- $\boldsymbol{a}_t = (a_{t,1}, \ldots, a_{t,K})$: search plan for time period t where $a_{t,k}$ is the amount of search effort allocated for location k at time period t.
- B_t: budget on search effort for time period t, i.e., $\sum_{k\in\mathcal{X}} a_{t,k} \leq B_t$.
- $\boldsymbol{a} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_T)$: search plan for all T time periods.

- $f(\alpha_{t,x_t} a_{t,x_t})$: detection function, i.e., the probability of detection at time period t with search plan $\boldsymbol{a}$ given the object trajectory $\boldsymbol{x}$. $\alpha_{t,k}$'s are known detection rates.

- $F(\boldsymbol{a}) = \sum_{\boldsymbol{x}\in\Omega} p(\boldsymbol{x})[1 - f(\sum_{s=1}^{T} \alpha_{s,x_s} a_{s,x_s})]$: total probability of non-detection after T time periods with search plan $\boldsymbol{a}$.

The objective is to find $\boldsymbol{a}$ to minimize the total probability of non-detection $F(\boldsymbol{a})$:

$$\underset{\boldsymbol{a}}{\text{minimize}} \;\; F(\boldsymbol{a}) = \sum_{\boldsymbol{x}\in\Omega} p(\boldsymbol{x})[1 - f(\sum_{s=1}^{T} \alpha_{s,x_s} a_{s,x_s})] \tag{2.2}$$

$$\text{subject to} \;\; \sum_{k=1}^{K} a_{t,k} \leq B_t, \quad \forall t \tag{2.3}$$

$$0 \leq a_{t,k} \leq B_t, \;\; \forall t, k \tag{2.4}$$

When the detection function $f(\zeta)$ is an exponential function $f(\zeta) = 1 - e^{-\zeta}$, Brown [Brown, 1980] established necessary and sufficient conditions of optimal search plans. Now $F(\boldsymbol{a})$ becomes

$$F(\boldsymbol{a}) = \sum_{\boldsymbol{x}\in\Omega} p(\boldsymbol{x}) e^{-\sum_{s=1}^{T} \alpha_{s,x_s} a_{s,x_s}}$$

Define $q(\boldsymbol{a}, t, k)$ as

$$q(\boldsymbol{a}, t, k) = \sum_{\boldsymbol{x}: x_t = k} p(\boldsymbol{x}) e^{-\sum_{s\neq t} \alpha_{s,x_s} a_{s,x_s}}$$

Note that $q(\boldsymbol{a}, t, \cdot)$ is proportional to the conditional probability distribution that the object is detected only at time period t but not at other times. And we have

$$F(\boldsymbol{a}) = \sum_{k=1}^{K} q(\boldsymbol{a}, t, k) e^{-\alpha_{t,k} a_{t,k}} \tag{2.5}$$

Without considering the time index t, the right hand side of (2.5) can be inter-

preted as a stationary search problem. $e^{-\alpha_{t,k}a_{t,k}}$ is the probability of detecting the object if a search effort of $a_{t,k}$ is allocated for location k, and $q(\boldsymbol{a}, t, k)$ is a defective "prior" probability that the object is at location k. The optimal solutions of the moving object search problem are closely related to the optimal solutions of these stationary search problems, as shown in the following result.

Theorem 2.2.1 (Proposition 4 of [Brown, 1980]; also Theorem 3.3 of [Washburn et al., 2016])**.** *A search plan $\boldsymbol{a}^*$ is optimal for problem (2.2)–(2.4) if and only if for all $t = 1, \ldots, T$, $\boldsymbol{a}_t^*$ is optimal for the problem below:*

$$\underset{\boldsymbol{a}_t}{\textit{minimize}} \quad \sum_{k=1}^{K} q(\boldsymbol{a}, t, k) e^{-\alpha_{t,k}a_{t,k}} \tag{2.6}$$

$$\textit{subject to} \quad \sum_{k=1}^{K} a_{t,k} \leq B_t \tag{2.7}$$

$$0 \leq a_{t,k} \leq B_t, \quad \forall k \tag{2.8}$$

When the search efforts are discrete, i.e., $a_{t,k} \in \{0, \ldots, B_t\}$ is the number of searches performed at location k at time period t, the conditions in Theorem 2.2.1 are necessary but not sufficient [Washburn, 1980].

Based on Theorem 2.2.1, Brown developed an algorithm for the case of continuous search effort and exponential detection function [Brown, 1980]. The algorithm is described briefly as follows:

- **Initialization:** Let $\boldsymbol{a} = 0$. Let $\epsilon > 0$ be a tolerance value. Let $F_{old} = 1$.
- **Step 1:** For $t = 1, \ldots, T$, find the optimal solution $\boldsymbol{a}_t'$ of the problem (2.6)–(2.8), and replace $\boldsymbol{a}_t$ in $\boldsymbol{a}$ with $\boldsymbol{a}_t'$.
- **Step 2:** Compute $F_{new} = F(T, \boldsymbol{a})$. If $|F_{new} - F_{old}| < \epsilon$, terminate the algorithm. Else, let $F_{old} = F_{new}$ and return to **Step 1** starting from $t = 1$.

Furthermore, when the object motion follows a Markov chain, Brown gave an

efficient implementation of his recursive algorithm [Brown, 1980]. Let P denote the transition probability matrix of the Markov chain whose (i, j)-th entry ρ_{ij} denotes the probability of the object moving from location i to location j in consecutive time periods. Let $\pi_{1,k}$ denote the prior probability of the object being at location k. Define

$$R(\boldsymbol{a}, t, k) = \sum_{\boldsymbol{x} \in \Omega: x_t = k} \pi_{1,x_1} \rho_{x_1,x_2} \cdots \rho_{x_{t-1},k} e^{-\sum_{s=1}^{t-1} \alpha_{s,x_s} a_{s,x_s}}$$

and

$$S(\boldsymbol{a}, t, k) = \sum_{\boldsymbol{x} \in \Omega: x_t = k} \rho_{k,x_{t+1}} \cdots \rho_{x_{T-1},x_T} e^{-\sum_{s=t+1}^{T} \alpha_{s,x_s} a_{s,x_s}}$$

Then,

$$q(\boldsymbol{a}, t, k) = R(\boldsymbol{a}, t, k) \cdot S(\boldsymbol{a}, t, k), \quad \forall \boldsymbol{a}, t, k \tag{2.9}$$

The efficient implementation of Brown's algorithm is:

- **Initialization:** Let $\boldsymbol{a} = 0$. Let $\epsilon > 0$ be a tolerance value. Let $F_{old} = 1$. Let $S(\boldsymbol{a}, t, k) = 1$, $\forall t, k$. Let $R(\boldsymbol{a}, 1, k) = \pi_{1,k}$, $\forall k$.

- **Step 1:** For $t = 1, \ldots, T$, compute $q(\boldsymbol{a}, t, k)$ using (2.9) for all k and find the optimal solution $\boldsymbol{a}'_t$ of the problem (2.6)–(2.8). Replace $\boldsymbol{a}_t$ in $\boldsymbol{a}$ with $\boldsymbol{a}'_t$. If $t < T$, compute

$$R(\boldsymbol{a}, t+1, k) = \sum_{j=1}^{K} R(\boldsymbol{a}, t, j) \rho_{jk} e^{-\alpha_{t,j} a_{t,j}}$$

- **Step 2:** Compute $F_{new} = F(T, \boldsymbol{a})$. If $|F_{new} - F_{old}| < \epsilon$, terminate the algorithm. Else, let $F_{old} = F_{new}$. Reset $R(\boldsymbol{a}, 1, k)$ to $\pi_{1,k}$ for all k. Let $S(\boldsymbol{a}, T, k) = 1$ and

starting from $t = T - 1$ to $t = 1$, compute

$$S(\boldsymbol{a}, t-1, k) = \sum_{j=1}^{K} \rho_{kj} e^{-\alpha_{t,j} a_{t,j}} S(\boldsymbol{a}, t, j)$$

Return to **Step 1**.

Washburn [Washburn, 1983] developed a Forward And Backward (FAB) algorithm which generalizes Brown's algorithm to more general objective functions than the probability of detection over a fixed time horizon, e.g., the expected time to detect. He had also used a version of the FAB algorithm to solve a problem where the object moves as a diffusion process approximated with 67 cells [Washburn, 1975].

Royset and Sato [Royset and Sato, 2010] studied problems of *multi-agent search of a moving object with simple error models* and developed cutting plane methods. They assumed that the agents have discrete search effort and motion constraints. They considered two cases: homogeneous agents and single object; heterogeneous agents and multiple objects. We will focus on reviewing the case of homogeneous agents and single object as the cutting plane methods developed for the case of heterogeneous agents and multiple objects are similar.

The problem for the case of homogeneous agents and single object is formulated as follows:

- $\mathcal{X} = \{1, \ldots, K\}$: a discrete search space of K possible locations.
- T: finite time horizon.
- $\boldsymbol{x} = (x_1, \ldots, x_T) \in \mathcal{X}^T$: object trajectory. At time period t, the object is at location x_t.
- Ω: space of all possible object trajectories.

- $p(\boldsymbol{x})$: probability that the object is taking trajectory $\boldsymbol{x}$. It is independent of the search actions. $\sum_{\boldsymbol{x}\in\Omega} p(\boldsymbol{x}) = 1$.
- M: number of agents. Each agent searches once at each time period.
- $b_{1,k}$: number of agents that are at location k at the initial time period 1.
- $b_{t,k,k'}$: the number of agents that search location k at time period t and will move to location k' next.
- $\mathcal{F}(k)$: the set of locations to which the agent can move from its current location k.
- $\mathcal{R}(k)$: the set of locations from which the agent can move to its current location k.
- $\Delta t_{k,k'}$: number of time periods for an agent to move directly from location k to location k' and search location k'.
- $\boldsymbol{a} = \{a_{t,k}, t = 1, \ldots, T, k \in \mathcal{X}\}$: search plan where $a_{t,k}$ is the number of agents searching location k at time period t.
- $f(\alpha_{t,x_t} a_{t,x_t}) = 1 - e^{-\alpha_{t,x_t} a_{t,x_t}}$: exponential detection function. The detection rates $\alpha_{t,k}$'s were constant in [Royset and Sato, 2010], but the cutting plane methods also work for non-constant $\alpha_{t,k}$'s.
- $F(\boldsymbol{a}) = \sum_{\boldsymbol{x}\in\Omega} p(\boldsymbol{x}) e^{-\sum_{t=1}^{T} \alpha_{t,x_t} a_{t,x_t}}$: total probability of non-detection after T time periods with search plan $\boldsymbol{a}$.

The objective is to find $\boldsymbol{a}$ to minimize the total probability of non-detection $F(\boldsymbol{a})$:

$$\underset{\boldsymbol{a}}{\text{minimize}} \;\; F(\boldsymbol{a}) = \sum_{\boldsymbol{x}\in\Omega} p(\boldsymbol{a}) e^{-\sum_{t=1}^{T} \alpha_{t,k} a_{t,k}} \tag{2.10}$$

$$\text{subject to} \quad \sum_{k' \in \mathcal{R}(k)} b_{t-\Delta t_{k',k},k',k} = \sum_{k' \in \mathcal{F}(k)} b_{t,k,k'}, \quad \forall t, k \tag{2.11}$$

$$\sum_{k' \in \mathcal{F}(k)} b_{1,k,k'} = b_{1,k}, \quad \forall k \tag{2.12}$$

$$\sum_{k' \in \mathcal{R}(k)} b_{t-\Delta t_{k',k},k',k} = a_{t,k} \quad \forall t, k \tag{2.13}$$

$$\sum_{k=1}^{K} a_{t,k} \leq M, \quad \forall t \tag{2.14}$$

$$b_{t,k,k'} \geq 0, \quad \forall t, k, k' \tag{2.15}$$

$$a_{t,k} \in \{0, \ldots, M\} \tag{2.16}$$

The cutting plane algorithm by Royset and Sato [Royset and Sato, 2010] is described briefly as follows:

- **Initialization**: Choose the final relative optimality tolerance $\delta > 0$ and the relative optimality tolerances $\delta^{(i)} > 0$, $i = 1, 2, \ldots$, at each iteration. Set lower bound $\underline{\xi}$ of the optimal value of the problem defined by (2.10)–(2.16) to be 0. Set upper bound $\bar{\xi}$ of the optimal value of the problem defined by (2.10)–(2.16) to be 1. Set $\boldsymbol{a}^{(0)} = 0$ and $i = 1$.

- **Step 1:** Solve the following mixed integer linear programming problem

$$\begin{aligned} &\underset{(\xi, \boldsymbol{a})}{\text{minimize}} \quad \xi \\ &\text{subject to} \quad F(\boldsymbol{a}^{(j)}) + \nabla F(\boldsymbol{a}^{(j)})'(\boldsymbol{a} - \boldsymbol{a}^{(j)}) \leq \xi, \quad \forall j = 0, \ldots, i-1 \\ &\qquad\qquad (2.11) - (2.16) \end{aligned} \tag{2.17}$$

  to near optimality. This gives us a feasible solution denoted by $(\xi^{(i)}, \boldsymbol{a}^{(i)})$ and a lower bound $\underline{\xi}^{(i)}$ such that $\xi^{(i)} - \underline{\xi}^{(i)} \leq \delta^{(i)} \underline{\xi}^{(i)}$ for the chosen relative optimality tolerance value $\delta^{(i)}$ at the i-th iteration.

- **Step 2:** If $\underline{\xi}^{(i)} > \underline{\xi}$, let $\underline{\xi} = \underline{\xi}^{(i)}$. If $F(\boldsymbol{a}^{(i)}) < \bar{\xi}$, let $\bar{\xi} = F(\boldsymbol{a}^{(i)})$. Then, if $\bar{\xi} - \underline{\xi} < \delta\underline{\xi}$, terminate the algorithm. Else, return to **Step 1**.

The cutting plane algorithm operates directly on an integer nonlinear program and can thus be easily generalized to apply to the case where there are no motion constraints, or the case of heterogeneous agents and multiple objects, etc. But it tends to converge slowly due to its lack of exploitation of the special structures in specific problems and the fact that an integer programming problem is solved at each iteration.

2.3 Search of Multiple Objects with Motion and Switching Cost Constraints

A typical orienteering problem is defined as follows:

- $G = (V, E)$: an undirected graph where the node set $V = \{v_1, \ldots, v_N\}$ represents the locations and the edge set $E = \{(v_i, v_j), 1 \leq i < j \leq N\}$ represents all the feasible direct movements between locations. Assume that symmetric relation holds, i.e., $(v_i, v_j) \in E$ if and only if $(v_j, v_i) \in E$.

- r_k: reward at node v_k which measures the information of location k containing an object of interest. The reward will be collected once the location is searched.

- c_{ij}: edge cost of (v_i, v_j) which is the switching cost between the two locations. Assume that $c_{ij} = c_{ji}$.

- M: number of agents.

- B: budget on the maximum cost that each agent can incur.

When $M = 1$, i.e., a single agent is used, we call it a *single orienteering problem* (SOP) [Tsiligirides, 1984, Golden et al., 1987]. When $M > 1$, i.e., multiple agents are used, the problem becomes a *team orienteering problem* (TOP) [Chao et al., 1996b].

The objective is to find closed paths, or *tours*, that start and end at v_1 to maximize the total rewards collected without exceeding the cost budgets. Since the reward will be collected once a location is searched, the tours of different agents do not overlap. Let binary variables $\beta_{km} \in \{0, 1\}$ denote if node v_k is visited by the m-th agent, and let binary variables $e_{ijm} \in \{0, 1\}$ denote if edge (v_i, v_j) is traveled by the m-th agent. An integer programming formulation of the orienteering problem is as follows:

$$
\begin{aligned}
\underset{\{\beta_{km}\}}{\text{maximize}} \quad & \sum_{m=1}^{M} \sum_{k=2}^{N} r_k \beta_{km} \\
\text{subject to} \quad & \sum_{m=1}^{M} \beta_{km} \leq 1, \ k = 2, \ldots, N && (2.18) \\
& \sum_{j=2}^{N} e_{1jm} = 2; \ \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} e_{ijm} \leq B, \ \forall m \\
& \sum_{i=1}^{k-1} e_{ikm} + \sum_{j=k+1}^{N} e_{kjm} = 2\beta_{km}, \ k = 2, \ldots, N, \ \forall m \\
& 2 \sum_{v_k \in S} \beta_{km} \leq |S| \Big(\sum_{v_i \in S, v_j \notin S} e_{ijm} + \sum_{v_i \notin S, v_j \in S} e_{ijm} \Big) && (2.19) \\
& \qquad\qquad S \subset V \setminus \{v_1\}, \ |S| \geq 2, \ \forall m \\
& e_{1jm} \in \{0, 1, 2\}, \quad j = 2, \ldots, N, \ \forall m \\
& e_{ijm} \in \{0, 1\}, \qquad 2 \leq i < j \leq N, \ \forall m \\
& \beta_{km} \in \{0, 1\}, \qquad k = 2, \ldots, N, \ \forall m
\end{aligned}
$$

where β_{km} decides whether agent m visits location k. Constraints (2.18) ensure that each node will be visited by at most one agent. There is also an exponential number of subtour elimination constraints (2.19).

However, exact solutions such as column generation, branch-and-bound, branch-and-cut and branch-and-price which are based on integer programming formulations are slow due to the inherent NP-hard complexity. Similarly, approximate algorithms with performance guarantees such as in [Blum et al., 2007] have theoretical importance but result in slow algorithms.

Heuristics exist for both SOP and TOP. For SOP, Tsiligirides [Tsiligirides, 1984] proposed a Monte Carlo based algorithm. The algorithm randomly generates many tours and picks the one with the highest total rewards. To generate one candidate tour, the algorithm uses a metric called *desirability*, $des(v_k)$, to pick the next node v_k to add to the tour:

$$des(v_k) = \big(\frac{r_k}{c(v_k, v_{last})}\big)^4$$

where $c(v_k, v_{last})$ is the edge cost from v_k to the node v_{last} that is most recently added to the tour. Then the algorithm randomly picks one of the four nodes with the highest $des(v_k)$ to add to the tour.

Golden et al. [Golden et al., 1987] proposed a *center-of-gravity* heuristic for SOPs defined in Euclidean space. It consists of three steps: tour construction, tour improvement, and center-of-gravity. The first two steps build an initial tour. In the third step, the algorithm computes the virtual center of gravity v_{cog} of the initial tour, as well as a metric, $cogr(v_k)$, for each node v_k:

$$cogr(v_k) = \frac{r_k}{c(v_k, v_{cog})}$$

where $c(v_k, v_{cog})$ is the Euclidean distance from v_k to the center of gravity v_{cog}. The nodes are ranked by their $cogr(v_k)$ and inserted to the tour with cheapest insertion without exceeding the budget. The three steps are repeated until two identical tours have been generated. Finally, the tour with the highest total rewards is selected from

all generated tours.

Chao et al. [Chao et al., 1996a] proposed a heuristic which restricts to considering only nodes within an ellipse whose two foci are the start and end nodes and the length of the major axis is equal to the budget B (will be a circle if the start and end nodes are identical). It creates and maintains many tours. At the following improvement steps, the algorithm exchanges nodes between or within tours to improve the total rewards of the current best tour.

Silberholz and Golden [Silberholz and Golden, 2010] proposed a *two-parameter iterative* heuristic for a generalized version of SOP. It can also be applied to standard SOP. The heuristic repeatedly runs a subroutine which initializes a tour and iteratively modifies the tour by removing nodes already in the tour and adding new nodes to the tour. One of the two user-defined parameters controls the number of iterations of modification in each run of the subroutine, and the other parameter controls the number of nodes to be removed at each iteration of modification.

For TOP, Chao et al. [Chao et al., 1996b] proposed a heuristic which is similar to their SOP heuristic in [Chao et al., 1996a]. Instead of selecting only one best tour, the TOP heuristic selects M best tours, where M is the number of search agents.

Other approaches based on meta-heuristics (tabu search, ant colony optimization, variable neighborhood search, etc.) have been proposed for both SOP and TOP. Interesting readers can refer to the survey paper [Vansteenwegen et al., 2011] for more information.

2.4 Adaptive Search with Complex Error Models

Adaptive search problems with complex error models are in general much harder than problems with simple error models. Existing solutions for many adaptive search problems typically require an error-free verification step to verify the authenticity of

a detection signal and rule out false objects [Stone, 1975, Kress et al., 2012].

In [Castañón, 1995], Castañón studied classes of adaptive search problems with complex error models in discrete search space and characterized optimal search strategies for these problems. The problem is formulated as follows:

- $\mathcal{X} = \{1, \ldots, K\}$: a discrete space of K possible locations.
- $X_k \in \{0, 1\}$: random variable which indicates whether location k contains an object of interest.
- $p_0(k)$: prior probability of location k containing an object of interest, i.e., $p_0(k) = Pr(X_k = 1)$.
- N: number of search stages. At each stage, the agent searches one location.
- a_n: location to be searched at stage n.
- Y_n: measurement at stage n. It satisfies

$$Pr(Y_n = y | a_n, X_{a_n}) = \begin{cases} f_1(y), & \text{if } X_{a_n} = 1 \\ f_0(y), & \text{else} \end{cases}$$

- $D_n = \{a_1, Y_1, \ldots, a_n, Y_n\}$: information available at stage n which consists of past actions and measurements.
- $p_n(k)$: posterior probability of location k containing an object of interest after n stages, i.e., $p_n(k) = Pr(X_k = 1 | D_n)$.
- $\boldsymbol{\gamma} = \{\gamma_n\}$: search strategy where $\gamma_n : p_{n-1} \to \mathcal{X}$.

Two assumptions on the objects were made in [Castañón, 1995]: (1) *Exclusive object assumption* where one and only one object is in $\mathcal{X}$; (2) *Independent objects*

assumption where the events that a location contains an object of interest are independent across locations. The objective is to find a search plan $\boldsymbol{\gamma}$ to maximize the final probability of correctly selecting a location which contains an object of interest, under either assumption:

$$\inf_{\boldsymbol{\gamma}} E_{\boldsymbol{\gamma}}[\max_{k=1,\ldots,K} p_N(k)]$$

Castañón [Castañón, 1995] showed that under either assumption, if the measurement error model of the agent satisfies a symmetry condition such that there exists a constant c that

$$f_1(y) = f_0(c - y), \quad \forall y \tag{2.20}$$

then optimal search strategies can be fully characterized by myopic strategies as follows:

Theorem 2.4.1 (Propositions 3 and 5 of [Castañón, 1995])**.** *Under either the exclusive object assumption or the independent objects assumption, if the measurement error model of the agent satisfies the symmetry condition in* (2.20)*, then the optimal search strategy is to search either of the two most likely locations given past information D_{n-1} at any stage n.*

For the special case where the measurements are binary, $f_1(y)$ can be characterized by the probability of detection α and $f_0(y)$ can be characterized by the probability of false alarm β. Castañón showed that under either object assumption, when $\beta = 0$, an optimal search strategy is to search the second most likely location given past information D_{n-1} at any stage n (Propositions 6 and 7 of [Castañón, 1995]). However, when $\alpha, \beta > 0$ but $\alpha \neq \beta$, i.e., both missed detections and false alarms exist but the symmetry condition does not hold, optimal strategies are no longer myopic.

Jedynak et al. [Jedynak et al., 2011] considered a problem of searching for a stationary object in a continuous search space using a single agent. The agent selects

a region to query, resulting in a noisy answer concerning the presence of the object in the query region. We refer to such agents as *Boolean agents*, as they are observing a noisy yes-no answer to their query. The objective is related to differential entropy. This problem is related to results in information theory such as the Rényi-Ulam game [Cover and Thomas, 2012], Horstein's [Horstein, 1963] probabilistic bisection scheme for sequential decoding, and Burnashev and Zigangirov's [Burnashev and Zigangirov, 1974] work on probabilistic search.

The problem in [Jedynak et al., 2011] is formulated as follows:

- $\mathcal{X} \subset \mathbb{R}^d$: a continuous space which is a compact subset of the Euclidean space.
- X: position of the stationary object.
- $p_0(x)$: prior probability distribution that is absolutely continuous with respect to Lebesgue measure.
- N: number of search stages.
- $A_n \subset \mathcal{X}$: search query at stage n.
- Y_n: measurement at stage n. It satisfies:

$$Pr(Y_n = y|A_n, X) \begin{cases} f_1(y), & \text{if } X \in A_n \\ f_0(y), & \text{if } X \notin A_n \end{cases}$$

  For simplicity, let Y_n take values from a discrete set.

- $D_n = \{A_1, Y_1, \ldots, A_n, Y_n\}$: information available at stage n which consists of past actions and measurements.
- $p_n(x)$: posterior probability of the object location X after n stages, i.e., $p_n(x) = p(x|D_n)$.

- $\boldsymbol{\gamma} = \{\gamma_n\}$: search strategy where $\gamma_n : p_{n-1} \to \mathcal{X}$.
- $H(p_n) = -\int_{\mathcal{X}} p_n(x) \log_2(p_n(x))dx$: posterior differential entropy after n stages.

The objective is to find search strategy $\boldsymbol{\gamma}$ to minimize $H(p_N)$ – the posterior differential entropy after N stages:

$$\inf_{\boldsymbol{\gamma}} E_{\boldsymbol{\gamma}}[H(p_N)|p_0]$$

To solve the problem, define the optimal value function $V(p_n, n)$ as:

$$V(p_n, n) = \inf_{A_{n+1}} E_{Y_{n+1}}[V(p_{n+1}, n+1)|A_{n+1}, p_n]$$

and define function $\varphi(u)$ as:

$$\varphi(u) = \mathcal{H}(uf_1 + (1-u)f_0) - u\mathcal{H}(f_1) - (1-u)\mathcal{H}(f_0)$$

where $\mathcal{H}(\cdot)$ is the standard Shannon entropy. Jedynak et al. [Jedynak et al., 2011] showed the following optimality result:

Theorem 2.4.2 (Theorem 2 and Proposition 1 of [Jedynak et al., 2011])**.** *A search strategy that selects A_n such that $\int_{A_n \subset \mathcal{X}} p_n(x)dx = u^* \in \arg\max_u \varphi(u)$ is optimal. Also, the optimal value function at any stage n is given by*

$$V(p_n, n) = H(p_n) - (N-n)\varphi(u^*)$$

The work in [Jedynak et al., 2011] has been generalized in several directions. Sznitman et al. [Sznitman et al., 2013] considered the case where the agent is allowed to choose a precision mode l_n with a cost $W(l_n)$ at each stage n. Different precision modes will trigger different measurement error models. The objective is to find a sequence of (A_n, l_n), $n = 1, \ldots, N$, to minimize the sum of the final-stage expected posterior differential entropy $H(p_N)$ and the discounted total precision mode costs:

$$\inf_{(A_n,l_n),n=1,\ldots,N} E[H(p_N) + \lambda \sum_{n=1}^{N} W(l_n)|p_0]$$

They showed that the myopic strategies are optimal and the optimal value functions have closed-form expression.

Rajan et al. [Rajan et al., 2015] generalized [Jedynak et al., 2011] to consider searching for multiple objects using a single agent. There are J objects whose positions are denoted by $X_1, \ldots, X_J$. At each stage, the agent queries A_n and receives a random measurement Y_n such that

$$Pr(Y_n|A_n, X_1, \ldots, X_J) = Pr(Y_n|\mathbb{1}_{\{X_1 \in A_n\}} + \cdots + \mathbb{1}_{\{X_J \in A_n\}})$$

That is, the measurement Y_n is a random function of the number of objects being queried by A_n. Similar as in [Jedynak et al., 2011], the objective is to minimize the final-stage expected posterior differential entropy. They derived a lower bound for the optimal value of the problem, and provided a search strategy with approximation guarantee.

Tsiligkaridis et al. [Tsiligkaridis et al., 2014] generalized [Jedynak et al., 2011] to the problem of searching for a single object using multiple Boolean agents with independent errors, and developed characterizations of optimal strategies. They focused on the case where the agents have binary symmetric error models, and established the equivalence of simultaneous query strategies and sequential query strategies for the case.

Chapter 3

Multi-Agent Sparse Search of a Stationary Object with Simple Error Models

In this chapter, we generalize the problem in [Song and Teneketzis, 2004] to the case where each agent can only search in part of the search space. We refer to this problem as the sparse multi-agent discrete search problem. For the special case where the agents are homogeneous and thus the probability of detection depends only on the location being searched but not on the agent, we provide a novel perspective of viewing the problem as a minimum cost network flow problem, inspired by [Castañón, 1987]. We develop a fast specialized algorithm for solving the problem based on the min-cost flow perspective. We show that the algorithm always terminates in finite time, and analyze the time complexity of the algorithm. We prove that the algorithm yields an optimal agent allocation. We perform experiments to show that our specialized algorithm is faster than general min-cost flow algorithms.

We also consider the general case where the agents are heterogeneous and thus the probability of detection depends on both the agent performing the search and the location being searched. This problem has been studied previously and found to be NP-hard [Lloyd and Witsenhausen, 1986, Ahuja et al., 2007]. In [Ahuja et al., 2007], several approximate algorithms are considered, including branch-and-bound and local search. We show that this problem can be posed as a submodular maximization problem over a matroid, and establish that the greedy algorithm is guaranteed to perform within a factor of $\frac{1}{2}$ of the optimal value. We also develop a block coordinate

ascent algorithm, and provide an upper bound of the optimal objective value (total probability of detection).

The chapter is organized as follows: In Section 3.1, we formulate the sparse multi-agent discrete search problem. In Section 3.2, for the special case of homogeneous agents, we introduce the min-cost flow perspective of viewing the problem. We provide the primal and dual linear programming formulations of the problem, and derive conditions in which an agent allocation is optimal. We propose and analyze a fast algorithm for solving the problem in Section 3.3, and prove that it yields an optimal solution. Section 3.4 discusses the general case of heterogeneous agents where the probability of detection depends on both the location and the agent. Section 3.5 contains the experiment results.

3.1 Problem Formulation

Consider M agents that search for a stationary object hidden in one of K discrete locations. Each agent has limited search accessibility to only a subset of locations. Define an *accessibility pair* of agent m and location k if and only if agent m has accessibility to location k, denoted by (m, k). Let $\mathcal{A}$ denote the set of all accessibility pairs. Without loss of generality, assume that each location can be accessed by at least one agent and each agent can access at least one location. Assume that a search of any location by any agent that can access it costs one unit of search effort. Let the budget of total effort agent m can allocate be $N_m \in \mathcal{Z}^+$, the set of positive integers. Let $N = \sum_{m=1}^{M} N_m$ be the total search effort available.

Denote the prior probability of location k containing the object as p_{k0}. If the object is in location k, a search of the location with agent m will find the object with probability α_{mk}. If the object is not in location k, searching it with any agent will always yield "no detection". Assume that agent observations are conditionally

independent across locations and of previous searches.

Our objective is to allocate agent effort among accessible locations so as to maximize the probability of finding the object after spending all budgets. Suppose that we allocate x_{mk} units of effort to search location k using agent m. Let $\boldsymbol{x}_k = (x_{1k}, \ldots, x_{Mk})$. Due to the independence assumption on agent observations, the probability of finding the object at location k under allocation $\boldsymbol{x}_k$ is

$$P_k(\boldsymbol{x}_k) = p_{k0}(1 - \prod_{m=1}^{M}(1-\alpha_{mk})^{x_{mk}}), \quad k = 1, \ldots, K$$

The problem is to find an agent allocation $\boldsymbol{x} = \{x_{mk}, \forall m, k\}$ to maximize the probability of detecting the object:

$$\underset{\boldsymbol{x}}{\text{maximize}} \ \sum_{k=1}^{K} p_{k0}\left(1 - \prod_{m=1}^{M}(1-\alpha_{mk})^{x_{mk}}\right) \tag{3.1}$$

$$\text{subject to} \sum_{k:(m,k)\in\mathcal{A}} x_{mk} = N_m, \ \forall m \tag{3.2}$$

$$x_{mk} \in \{0, 1, \ldots, N_m\}, \ \forall (m,k) \in \mathcal{A} \tag{3.3}$$

$$x_{mk} = 0, \ \forall (m,k) \notin \mathcal{A} \tag{3.4}$$

Note that the equality constraint for the agent search budgets can be included because of the monotonicity of the objective function.

3.2 A Min-Cost Flow Interpretation for Homogeneous Agents

3.2.1 Problem Under Homogeneous Agent Assumption

When the agents have homogeneous error models, the probabilty of detecting the object at location k if the object is at the location is α_k, independent of the agent. Let $u_k = \sum_{m=1}^{M} x_{mk}$ denote the number of searches for location k, then the probability of finding the object after u_k searches of location k is $p_{k0}(1-(1-\alpha_k)^{u_k})$. The problem

becomes:

$$\underset{\boldsymbol{x}}{\text{maximize}} \sum_{k=1}^{K} p_{k0}(1-(1-\alpha_k)^{u_k}) \tag{3.5}$$

$$\text{subject to} \sum_{m:(m,k)\in\mathcal{A}} x_{mk} = u_k,\ \forall k$$

$$\sum_{k:(m,k)\in\mathcal{A}} x_{mk} = N_m,\ \forall m$$

$$x_{mk} \in \{0,1,\ldots,N_m\},\ \forall (m,k) \in \mathcal{A}$$

A useful quantity is p_{kj}, the probability that $j-1$ searches of location k have not found the object while the j-th search of location k finds the object. Using the conditional independence of search outcomes, this expression becomes

$$p_{kj} = p_{k0}(1-\alpha_k)^{j-1}\alpha_k$$

The problem has a separable nonlinear objective function (3.5) and linear constraints. Each of the individual functions in (3.5) is concave when the variables u_k are relaxed to be continuous. By introducing a few additional variables, we can transform (3.5) into linear form, as each of the individual functions in (3.5) can be decomposed as

$$p_{k0}(1-(1-\alpha_k)^{u_k}) = \underset{\{y_{kj}\}}{\text{maximize}} \sum_{j=1}^{N} p_{kj}y_{kj} \tag{3.6}$$

$$\text{subject to} \sum_{j=1}^{N} y_{kj} = u_k \tag{3.7}$$

$$y_{kj} \in \{0,1\},\ \forall j \tag{3.8}$$

3.2.2 A Min-Cost Flow Interpretation

We next map the above problem to a minimum cost network flow problem. By using the transformation in (3.6)–(3.8), we obtain the following integer programming problem:

$$\begin{aligned}
\underset{\boldsymbol{x},\boldsymbol{y}}{\text{minimize}} \quad & \sum_{k=1}^{K}\sum_{j=1}^{N} -p_{kj}y_{kj} \\
\text{subject to} \quad & \sum_{k:(m,k)\in\mathcal{A}} x_{mk} = N_m,\ m = 1,\ldots,M && (3.9)\\
& \sum_{m:(m,k)\in\mathcal{A}} x_{mk} = \sum_{j=1}^{N} y_{kj},\ k = 1,\ldots,K && (3.10)\\
& \sum_{k=1}^{K}\sum_{j=1}^{N} y_{kj} = N && (3.11)\\
& x_{mk} \in \{0,1,\ldots,N_m\},\ \forall (m,k) \in \mathcal{A} \\
& y_{kj} \in \{0,1\},\ \forall k,j
\end{aligned}$$

A graphical representation of the constraints of this optimization problem is depicted in the directed multigraph in Figure 3·1. Source node s_m with supply N_m represents the m-th search agent. Sink node t_k represents the k-th location. If agent m can access location k, there is a directed arc from source node s_m to sink node t_k.

Let $x_{mk} \in \{0,1,\ldots,N_m\}$ denotes the flow from s_m to t_k. There is also a dummy global sink node. The cost on each arc from source s_m to sink t_k is zero. From t_k to the global sink node, there are N directed arcs, with cost $-p_{kj}$ on the j-th arc. Denote the flow on the j-th arc from t_k to the global sink node by $y_{kj} \in \{0,1\}$.

Equations (3.9), (3.10) and (3.11) correspond to flow conservation at the source nodes s_m's, the sink nodes t_k's and the global sink node, respectively. This integer linear programming problem is equivalent to the transformed problem presented in the previous section.

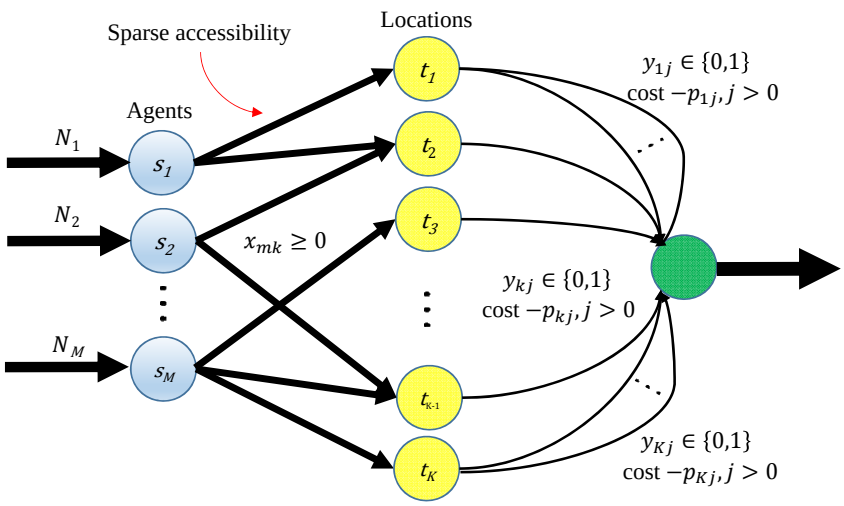


Figure 3·1: Sparse multi-agent discrete search problem viewed as a min-cost network flow problem.

From Fig. 3·1, it is clear that the constraints in this integer program are unimodular, and the right-hand sides of the constraints are integers (the search efforts.) Hence, the optimal solutions of the linear programming relaxation are integer-valued. Thus, we can obtain optimal solutions for the above problem using any network optimization package. However, doing so requires explicit construction of the network in Fig. 3·1, which requires creating a large number of additional arcs in the network. In our development below, we exploit the special structure of the network problem to develop a fast algorithm that avoids this extra construction.

3.2.3 Duality and Complementary Slackness

The Lagrangian of the primal is

$$
\begin{aligned}
& L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d^s}, \boldsymbol{d^t}, \lambda) \\
=& \sum_{k=1}^{K} \sum_{j=1}^{N} -p_{kj} y_{kj} + \sum_{m=1}^{M} d_m^s (N_m - \sum_{k:(m,k)\in\mathcal{A}} x_{mk}) \\
&+ \sum_{k=1}^{K} d_k^t (\sum_{m:(m,k)\in\mathcal{A}} x_{mk} - \sum_{j=1}^{N} y_{kj}) + \lambda (\sum_{k=1}^{K} \sum_{j=1}^{N} y_{kj} - N)
\end{aligned}
$$

Here we let d_m^s, d_k^t, and λ be the dual variables, a.k.a. the *prices*, of source node s_m, sink node t_k and the global sink node, respectively. The dual problem is then:

$$\begin{aligned} &\underset{\boldsymbol{d^s},\boldsymbol{d^t},\lambda}{\text{maximize}} \ \sum_{k=1}^{K}\sum_{j=1}^{N}\min\{0,\lambda-d_k^t-p_{kj}\}+\sum_{m=1}^{M}d_m^s N_m-\lambda N \\ &\text{subject to } \ d_m^s \le d_k^t,\ \forall(m,k)\in\mathcal{A} \end{aligned} \tag{3.12}$$

The complementary slackness conditions (see Theorem 9.4 of [Ahuja et al., 1993]) are

$$d_m^s < d_k^t \Rightarrow x_{mk}=0, \forall(m,k)\in\mathcal{A} \tag{3.13}$$

$$x_{mk}>0 \Rightarrow d_m^s = d_k^t, \forall(m,k)\in\mathcal{A} \tag{3.14}$$

$$d_k^t-\lambda < -p_{kj} \Rightarrow y_{kj}=0,\ \forall k,j \tag{3.15}$$

$$0<y_{kj}<1 \Rightarrow d_k^t-\lambda=-p_{kj},\ \forall k,j \tag{3.16}$$

$$d_k^t-\lambda > -p_{kj} \Rightarrow y_{kj}=1,\ \forall k,j \tag{3.17}$$

By Theorem 9.4 of [Ahuja et al., 1993], if a primal feasible solution $\{\boldsymbol{x},\boldsymbol{y}\}$ and a dual feasible solution $\{\boldsymbol{d^s},\boldsymbol{d^t},\boldsymbol{\lambda}\}$ satisfy the complementary slackness conditions (3.13)-(3.17), then $\{\boldsymbol{x},\boldsymbol{y}\}$ is also an optimal solution for the primal problem.

3.3 Fast Algorithm for Homogeneous Agent Search

Here we propose a specialized algorithm for the problem of sparse discrete search of a stationary object using multiple homogeneous agents without false alarm errors. Our algorithm is based on primal-dual techniques: it begins with a solution satisfying complementary slackness. It continually increases the flow and modifies prices until primal feasibility is achieved, while maintaining complementary slackness at every iteration. For the purpose of exposition, we show how to track the prices even though our algorithm does not require the prices to be tracked explicitly.

Algorithm

Input: $\mathcal{A}$, N_m, p_{k0}, α_k, $\forall m, k$.

Output: $x_{mk}, \forall (m,k) \in \mathcal{A}$.

Initialization:

Let $x_{mk} = 0$ and $y_{kj} = 0$, $\forall m, k, j$. Compute and insert p_{k1}, $\forall k$ into a max-oriented binary heap H of size K. Define the available supply at each source node s_m as R_m. Let $R_m = N_m$, $\forall m$. Initialize the prices of the sources nodes to be $d_m^s = 0$, $\forall m$. Initialize the prices of the sink nodes to be $d_k^t = 0$, $\forall k$. Initialize the price of the global sink node to be $\lambda = \max_k p_{k1}$. Mark all the sources and sinks as not eliminated.

Step 1: Repeat until $R_m = 0$ for all m:

Extract the maximum $p_{k^*j^*}$ from H. Lower the price of the global sink to $\lambda = p_{k^*j^*}$. If there are eliminated sources and sinks, lower the price of each such source or sink

$$d_m^s = p_{k^*j^*} - v_m^s$$

$$d_k^t = p_{k^*j^*} - v_k^t$$

where v_m^s, v_k^t are the elimination values for source s_m and sink t_k, respectively.

a) If the sink node t_{k^*} is marked as eliminated, continue **Step 1**.

b) Run `Assign-Extra-Demand(`t_{k^*}`)` to find an augmenting path from the sink node t_{k^*} to one of the source nodes with available supply and let its returned source index be m^*.

c) If $m^* > 0$, decrement $R_{m^*} \leftarrow R_{m^*} - 1$. Let $y_{k^*j^*} = 1$. Insert p_{k^*,j^*+1} into H.

d) Else, identify s_m's and t_k's marked as visited in `Assign-Extra-Demand`(t_{k^*}) as an *isolated group*. Mark them as eliminated, and assign their elimination value as $p_{k^*j^*}$ to all such sources and sinks.

e) Return to **Step 1**.

Step 2: When $R_m = 0$ for all m, terminate the algorithm.

In the algorithm, we use subroutine `Assign-Extra-Demand` to search for an augmenting path from a sink node t_{k^*} that requires assignment of an extra unit of demand to a source node that has available supply. It visits a group of source and sink nodes whose prices will be updated implicitly.

The subroutine is described as follows:

`Assign-Extra-Demand`(t_{k^*})

Initialization:

Begin with a queue Q and a set of visited nodes V, both containing the single node t_{k^*}. Define available supply at each source s_m as R_m. Initialize an array of predecessors for all sources and sinks as $pred(s_m) = -1$, $pred(t_k) = -1$.

Step 1: Remove the top element in Q:

a) If the element is a sink t_k, check each source s_m such that: (1) s_m is not eliminated; (2) $(m, k) \in \mathcal{A}$; (3) $s_m \notin Q$; (4) $s_m \notin V$. Add each such source s_m to V. If one source s_m has $R_m > 0$, an augmenting path has been found; set $t_k = pred(s_m)$ and go to **Step 3** below. Else, add each such s_m to Q and set $pred(s_m) = t_k$.

b) If the element is a source s_m and $R_m = 0$, find each sink t_k such that: (1) t_k is not eliminated; (2) $(m,k) \in \mathcal{A}$; (3) $x_{mk} > 0$; (4) $t_k \notin Q$; (5) $t_k \notin V$. Add each such sink t_k to Q and V, and set $pred(t_k) = s_m$.

Step 2: If Q is not empty, repeat **Step 1**. Else, terminate the subroutine and return -1.

Step 3: An augmenting path has been found. Set $m^* = m$. Then, recursively, starting with $s = s_m$, perform an augmentation as follows:

i. For source s_m, find $t_k = pred(s_m)$. Increment $x_{mk} \leftarrow x_{mk} + 1$.

ii. For sink t_k, if $pred(t_k) = -1$, finish. Else, let $s_m = pred(t_k)$ and decrement $x_{mk} \leftarrow x_{mk} - 1$.

Terminate the subroutine and return m^*, the index of the source node providing the supply for the extra unit of demand at sink t_{k^*}.

If `Assign-Extra-Demand` fails to assign the extra unit of demand to any supply source, we identify all source and sink nodes marked as visited during the search as an isolated group, and remove them from further consideration in the subsequent assignment problems by marking their status as eliminated.

Our algorithm has two notable features. First, it exploits the implicit ordering of p_{kj}'s for the same location and computes p_{kj}'s on the fly only when they are needed. This is assisted by maintaining a max-oriented heap. Second, it uses successive shortest path backtracking for feasible flow augmentations, and updates prices implicitly such that dual feasibility and complementary slackness are maintained. At each iteration an element is popped out of the heap, creating one unit of demand at a sink. The subroutine `Assign-Extra-Demand` then tries to assign this extra unit of demand to a

source with available supply. It marks a group of source and sink nodes as visited, and raise the prices of these nodes while maintaining dual feasibility and complementary slackness. When `Assign-Extra-Demand` fails to find an augmenting path, we identify an isolated group, which no longer need to be considered in the algorithm for augmentation.

We now show that the algorithm maintains dual feasibility and complementary slackness.

Theorem 3.3.1. *Assume that the prices* $\{\boldsymbol{d^s}, \boldsymbol{d^t}, \lambda\}$ *are updated as described in the algorithm. Then, the prices* $\{\boldsymbol{d^s}, \boldsymbol{d^t}, \lambda\}$ *satisfy the dual feasibility condition* (3.12) *at each iteration. Also, together with the primal variables* $\{\boldsymbol{x}, \boldsymbol{y}\}$*, they satisfy the complementary conditions* (3.13)-(3.17) *at each iteration.*

Proof. The dual constraints are (3.12) (restated here):

$$d_m^s \leq d_k^t, \ \forall (m,k) \in \mathcal{A} \tag{3.12}$$

and the complementary slackness conditions (3.13)-(3.17) (restated here):

$$d_m^s < d_k^t \Rightarrow x_{mk} = 0, \forall (m,k) \in \mathcal{A} \tag{3.13}$$

$$x_{mk} > 0 \Rightarrow d_m^s = d_k^t, \forall (m,k) \in \mathcal{A} \tag{3.14}$$

$$d_k^t - \lambda < -p_{kj} \Rightarrow y_{kj} = 0, \ \forall k, j \tag{3.15}$$

$$0 < y_{kj} < 1 \Rightarrow d_k^t - \lambda = -p_{kj}, \ \forall k, j \tag{3.16}$$

$$d_k^t - \lambda > -p_{kj} \Rightarrow y_{kj} = 1, \ \forall k, j \tag{3.17}$$

Initially,

$$\begin{aligned} x_{mk} &= 0, \ \ \forall m, k \\ y_{kj} &= 0, \ \ \forall k, j \\ d_m^s &= 0, \ \ \forall m \\ d_t^k &= 0, \ \ \forall k \\ \lambda &= \max_k p_{k1} \end{aligned}$$

Since $d_m^s = d_k^t = 0$ for all $(m,k) \in \mathcal{A}$, the prices are dual feasible, and the

assignments $x_{mk} = 0$ satisfy complementary slackness. Note also that, by choice of λ, we have $d_k^t - \lambda \leq -p_{kj}$ for all k, j, so the assignments $y_{kj} = 0$ also satisfy complementary slackness.

Now, consider any algorithm iteration before any nodes are marked as eliminated. Let $p_{k^*j^*}$ denote the value extracted from the heap H. This iteration will lower λ to $p_{k^*j^*}$. It will also increase $y_{k^*j^*}$ to 1, and modify several x_{mk} to be non-zero or zero. Note that $d_m^s = d_k^t$ for all $(m, k) \in \mathcal{A}$ still, so the prices are dual feasible, and any assignments $x_{mk} \geq 0$ satisfy complementary slackness. Thus, the main item to verify is that the previous assignments $y_{kj} = 1$ still satisfy complementary slackness with the new price λ, the new assignment $y_{k^*j^*} = 1$ satisfies complementary slackness, and the previous assignments $y_{kj} = 0, k \neq k^*, j \neq j^*$ satisfy complementary slackness.

Since now $\lambda = p_{k^*j^*}$, we have $d_{k^*}^t - \lambda = -p_{k^*,j^*}$, so the new assignment $y_{k^*j^*} = 1$ satisfies complementary slackness. Furthermore, for any $y_{kj} = 1$ prior to the iteration, since λ was not increased in the iteration, we still have $d_k^t - \lambda \geq -p_{kj}$; thus they satisfy complementary slackness. If $y_{kj} = 0$ prior to the iteration, we have $-\lambda \leq -p_{kj}$ at the beginning of the iteration. Since $p_{k^*j^*}$ is the largest value in the heap H, the new value of λ will still satisfy $-\lambda \leq -p_{kj}$, $y_{jk} = 0$ after the iteration will still satisfy complementary slackness.

Now consider the output of an iteration where eliminated nodes are present. In such an iteration, none of the arc flows involving eliminated nodes change, as the augmentation path only searches for sources and sinks that are not eliminated. However, the prices of eliminated nodes change, as does the price of the global sink, so we must guarantee that these price changes maintain dual feasibility and complementary slackness.

The only way to violate dual feasibility is for the price of a sink t_k that is eliminated to drop below the price of a source s_m, where $(m, k) \in \mathcal{A}$. If a sink t_k is eliminated, any source s_m such that $(m, k) \in \mathcal{A}$ is also eliminated in the same isolated group, because it will be visited by `Assign-Extra-Demand`. Hence, the elimination values will satisfy $v_m^s = v_k^t$, and the resulting prices will satisfy $d_m^s = d_k^t$, and hence will not violate dual feasibility.

Let's show that modifying the prices maintains complementary slackness for all flows $x_{mk} > 0$ at the end of the iteration. As discussed earlier, if a sink t_k is eliminated, then any source s_m such that $(m, k) \in \mathcal{A}$ is also eliminated and has the same elimination value as t_k. As such, the new source and sink prices will be equal after modification, and the original flows x_{mk} will satisfy complementary slackness. If a

sink t_k is not eliminated, any of the flows modified by the augmenting path will involve only sources and sinks that are not eliminated and thus their prices start and remain at value 0. Hence, all the flows x_{mk} resulting from the augmentation will satisfy complementary slackness.

The final step is to establish that the flows y_{kj} satisfy complementary slackness with the new prices d_k^t and λ. For any t_k that is not eliminated, its price d_k^t remains zero, and all flows y_{kj} will satisfy complementary slackness by the same argument as above, because p_{k^*,j^*} is the largest value in the heap H. Hence, consider a node t_k that has been eliminated. If $y_{kj} = 1$, the elimination value $v_k^t \leq p_{kj}$, since it was set in an iteration when no augmentation was possible, and thus after $y_{kj} > 0$. Then, $d_k^t = p_{k^*,j^*} - v_k^t$, and $\lambda = p_{k^*,j^*}$. Thus, $d_k^t - \lambda = -v_k^t \geq -p_{kj}$ and $y_{kj} = 1$ satisfies complementary slackness. If $y_{kj} = 0$, the elimination value $v_k^t \geq p_{kj}$ because of the heap property. Then, $d_k^t - \lambda = -v_k^t \leq -p_{kj}$ and $y_{kj} = 0$ satisfies complementary slackness.

The proof is now complete. ■

The advantage of our algorithm is that it maintains a set of prices for sources and sinks so that all the arcs between non-eliminated sources and non-eliminated sinks have zero reduced costs. As such, finding an augmenting path can be done by breadth-first search, instead of a more complex shortest path algorithm with reduced costs. Furthermore, since the only price changes are for nodes that are eliminated (and the global sink), the entire algorithm can be executed without updating any dual prices, leading to efficient implementation.

Next, we show that the algorithm always terminates with finite time complexity, and analyze its time complexity.

Theorem 3.3.2. *The algorithm terminates in finite time, with complexity is $O(N|\mathcal{A}|)$, where $N = \sum_{m=1}^{M} N_m$ is the total supply from all source nodes, and $|\mathcal{A}|$ is the cardinality of the set of all accessibility pairs.*

Proof. We first show finite time termination. Each iteration of the algorithm either successfully assigns one unit of demand to some supply source, or identifies and eliminates an isolated group which consists of at least one source node and one sink node. Given the connectivity assumptions, there is a feasible assignment of all source

supply to sink nodes. Since we have a finite number of source and sink nodes, and a finite integer number of supply, the algorithm will eventually terminate. Upon the termination of the algorithm, due to the assumption that each location is accessible by at least one agent and each agent can access at least one location, there will be no remaining unassigned supply.

In terms of computation, the overhead for constructing the binary heap is $O(K)$. The worst-case running time for heap insertion and deletion is $O(\log(K))$. The worst-case running time for finding an augmenting path is $O(|\mathcal{A}|)$ since we might have to explore every arc between the source and sink nodes. At each iteration of the algorithm, the algorithm may successfully assign one unit of supply and do heap insertion and deletion, or fail to find an augmenting path. There are $N = \sum_{m=1}^{M} N_m$ units of supply to assign, and there are at most $\min(M, K)$ failures since each failure eliminates at least one source and one sink. Thus, the overall running time is $O\big(K + N(\log(K) + |\mathcal{A}|) + \min(M, K)|\mathcal{A}|\big) = O\big(N|\mathcal{A}|\big)$ since $\max(M, K) \leq |\mathcal{A}|$ and $\min(M, K) \leq M \leq N$. ■

In Table 3.1, we provide the time complexity of several general min-cost flow algorithms. It can be seen that our specialized algorithm for solving the integer subproblems has a significantly smaller time complexity compared to general min-cost flow algorithms.

Our algorithm also works for the problem considered in [Song and Teneketzis, 2004], in which every agent can access all the locations. Assuming that there are M agents, the amortized complexity of finding an augmenting path is $O(1)$, so the complexity of our algorithm becomes $O(N \log(K))$. In contrast, the algorithm of [Song and Teneketzis, 2004] requires $O(KN/M \log(KN/M))$, with the assumption that $K > M$, which is slower than our algorithm.

The algorithm constructs an optimal allocation in that the dual prices assigned by the algorithm are feasible and satisfy complementary slackness along with the primal feasible allocation constructed by the algorithm.

Theorem 3.3.3. *The algorithm constructs an optimal multi-agent search allocation.*

Table 3.1: Time complexity of general min-cost flow algorithms when applied on the integer subproblems ([Ahuja et al., 1993, Kelly and O'Neill, 1991]). M is the number of agents, K is the number of locations, and $|\mathcal{A}|$ is the cardinality of the set of all accessibility pairs. Let $N_n = M + K$ be the number of nodes, $N_a = |\mathcal{A}| + NK$ be the number of arcs, $U = M$ be the largest supply/demand or arc capacity, and $C = \max_{1 \le k \le K}\{-q_{k,1}\}$ be the largest cost coefficient.

Capacity scaling algorithm [Ahuja et al., 1993]	$O\Big(N_a N_n \big(N_a + N_n \log(N_n)\big)\Big)$
Cost scaling algorithm [Ahuja et al., 1993]	$O\Big(N_n^2 N_a \log\big(N_n C\big)\Big)$
Double scaling algorithm [Ahuja et al., 1993]	$O\Big(N_n N_a \log(U) \log\big(N_n C\big)\Big)$
Minimum mean cycle-canceling algorithm [Ahuja et al., 1993]	$O\Big(N_n^2 N_a^3 \log(N_n)\Big)$
Repeated capacity scaling algorithm [Ahuja et al., 1993]	$O\Big(N_a^2 \log(N_n)(N_a + N_n \log(N_n))\Big)$
Enhanced capacity scaling algorithm [Ahuja et al., 1993]	$O\Big(N_a \log(N_n)(N_a + N_n \log(N_n))\Big)$
Network simplex algorithm [Kelly and O'Neill, 1991]	$O\Big((N_n + N_a) N_n N_a C^2 U\Big)$

Proof. The algorithm constructs a primal feasible solution $\{\boldsymbol{x}, \boldsymbol{y}\}$ which satisfies all the constraints of the primal problem. Also, if we track prices $\{\boldsymbol{d^s}, \boldsymbol{d^t}, \lambda\}$ as described in the algorithm, the dual feasibility and complementary slackness are maintained throughout the algorithm by Theorem 3.3.1. Therefore, the solution $\{\boldsymbol{x}, \boldsymbol{y}\}$ is also primal optimal. ■

3.4 Heterogeneous Agent Search

3.4.1 Set Based Formulation

When the agents are heterogeneous, the probabilities of detection depend on both agents and locations. Recall that the detection probability at location k using agent m is α_{mk} and the problem is to find an agent allocation $\boldsymbol{x} = \{x_{mk}, \forall m, k\}$ to maximize the probability of detecting the object:

$$\underset{\boldsymbol{x}}{\text{maximize}} \ \sum_{k=1}^{K} p_{k0} \left(1 - \prod_{m=1}^{M} (1 - \alpha_{mk})^{x_{mk}}\right) = 1 - \sum_{k=1}^{K} p_{k0} \prod_{m=1}^{M} (1 - \alpha_{mk})^{x_{mk}} \qquad (3.1)$$

$$\text{subject to} \sum_{k:(m,k)\in\mathcal{A}} x_{mk} \leq N_m, \ \forall m \tag{3.2'}$$

$$x_{mk} \in \{0, 1, \ldots, N_m\}, \ \forall (m,k) \in \mathcal{A} \tag{3.3}$$

$$x_{mk} = 0, \ \forall (m,k) \notin \mathcal{A} \tag{3.4}$$

Note that due to the monotonicity of the objective function, the inequality constraint (3.2′) is equivalent to the equality constraint (3.2).

This problem is known to be NP-hard [Lloyd and Witsenhausen, 1986,Ahuja et al., 2007]. Approximation schemes such as branch-and-bound and local search have been proposed for these problems [Ahuja et al., 2007].

An alternative formulation of the problem is based on a monotone submodular objective function and matroid constraints. Assume that we split agent m into N_m micro agents, where N_m is the budget of the original agent m. Denote the j-th micro agent of the original agent m by $\langle m, j\rangle$. Recall that $N = \sum_{m=1}^{M} N_m$ is the total budget of all original agents. So we have an equivalent problem with N micro agents. Each micro agent $\langle m, j\rangle$ $(j = 1, \ldots, N_m)$ has budget $N_{\langle m,j\rangle} = 1$.

Let $\mathcal{E}$ be the set of all accessibility pairs after the split. Denote by the subset of $\mathcal{E}$ as $\mathcal{E}_{\langle m,j\rangle} = \{(\langle m, j\rangle, k) | k \in \{1, \ldots, K\}, (\langle m, j\rangle, k) \in \mathcal{E}\}$, the set of accessibility pairs of micro agent $\langle m, j\rangle$. Note that $\mathcal{E}_{\langle m,j\rangle} \cap \mathcal{E}_{\langle m',j'\rangle} = \emptyset$ if $m \neq m'$ or $j \neq j'$, and

$$\mathcal{E} = \bigcup_{m=1}^{M} \bigcup_{j=1}^{N_m} \mathcal{E}_{\langle m,j\rangle}$$

so that $\{\mathcal{E}_{\langle m,j\rangle}\}$ is a partition of $\mathcal{E}$. A feasible assignment S from the agents to the locations is a subset $S \subset \mathcal{E}$ such that

$$|S \cap \mathcal{E}_{\langle m,j\rangle}| \leq 1, \quad \forall \langle m, j\rangle$$

The value of this feasible assignment is given by

$$f(S) = 1 - \sum_{k=1}^{K} p_{k0} \prod_{(\langle m,j\rangle,k)\in S} (1 - \alpha_{mk}) \tag{3.18}$$

$f(S)$ has the following properties:

Lemma 3.4.1. *$f(S)$ is increasing and submodular in S.*

Proof. Given two feasible assignments $S' \subset S$, it is easy to see that $f(S') \leq f(S)$, because each term of the sum in (3.18) is smaller in $f(S')$, as all the products in S' are in S and S has additional products. Thus, $f(S)$ is increasing.

Furthermore, given $S' \subset S$ and $e = (\langle \tilde{m}, \tilde{j}\rangle, \tilde{k}) \notin S$

$$\begin{aligned} f(S' \cup \{e\}) - f(S') &= p_{k0}\alpha_{\tilde{m},\tilde{k}} \prod_{(\langle m,j\rangle,k)\in S'} (1 - \alpha_{mk}) \\ &\geq p_{k0}\alpha_{\tilde{m},\tilde{k}} \prod_{(\langle m,j\rangle,k)\in S} (1 - \alpha_{mk}) \\ &= f(S \cup \{e\}) - f(S) \end{aligned}$$

Thus, $f(S)$ is submodular in that it has the diminishing return property [Fisher et al., 1978]. ■

Let the collection of feasible assignments be $\mathcal{I}$. Then $(\mathcal{E}, \mathcal{I})$ has the following property:

Lemma 3.4.2. *The pair $(\mathcal{E}, \mathcal{I})$ is a matroid.*

Proof. A pair $(\mathcal{E}, \mathcal{I})$ where $\mathcal{E}$ is a finite set and $\mathcal{I}$ is a family of subsets of $\mathcal{E}$ is a matroid if [Welsh, 1976]:

1. $\emptyset \in \mathcal{I}$.
2. (Hereditary Property) For each $S' \subset S \subset \mathcal{E}$, if $S \in \mathcal{E}$ then $S' \in \mathcal{E}$.
3. (Augmentation Property) If $S \in \mathcal{I}$, $S' \in \mathcal{I}$ and $|S'| < |S|$, then there exists $e \in S \setminus S'$ such that $S' \cup \{e\} \in \mathcal{I}$.

For any feasible assignment $S \subset \mathcal{I}$, any $S' \subset S$ is also a feasible assignment, satisfying the hereditary property of independent sets required for a matroid.

Furthermore, assume we have two feasible assignments S, S' such that $|S'| < |S|$. Then, there exists a micro agent $\langle m, j\rangle$ such that $(\langle m, j\rangle, k) \in S$, and $(\langle m, j\rangle, k') \notin S'$ for all $k' = 1, \ldots, K$. This implies that $S' \cup \{(\langle m, j\rangle, k)\}$ is also a feasible assignment, satisfying the augmentation property and establishing that $(\mathcal{E}, \mathcal{I})$ is a matroid. ■

Hence, the equivalent problem of maximizing (3.18) subject to $S \in \mathcal{I}$ is a submodular maximization problem subject to a matroid constraint.

3.4.2 A Greedy Algorithm

We propose a simple greedy algorithm here. Before we present the algorithm, we first show how to compute the marginal gain when adding one more element to a feasible set S.

Lemma 3.4.3. *Given a feasible assignment S and a micro agent $\langle m', j'\rangle$ such that $S \cap \mathcal{E}_{\langle m', j'\rangle} = 0$, then for $(m', k') \in \mathcal{E}_{\langle m', j'\rangle}$,*

$$f(S \cup \{(m', k')\}) - f(S) = \Big[p_{k'0} \prod_{(\langle m,j\rangle, k') \in S} (1 - \alpha_{mk'})\Big] \alpha_{m'k'}$$

Proof.

$$\begin{aligned} & f(S \cup \{(m', k')\}) \\ =& 1 - \sum_{k \neq k'} p_{k0} \prod_{(\langle m,j\rangle, k) \in S} (1 - \alpha_{mk}) - p_{k'0} \prod_{(\langle m,j\rangle, k') \in S} (1 - \alpha_{mk'})(1 - \alpha_{m'k'}) \end{aligned}$$

and

$$\begin{aligned} & f(S\}) \\ =& 1 - \sum_{k} p_{k0} \prod_{(\langle m,j\rangle, k) \in S} (1 - \alpha_{mk}) \\ =& 1 - \sum_{k \neq k'} p_{k0} \prod_{(\langle m,j\rangle, k) \in S} (1 - \alpha_{mk}) - p_{k'0} \prod_{(\langle m,j\rangle, k') \in S} (1 - \alpha_{mk'}) \end{aligned}$$

Thus,

$$f(S \cup \{(m', k')\}) - f(S) = \Big[p_{k'0} \prod_{(\langle m,j\rangle,k')\in S} (1 - \alpha_{mk'})\Big]\alpha_{m'k'}$$

■

From Lemma 3.4.3, we can keep track of the values $p_{k0}\prod_{(\langle m,j\rangle,k)\in S}(1-\alpha_{mk})$ for all k in a vector $\boldsymbol{V}$. For a given agent m, to pick the location k that yields the largest marginal gain, we only need to pick $k^* = \arg\max_{k:(m,k)\in\mathcal{A}} \boldsymbol{V}_{(k)}\alpha_{mk}$. The algorithm thus goes as follows:

- Start with $S_0 = \emptyset$ (i.e., $\boldsymbol{x} = 0$) and $i = 1$. Define available supply at each source as R_m and let $R_m = N_m$. Let $\boldsymbol{V} = (p_{10}, \ldots, p_{K0})$.
- For $m = 1, \ldots, M$:
  - For $j = 1, \ldots, N_m$:
    - Find $e^* = \arg\max_{e\in\mathcal{E}_{\langle m,j\rangle}} \big[f(S_{i-1} \cup \{e\}) - f(S_{i-1})\big]$. This is equivalent to finding

      $$k^* = \underset{k:(m,k)\in\mathcal{A}}{\arg\max}\ \boldsymbol{V}_{(k)}\alpha_{mk}$$

      where $\boldsymbol{V}_{(k)}$ is the k-th element of $\boldsymbol{V}$.
    - Let $S_i = S_{i-1} \cup \{e^*\}$ and $x_{mk^*} = x_{mk^*} + 1$. Update $V_{(k^*)} = V_{(k^*)}(1 - \alpha_{mk^*})$. Increment $i = i + 1$.
- Return the greedy solution $S_{greedy} = S_N$.

Theorem 3.4.4. *The value of the solution S_{greedy} returned by the greedy algorithm above is at least $\frac{1}{2}$ of the value of the optimal solution S^*.*

Proof. Denote the solution returned by the greedy algorithm by $S_{greedy} = S_N$ and the optimal solution by S^*. Let $T^* = S^* \backslash S_{greedy}$ and denote the elements in T^* by $\{e_1^*, \ldots, e_{|T^*|}^*\}$.

Let $T_{\langle m,j\rangle} = (S^* \cap \mathcal{E}_{\langle m,j\rangle}) \backslash S_{greedy}, \forall \langle m, j\rangle$. Since $\{\mathcal{E}_{\langle m,j\rangle}\}$ is a partition of $\mathcal{E}$, $\{T_{\langle m,j\rangle}\}$ is a partition of T^*. Then,

$$
\begin{aligned}
f(S^*) \leq & f(S_{greedy} \cup S^*) \\
= & f(S_{greedy} \cup T^*) \\
= & f(S_{greedy}) + \sum_{l=1}^{|T^*|} [f(S_{greedy} \cup \{e_1^*, \ldots, e_l^*\}) - f(S_{greedy} \cup \{e_1^*, \ldots, e_{l-1}^*\})] \\
\leq & f(S_{greedy}) + \sum_{l=1}^{|T^*|} [f(S_{greedy} \cup \{e_l^*\}) - f(S_{greedy})] \qquad (3.19) \\
= & f(S_{greedy}) + \sum_{e \in T^*} [f(S_{greedy} \cup \{e\}) - f(S_{greedy})] \\
= & f(S_{greedy}) + \sum_{m=1}^{M} \sum_{j=1}^{N_m} \sum_{e \in T_{\langle m,j\rangle}} [f(S_{greedy} \cup \{e\}) - f(S_{greedy})] \\
\leq & f(S_{greedy}) + \sum_{m=1}^{M} \sum_{j=1}^{N_m} \sum_{e \in S^* \cap \mathcal{E}_{\langle m,j\rangle}} [f(S_{greedy} \cup \{e\}) - f(S_{greedy})] \\
\leq & f(S_{greedy}) + \sum_{i=1}^{N} \sum_{e \in S^* \cap \mathcal{E}_{\langle m,j\rangle}} [f(S_{i-1} \cup \{e\}) - f(S_{i-1})] \qquad (3.20) \\
\leq & f(S_{greedy}) + \sum_{i=1}^{N} [f(S_i) - f(S_{i-1})] \qquad (3.21) \\
= & 2f(S_{greedy})
\end{aligned}
$$

Inequalities (3.19) and (3.20) are due to the submodularity of $f(S)$. Inequality (3.21) holds since the greedy algorithm finds the best element in $\mathcal{E}_{\langle m,j\rangle}$ that augments the previous set S_{i-1} and $|S^* \cap \mathcal{E}_{\langle m,j\rangle}| = 1$. ■

This approximation ratio can be further improved to $(1 - \frac{1}{e})$ by using a randomized algorithm with pipage rounding and a continuous greedy process [Calinescu et al., 2011].

3.4.3 A Block Coordinate Ascent Algorithm

In this section, we develop another algorithm based on block coordinate ascent methods. The algorithm picks one agent $\tilde{m}$ at each iteration and solves an equivalent single-agent allocation problem:

$$\underset{\boldsymbol{x}_{\tilde{m}}=(x_{\tilde{m}1},\ldots,x_{\tilde{m}K})}{\text{maximize}} \quad \sum_{k=1}^{K} q_{k0}(1-(1-\alpha_{\tilde{m}k})^{x_{\tilde{m}k}}) \tag{3.22}$$

$$\text{subject to} \quad \sum_{k:(\tilde{m},k)\in\mathcal{A}} x_{\tilde{m}k} = N_{\tilde{m}} \tag{3.23}$$

$$x_{\tilde{m}k} \in \{0,1,\ldots,N_{\tilde{m}}\}, \ \ \forall(\tilde{m},k)\in\mathcal{A} \tag{3.24}$$

$$x_{\tilde{m}k} = 0, \ \ \forall(\tilde{m},k)\notin\mathcal{A} \tag{3.25}$$

where $q_{k0} = p_{k0}\prod_{m\neq\tilde{m}}(1-\alpha_{mk})^{x_{mk}}$ for $k = 1,\ldots,K$ are constants. Its objective function (3.22) is obtained by fixing the allocations of the other $M-1$ agents and adding a constant $\sum_{k=1}^{K} q_{k0} - 1$ in the multi-agent objective function (3.1).

To solve the equivalent single-agent problem (3.22)–(3.25), note that each term in (3.22) can be transformed into linear form:

$$q_{k0}(1-(1-\alpha_{\tilde{m}k})^{x_{\tilde{m}k}}) = \underset{\{y_{kj}\}}{\text{maximize}} \quad \sum_{j=1}^{N_{\tilde{m}}} q_{kj}y_{kj}$$

$$\text{subject to} \quad \sum_{j=1}^{N_{\tilde{m}}} y_{kj} = x_{\tilde{m}k}$$

$$y_{kj}\in\{0,1\}, \ \ \forall(\tilde{m},k)\in\mathcal{A}, j$$

$$y_{kj} = 0, \ \ \forall(\tilde{m},k)\notin\mathcal{A}, j$$

where $q_{kj} = q_{k0}(1-\alpha_{\tilde{m}k})^{j-1}\alpha_{\tilde{m}k}$.

Thus, the problem (3.22)–(3.25) becomes:

$$\underset{\boldsymbol{x}_{\tilde{m}}=(x_{\tilde{m}1},\ldots,x_{\tilde{m}K})}{\text{maximize}} \quad \sum_{k=1}^{K}\sum_{j=1}^{N_{\tilde{m}}} q_{kj}y_{kj} \tag{3.26}$$

$$\text{subject to} \quad \sum_{k:(\tilde{m},k)\in\mathcal{A}} x_{\tilde{m}k} = N_{\tilde{m}} \tag{3.27}$$

$$\sum_{j=1}^{N_{\tilde{m}}} y_{kj} = x_{\tilde{m}k}, \forall k \tag{3.28}$$

$$y_{kj} \in \{0,1\}, \ \ \forall(\tilde{m},k)\in\mathcal{A}, j \tag{3.29}$$

$$y_{kj} = 0, \ \ \forall(\tilde{m},k)\notin\mathcal{A}, j \tag{3.30}$$

To solve (3.26)–(3.30), one needs to find the $N_{\tilde{m}}$ largest values of q_{kj} for k such that $(\tilde{m},k)\in\mathcal{A}$ and $j = 1,\ldots,N_{\tilde{m}}$. An efficient implementation is to use a max-oriented heap. Initially, let $x_{\tilde{m}k} = 0$ for $k = 1,\ldots,K$ and let $R_{\tilde{m}} = N_{\tilde{m}}$. Insert $q_{k0}\alpha_{\tilde{m}k}$ for k such that $(\tilde{m},k)\in\mathcal{A}$ into a max-oriented heap. Then, while $R_{\tilde{m}} > 0$, repeat: (a) pop the smallest element v^* from the heap which corresponds to location k^*; (b) increment $x_{\tilde{m}k^*} = x_{\tilde{m}k^*} + 1$ and decrement $R_{\tilde{m}} = R_{\tilde{m}} - 1$; (c) insert $v^*(1-\alpha_{\tilde{m}k})$ into the heap.

Our block coordinate ascent algorithm is as follows:

- **Initialization:** Let x_{mk}, $\forall m,k$ have arbitrary values. They do not have to be either integer-valued or feasible.
- **Step 1:** For $\tilde{m} = 1,\ldots,M$, solve the equivalent single-agent allocation problem for agent $\tilde{m}$ (3.22)–(3.25) using the heap implementation.
- **Step 2:** Repeat **Step 1** until convergence, i.e., the objective value of the multi-agent problem (3.1) does not improve after one cycle of iterations.

3.4.4 Upper Bound of the Optimal Value

Consider the following nonlinear program which is a continuous relaxation of the integer nonlinear program (3.1)–(3.4):

$$\underset{\boldsymbol{x}}{\text{maximize}} \ 1 - \sum_{k=1}^{K} p_{k0} \prod_{m=1}^{M} (1-\alpha_{mk})^{x_{mk}} \tag{3.31}$$

$$\text{subject to} \sum_{k:(m,k)\in\mathcal{A}} x_{mk} = N_m, \ \forall m \tag{3.32}$$

$$0 \le x_{mk} \le N_m, \ \forall (m,k) \in \mathcal{A} \tag{3.33}$$

$$x_{mk} = 0, \ \forall (m,k) \notin \mathcal{A} \tag{3.34}$$

It is easy to see that the optimal value of the relaxed problem (3.31)–(3.34) is an upper bound of the integer nonlinear program (3.1)–(3.4). Also, the objective function (3.31) is concave and continuously differentiable, since it is a sum of negative exponential functions in the decision variables.

We develop a block coordinate ascent algorithm to solve this relaxed problem. By optimizing over $\boldsymbol{x}_{\tilde{m}} = (x_{\tilde{m}1}, \ldots, x_{\tilde{m}K})$ for agent $\tilde{m}$ while fixing the allocations of the other $M-1$ agents, we get an equivalent single-agent allocation problem:

$$\begin{aligned} \underset{\boldsymbol{x}_{\tilde{m}}=(x_{\tilde{m}1},\ldots,x_{\tilde{m}K})}{\text{maximize}} \ & 1 - \sum_{k:(m,k)\in\mathcal{A}} q_{k0}(1-\alpha_{\tilde{m}k})^{x_{\tilde{m}k}} \\ \text{subject to} & \sum_{k:(\tilde{m},k)\in\mathcal{A}} x_{\tilde{m}k} = N_{\tilde{m}} \\ & x_{\tilde{m}k} \ge 0, \ \forall (\tilde{m},k) \in \mathcal{A} \end{aligned}$$

where $q_{k0} = p_{k0} \prod_{m\neq\tilde{m}} (1-\alpha_{mk})^{x_{mk}}$. Its Lagrangian function is

$$L(\boldsymbol{x}_{\tilde{m}}, \lambda) = 1 - \sum_{k:(\tilde{m},k)\in\mathcal{A}} q_{k0}(1-\alpha_{\tilde{m}k})^{x_{\tilde{m}k}} + \lambda\Big(\sum_{k:(\tilde{m},k)\in\mathcal{A}} x_{\tilde{m}k} - N_{\tilde{m}}\Big)$$

If $\boldsymbol{x}^*_{\tilde{m}}$ is a global maximum, then there exists a scalar λ^* such that

$$\boldsymbol{x}^*_{\tilde{m}} \in \arg\max_{\boldsymbol{x}_{\tilde{m}}\ge 0} L(\boldsymbol{x}_{\tilde{m}}, \lambda^*)$$

The optimality conditions for an orthant constraint yield

$$-q_{k0}\ln(1-\alpha_{\tilde{m}k})(1-\alpha_{\tilde{m}k})^{x^*_{\tilde{m}k}}+\lambda^*\leq 0,\ \ k=1,\ldots,K$$

$$-q_{k0}\ln(1-\alpha_{\tilde{m}k})(1-\alpha_{\tilde{m}k})^{x^*_{\tilde{m}k}}+\lambda^*= 0,\ \text{if } x^*_{\tilde{m}k}>0$$

Thus, the equivalent single-agent problem can be solved using a scalar optimization technique to find λ^* and the corresponding $\boldsymbol{x}^*_{\tilde{m}}$ that satisfy the feasibility constraints and the optimality conditions.

To the multi-agent relaxed problem, we can iteratively solve equivalent single-agent problems, for $\tilde{m}=1,\ldots,M$, until convergence. It converges to a global maximum.

Theorem 3.4.5. *The block coordinate ascent algorithm which iteratively solves equivalent single-agent problems finds a global maximum of the multi-agent relaxed problem.*

Proof. The objective function (3.31) is continuously differentiable and concave, maximizing over some $\boldsymbol{x}_{\tilde{m}}$ at each iteration. The feasible sets of $\boldsymbol{x}_{\tilde{m}}$ are compact and convex for all $\tilde{m}$, as is the feasible set of $\boldsymbol{x}$. By Proposition 6 of [Grippo and Sciandrone, 2000], the limit point of the block coordinate ascent algorithm is a local maximum of (3.31), which is also a global maximum due to the concavity of (3.31). ■

3.5 Experiments

3.5.1 Experiments for Homogeneous Agent Search

In this section, we perform four experiments to compare the running time of our algorithm with three general min-cost flow algorithms: the cost scaling algorithm, the network simplex algorithm, and the capacity scaling algorithm. The cost scaling algorithm used here is an efficient implementation by Goldberg [Goldberg, 1997], known as *CS2*. The network simplex algorithm and the capacity scaling algorithm used here are implemented in the LEMON graph library [Dezső et al., 2011]. It is worth noting that these algorithms require integral arc costs, so we truncated the fractional costs in our models to 5 significant digits and multiplied them by 10^5.

We conduct the experiments using the C language on a laptop computer with Intel i7-4600M processor and 8GB RAM. For each experiment, we randomly generate 10 instances. Each instance has 200 agents and 20000 locations. For each instance, we randomly generate the prior probability p_{k0} and the probability of detection α_k at each location k. We also choose a sparsity ratio defined as:

$$r = \frac{|\mathcal{A}|}{MK}$$

where $\mathcal{A}$ is the accessibility set M is the number of agents, and K is the number of locations. We randomly generate the accessibility set $\mathcal{A}$ with the given sparsity ratio.

In the first experiment, we let each agent have a constant budget equal to 10, and set the sparsity ratio to be 5%. In the second experiment, we still let each agent have a constant budget equal to 10, but reduce the sparsity ratio to 3% to make the corresponding network flow graph more sparse. In the third experiment, we randomly generate budgets for each agent while making the total budgets of all agents sum up to $10MK$, and set the sparsity ratio to be 5%. In the fourth experiment, we randomly generate budget for each agent while making the total budgets of all agents sum up to $10MK$, and set the sparsity ratio to be 3%.

For each experiment, we run all four algorithms and average the computation time over 10 random instances. Computation time in milliseconds is given in Table 3.2. From the results, we can see that our algorithm is faster by factors of 20 to 50 than optimized library algorithms.

In addition, it can be seen that the more sparse the problem is, the longer it takes to solve the problem for any algorithm. For the same sparsity ratio, problems with random supply take longer to solve than problems with constant supply for any algorithm. We can also see that our algorithm scales the best when the problems become more sparse and random.

Table 3.2: Comparison of computation time for four different algorithms.

Algorithms	200×20000, 5% sparsity, const. supply	200×20000, 3% sparsity, const. supply	200×20000, 5% sparsity, rand. supply	200×20000, 3% sparsity, rand. supply
Our algorithm	3.2ms	4.6ms	4.2ms	12.5ms
Cost scaling (CS2)	75.8ms	186ms	435ms	764ms
Network simplex	213ms	1870ms	1214ms	3220ms
Capacity scaling	117ms	311ms	625ms	987ms

3.5.2 Experiments for Heterogeneous Agent Search

In this section, we do experiments for the case of multiple heterogeneous agents. We generate 5 models. Each model has 20 agents and 400 locations. The sparsity ratio is set to 5%. Each agent has a budget of 10. We compute the upper bound of the optimal values by solving the relaxed problem. We compare the objective values (final probability of detection) of the greedy algorithm in Section 3.4.2, the block coordinate ascent algorithm (the initial assignment $\boldsymbol{x}$ is using the optimal solution for the relaxed problem) in Section 3.4.3, and a purely random approach which randomly generates 1000 feasible search plans and picks the one with the best value.

The results are given in Table 3.3. From the results, we can see that the greedy algorithm has found solutions slightly better than the block coordinate ascent algorithm. Both algorithms have found solutions that are close to the upper bound and are much better than the purely random approach.

Table 3.3: Comparison of objective values found by different algorithms for heterogeneous agent search. The asterisk (*) indicates the best value found.

Model No.	Greedy	Block coordinate ascent	Purely random	Upper bound
1	0.583*	0.574	0.353	0.611
2	0.558*	0.552	0.333	0.610
3	0.585*	0.573	0.356	0.612
4	0.550*	0.541	0.337	0.603
5	0.576*	0.569	0.349	0.605

Chapter 4

Multi-Agent Sparse Search of a Markovian Moving Object with Simple Error Models

In this chapter, we study the problem of searching for a Markovian object with multiple search agents in discrete time and space. Similar like in Chapter 3, we restrict the agents to have limited visibilities, so each agent can only search a subset of the locations at each time (i.e. a sparse search problem). We generalize the approach of Chapter 3 to search for Markovian objects, formulating the problem as an integer nonlinear programming problem. We develop necessary conditions for an optimal search plan, extending results in [Brown, 1980, Washburn, 1983]. We develop a forward-and-backward (FAB) algorithm which repeatedly solves subproblems in both forward and backward passes and generates solutions satisfying the necessary conditions. It is essentially a block coordinate descent algorithm and may not converge to globally optimal solutions. For the special case where the probabilities of detection depend only on the location being searched and the time of search (but not the agent), the coordinate descent iterations reduce to the solution of an equivalent stationary target search problem using the fast algorithms of Chapter 3. To improve the quality of the solutions, we develop a convex relaxation of the integer programming problem, and provide coordinate descent algorithms for obtaining optimal solutions. These relaxed solutions are used to generate random multiple starting solutions for the iterative integer coordinate descent algorithms. We illustrate the performance of our algorithms with experiments and compare the results with alternative algorithms in [Royset and

Sato, 2010] based on combinatorial techniques.

We also address the heterogeneous agent search problem where the probabilities of detection depend on the location, the time period as well as the agent. We formulate an integer nonlinear program, and show that there is an equivalent set based formulation with a monotone submodular objective function and matroid constraints. We propose two greedy algorithms, and show that a greedy-style algorithm is guaranteed to produce a solution with value at least $\frac{1}{2}$ of the optimal objective value. We also provide a lower bound of the optimal objective value (total probability of non-detection) by solving the relaxed heterogeneous agent search problem.

The chapter is organized as follows: In Section 4.1, we formulate the problem. In Section 4.2, we present the necessary conditions of optimal search plans for multi-agent sparse search of a moving object. In Section 4.3.2, we first present the integer forward-and-backward (FAB) algorithm with arbitrary initial search allocation. We give a counterexample that the algorithm does not find an optimal plan of the integer problem. We propose a convex relaxation of the integer problem, and show that by applying the block coordinate descent method we can find an optimal solution of the relaxed problem. Then we propose a two-stage FAB algorithm. In Section 4.4, we study the general case of heterogeneous agents where the probabilities of detection also depend on agents. Section 4.5 contains the experiment results.

4.1 Problem Formulation

Consider searching a moving object hidden in one of K locations with M agents in T discrete time periods. The object moves according to a known time-homogeneous Markov chain with transition probability matrix P where ρ_{ij} denotes the probability of the object moving from location i to location j in consecutive time periods. Let x_t denote the object location at time t, and let $\boldsymbol{x}$ denote the trajectory of the object.

Assume that before any searches take place, the prior probability vector of the object over locations is $\boldsymbol{\pi}_1 = [\pi_{1,1}, \ldots, \pi_{1,K}]$.

Each agent can search only a subset of the locations. Define an *accessibility pair* (m, k) if agent m has access to location k. Let $\mathcal{A}$ denote the set of all accessibility pairs. Without loss of generality, assume that each agent can access at least one location and each location is accessible to at least one agent.

Assume that an agent detects the object at its accessible location k with probability α_k, if the object is at location k. Note that the probability of detection only depends on the location, but not on the agent. If the object is not at location k, a search of the location always yields no detection. Assume that agent observations are conditionally independent across locations and time, given current object and agent locations.

Let $b_{t,m,k} \in \{0, 1\}$ denote whether agent m searches location k at time t. Agent m can search N_m locations at each time. Hence, $\sum_{k:(m,k)\in\mathcal{A}_t} b_{t,m,k} = N_m$ for all t and m. Let $\boldsymbol{b}_t = \{b_{t,m,k}, \forall (m, k) \in \mathcal{A}\}$ be the search allocation for time t. Our objective is to find a finite-horizon search allocation $\boldsymbol{b} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_T]$ to minimize the probability of not detecting the object in T time periods.

With the above notation, the probability that an object at location x_t at time t is not detected given $\boldsymbol{b}_t$ is

$$P_{miss}(x_t, \boldsymbol{b}_t) = \prod_{m=1}^{M} (1 - \alpha_{t,m,x_t})^{b_{t,m,x_t}}$$

Let $p(\boldsymbol{x})$ denote the probability that the object follows trajectory $\boldsymbol{x}$. Then, the probability of not detecting the object given a search $\boldsymbol{b}$ is

$$f(\boldsymbol{b}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) \prod_{t=1}^{T} P_{miss}(x_t, \boldsymbol{b}_t)$$

$$= \sum_{\boldsymbol{x}} p(\boldsymbol{x}) \prod_{t=1}^{T} \prod_{m=1}^{M} (1 - \alpha_{t,m,x_t})^{b_{t,m,x_t}} \tag{4.1}$$

The optimal multi-agent sparse search problem is then

$$\underset{\boldsymbol{b}}{\text{minimize}} \;\; f(\boldsymbol{b}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) \prod_{t=1}^{T} \prod_{m=1}^{M} (1 - \alpha_{t,m,x_t})^{b_{t,m,x_t}} \tag{4.2}$$

$$\text{subject to} \sum_{k:(m,k)\in\mathcal{A}_t} b_{t,m,k} = N_m, \;\; \forall t, m \tag{4.3}$$

$$b_{t,m,k} \in \{0, \ldots, N_m\}, \;\; \forall t, \forall (m,k) \in \mathcal{A}_t \tag{4.4}$$

$$b_{t,m,k} = 0, \;\; \forall t, \forall (m,k) \notin \mathcal{A}_t \tag{4.5}$$

When the object motion is Markovian, the objective function $f(\boldsymbol{b})$ has a matrix form. Let the transition probability matrix P be written as

$$P = \begin{bmatrix} \rho_{11} & \cdots & \rho_{K1} \\ \vdots & \ddots & \vdots \\ \rho_{1K} & \cdots & \rho_{KK} \end{bmatrix}$$

P is a left stochastic matrix, i.e., each column of P sums up to one.

Define diagonal matrices $I_t(\boldsymbol{b}_t) \in \mathcal{R}^{K\times K}$ as

$$I_t(\boldsymbol{b}_t) = \text{diag} \begin{bmatrix} \prod_{m=1}^{M} (1 - \alpha_{t,m,1})^{b_{t,m,1}} \\ \vdots \\ \prod_{m=1}^{M} (1 - \alpha_{t,m,K})^{b_{t,m,K}} \end{bmatrix}$$

Using the Markov property, we can rewrite the objective function $f(\boldsymbol{b})$ in matrix form.

Lemma 4.1.1.

$$f(\boldsymbol{b}) = \mathbf{1}^T \Big[\prod_{t=1}^{T} P I_t(\boldsymbol{b}_t) \Big] \boldsymbol{\pi}_1$$

where $\mathbf{1}$ *is the column vector* $[1\ 1\ \cdots\ 1]$ *and* $\mathbf{1}^T$ *its transpose.*

Proof. By Markov property, we can expand $p(\boldsymbol{x})$ and write $f(\boldsymbol{b})$ as

$$f(\boldsymbol{b}) = \sum_{x_T=1}^{K} \cdots \sum_{x_1=1}^{K} \pi_{1,x_1}\rho_{x_1,x_2}\cdots\rho_{x_{T-1},x_T} \prod_{t=1}^{T}\prod_{m=1}^{M}(1-\alpha_{t,m,x_t})^{b_{t,m,x_t}}$$

Let $\boldsymbol{v}_{(k)}$ denote the k-th element of a vector $\boldsymbol{v}$. Then,

$$\begin{aligned}
&f(\boldsymbol{b}) \\
&= \sum_{x_T=1}^{K} \cdots \sum_{x_2=1}^{K} \rho_{x_2,x_3}\cdots\rho_{x_{T-1},x_T} \prod_{t=2}^{T}\prod_{m=1}^{M}(1-\alpha_{t,m,x_t})^{b_{t,m,x_t}} \\
&\quad \cdot \Big(\sum_{x_1=1}^{K} \pi_{1,x_1}\rho_{x_1,x_2} \prod_{m=1}^{M}(1-\alpha_{1,m,x_1})^{b_{1,m,x_1}} \Big) \\
&= \sum_{x_T=1}^{K} \cdots \sum_{x_2=1}^{K} \rho_{x_2,x_3}\cdots\rho_{x_{T-1},x_T} \prod_{t=2}^{T}\prod_{m=1}^{M}(1-\alpha_{t,m,x_t})^{b_{t,m,x_t}} \\
&\quad \cdot \Big(\pi_{1,1}\rho_{1,x_2} \prod_{m=1}^{M}(1-\alpha_{1,m,1})^{b_{1,m,1}} + \cdots + \pi_{1,K}\rho_{K,x_2} \prod_{m=1}^{M}(1-\alpha_{1,m,K})^{b_{1,m,K}} \Big) \\
&= \sum_{x_T=1}^{K} \cdots \sum_{x_2=1}^{K} \rho_{x_2,x_3}\cdots\rho_{x_{T-1},x_T} \prod_{t=2}^{T}\prod_{m=1}^{M}(1-\alpha_{t,m,x_t})^{b_{t,m,x_t}} \big(PI_1(\boldsymbol{b}_1)\boldsymbol{\pi}_1\big)_{(x_2)} \\
&= \sum_{x_T=1}^{K} \cdots \sum_{x_3=1}^{K} \rho_{x_3,x_4}\cdots\rho_{x_{T-1},x_T} \prod_{t=3}^{T}\prod_{m=1}^{M}(1-\alpha_{t,m,x_t})^{b_{t,m,x_t}} \\
&\quad \cdot \Big(\sum_{x_2=1}^{K} (PI_1(\boldsymbol{b}_1)\boldsymbol{\pi}_1)_{(x_2)}\rho_{x_2,x_3} \prod_{m=1}^{M}(1-\alpha_{2,m,x_2})^{b_{2,m,x_2}} \Big) \\
&= \sum_{x_T=1}^{K} \cdots \sum_{x_3=1}^{K} \rho_{x_3,x_4}\cdots\rho_{x_{T-1},x_T} \prod_{t=3}^{T}\prod_{m=1}^{M}(1-\alpha_{t,m,x_t})^{b_{t,m,x_t}} \\
&\quad \cdot \Big((PI_1(\boldsymbol{b}_1)\boldsymbol{\pi}_1)_{(1)}\rho_{1,x_3} \prod_{m=1}^{M}(1-\alpha_{2,m,1})^{b_{2,m,1}} + \\
&\quad \cdots + (PI_1(\boldsymbol{b}_1)\boldsymbol{\pi}_1)_{(K)}\rho_{K,x_3} \prod_{m=1}^{M}(1-\alpha_{2,m,K})^{b_{2,m,K}} \Big) \\
&= \sum_{x_T=1}^{K} \cdots \sum_{x_3=1}^{K} \rho_{x_3,x_4}\cdots\rho_{x_{T-1},x_T} \prod_{t=3}^{T}\prod_{m=1}^{M}(1-\alpha_{t,m,x_t})^{b_{t,m,x_t}} \big(PI_2(\boldsymbol{b}_2)PI_1(\boldsymbol{b}_1)\boldsymbol{\pi}_1\big)_{(x_3)}
\end{aligned}$$

$$= \cdots = \sum_{x_T=1}^{K} \prod_{m=1}^{M} (1 - \alpha_{T,m,x_T})^{b_{T,m,x_T}} \Big(\Big[\prod_{t=1}^{T-1} PI_t(\boldsymbol{b}_t) \Big] \boldsymbol{\pi}_1 \Big)_{(x_T)}$$

$$= \mathbf{1}^T I_T(\boldsymbol{b}_T) \Big[\prod_{t=1}^{T-1} PI_t(\boldsymbol{b}_t) \Big] \boldsymbol{\pi}_1 = \mathbf{1}^T \Big[\prod_{t=1}^{T} PI_t(\boldsymbol{b}_t) \Big] \boldsymbol{\pi}_1$$

The last equality holds since $\mathbf{1}^T P = \mathbf{1}^T$. ■

4.2 Optimality Conditions

We refer to (4.2)–(4.5) as *the integer problem*. We now provide necessary conditions for optimal solutions of the integer problem.

For a given time period t, $f(\boldsymbol{b})$ can be decomposed as

$$f(\boldsymbol{b}) = \mathbf{1}^T I_T(\boldsymbol{b}_T) \Big[\prod_{s=t+1}^{T-1} PI_s(\boldsymbol{b}_s) \Big] PI_t(\boldsymbol{b}_t) \Big[\prod_{s=1}^{t-1} PI_s(\boldsymbol{b}_s) \Big] \boldsymbol{\pi}_1$$

$$= \Big(\mathbf{1}^T \Big[\prod_{s=t+1}^{T} I_s(\boldsymbol{b}_s) P \Big] \Big) I_t(\boldsymbol{b}_t) \Big(\Big[\prod_{s=1}^{t-1} PI_s(\boldsymbol{b}_s) \Big] \boldsymbol{\pi}_1 \Big)$$

We define column vectors $\boldsymbol{q}_R(t) \in \mathcal{R}^K$ and $\boldsymbol{q}_L(t) \in \mathcal{R}^K$, $t = 1, \ldots, T$, as follows:

$$\boldsymbol{q}_R(t) = \begin{cases} \boldsymbol{\pi}_1, \text{ if } t = 1 \\ PI_t(\boldsymbol{b}_t) \boldsymbol{q}_R(t-1), \text{ if } 2 \le t \le T \end{cases} \tag{4.6}$$

$$\boldsymbol{q}_L(t) = \begin{cases} \mathbf{1}, \text{ if } t = T \\ P^T I_t(\boldsymbol{b}_t) \boldsymbol{q}_L(t+1), \text{ if } 1 \le t \le T-1 \end{cases} \tag{4.7}$$

Therefore, for any given t, $f(\boldsymbol{b}) = \boldsymbol{q}_L^T(t) I_t(\boldsymbol{b}_t) \boldsymbol{q}_R(t)$.

Given $\boldsymbol{q}_R(t)$ and $\boldsymbol{q}_L(t)$, define *the t-th integer subproblem* as

$$\underset{\boldsymbol{b}_t}{\text{minimize}} \;\; f_t(\boldsymbol{b}_t) = \boldsymbol{q}_L^T(t) I_t(\boldsymbol{b}_t) \boldsymbol{q}_R(t) \tag{4.8}$$

$$\text{subject to} \sum_{k:(m,k)\in\mathcal{A}} b_{t,m,k} = N_m, \ \ \forall m \tag{4.9}$$

$$b_{t,m,k} \in \{0,\ldots,N_m\}, \ \ \forall (m,k) \in \mathcal{A}_t \tag{4.10}$$

$$b_{t,m,k} = 0, \ \ \forall (m,k) \notin \mathcal{A}_t \tag{4.11}$$

The following result states that an optimal solution for the integer problem must also be optimal for the related integer subproblems. It extends the necessary conditions in [Washburn, 1980] to multi-agent sparse search.

Theorem 4.2.1 (Necessary Conditions)**.** *Let* $\boldsymbol{b}^* = [\boldsymbol{b}_1^*,\ldots,\boldsymbol{b}_T^*]$ *denote the optimal solution for the integer problem. Let* $\boldsymbol{q}_R(t)$ *and* $\boldsymbol{q}_L(t)$ *be computed using all* $\boldsymbol{b}_s^*$ *for* $s \neq t$*. Then, for all* $t = 1,\ldots,T$*,* $\boldsymbol{b}_t^*$ *must minimize the* t*-th integer subproblem* (4.8)–(4.11) *given* $\boldsymbol{q}_R(t)$ *and* $\boldsymbol{q}_L(t)$*.*

Proof. Suppose there exists $\hat{\boldsymbol{b}}_t$ such that it is a feasible solution to the t-th subproblem and $f_t(\hat{\boldsymbol{b}}_t) < f_t(\boldsymbol{b}_t^*)$. Let $\hat{\boldsymbol{b}} = [\boldsymbol{b}_1^*,\ldots,\hat{\boldsymbol{b}}_t,\ldots,\boldsymbol{b}_T^*]$. Then,

(1) $\hat{\boldsymbol{b}}$ is a feasible solution to the integer problem since if $\hat{\boldsymbol{b}}_t$ satisfies the feasibility constraints (4.9)–(4.11) of the t-th subproblem, it also satisfies the feasibility constraints (4.3)–(4.5) of the integer problem. Besides, $\boldsymbol{b}_s^*$ for $s \neq t$ satisfies (4.3)–(4.5) too.

(2) $f(\hat{\boldsymbol{b}}) < f(\boldsymbol{b}^*)$ since $f(\hat{\boldsymbol{b}}) = f_t(\hat{\boldsymbol{b}}_t)$ and $f(\boldsymbol{b}^*) = f_t(\boldsymbol{b}_t^*)$.

This contradicts the assumption. Thus, $\boldsymbol{b}_t^*$ must minimize the t-th integer subproblem. ■

The integer subproblems of (4.8)-(4.11) are NP-hard subproblems, as proven in [Lloyd and Witsenhausen, 1986]. While there are approximate solutions available such as in [Ahuja et al., 2007] and Chapter 3, there is a subclass of problems which can be solved exactly by fast algorithms given in Chapter 3: problems for which the probability of detection $\alpha_{t,m,k}$ does not depend on the agent m. We refer to this assumption as homogeneous agent assumption, and focus on class of problems with this assumption next.

4.3 Algorithms for Homogeneous Agent Search

Under the homogeneous agent assumption, if any two agents can search the same location at a given time t, their probabilities of detection are equal: $\alpha_{t,m_1,k} = \alpha_{t,m_2,k} \triangleq \alpha_{t,k}$ for $(m_1, k) \in \mathcal{A}_t, (m_2, k) \in \mathcal{A}_t$. Then,

$$I_t(\boldsymbol{b}_t) = \text{diag} \begin{bmatrix} (1-\alpha_{t,1})^{\sum_{m=1}^{M} b_{t,m,1}} \\ \vdots \\ (1-\alpha_{t,K})^{\sum_{m=1}^{M} b_{t,m,K}} \end{bmatrix}$$

The integer subproblem of (4.8)-(4.11) reduces to

$$IP(t) \equiv \underset{\boldsymbol{b}_t}{\text{minimize}} \sum_{k=1}^{K} q_{k,0}(t)(1-\alpha_{t,k})^{\sum_{m=1}^{M} b_{t,m,k}} \tag{4.12}$$

$$\text{subject to} \sum_{k:(m,k)\in\mathcal{A}_t} b_{t,m,k} = N_m, \ \forall m \tag{4.13}$$

$$b_{t,m,k} \in \{0, \ldots, N_m\}, \ \forall (m,k) \in \mathcal{A}_t \tag{4.14}$$

$$b_{t,m,k} = 0, \ \forall (m,k) \notin \mathcal{A}_t \tag{4.15}$$

where $q_{k,0}(t) = \boldsymbol{q}_L(t)_{(k)}\boldsymbol{q}_R(t)_{(k)}, \forall k$. The results in Chapter 3 and [Castañón, 1987] show that problem $IP(t)$ can be reduced to a min-cost network flow problem, and solved optimally using fast algorithms as in Chapter 3.

4.3.1 The Integer FAB Algorithm

To solve the dynamic search problem of (4.2)–(4.5), we propose a forward-and-backward (FAB) algorithm corresponding to block coordinate descent optimization using the recursions (4.6) and (4.7). The algorithm is presented below.

Integer FAB Algorithm with Arbitrary Initial Allocation

Input: $\{\alpha_k, k = 1, \ldots, K\}$, $\boldsymbol{\pi}_1$.

Output: $\boldsymbol{b}$.

Initialization:

Start with an initial set of allocations $\boldsymbol{b}$. They do not have to be either feasible or integer-valued. Using (4.7) and $\boldsymbol{b}$, compute $\boldsymbol{q}_L(t)$ for $t = T, \ldots, 1$, and $\boldsymbol{q}_R(1)$.

Step 1 (Integer Forward Pass) Starting from $t = 1$ to $T - 1$:

1. Solve the integer subproblem $IP(t)$ in (4.12)–(4.15) to obtain $\boldsymbol{b}_t$.
2. Update $\boldsymbol{q}_R(t + 1)$ using (4.6).

Step 2 (Integer Backward Pass) Starting from $t = T$ to $t = 2$:

1. Solve the integer subproblem $IP(t)$ in (4.12)–(4.15) to obtain $\boldsymbol{b}_t$.
2. Update $\boldsymbol{q}_L(t - 1)$ using (4.7).

Step 3 (Repeat Integer Forward-Backward Passes) Repeat **Step 1–2** until convergence, i.e., after one cycle of iterations $f(\boldsymbol{a})$ does not improve.

After the first forward pass, the algorithm has a feasible integer solution to the multi-agent sparse search integer problem of (4.2)–(4.5). Furthermore, each subsequent step of the algorithm results in a non-increasing cost while maintaining feasibility. Since the number of feasible integer search allocations is finite, this establishes the following result:

Theorem 4.3.1. *The intermediate values found by the FAB algorithm for multi-agent sparse search are nonincreasing, and the algorithm converges in finite time.*

Proof. Suppose at one iteration, the t-th integer subproblem is solved and the search plan changes from $\boldsymbol{b}'$ to $\boldsymbol{b}''$. Let $\boldsymbol{b}' = \{\boldsymbol{b}'_1, \ldots, \boldsymbol{b}'_t, \ldots, \boldsymbol{b}'_T\}$, then $\boldsymbol{b}'' = \{\boldsymbol{b}'_1, \ldots, \boldsymbol{b}''_t, \ldots, \boldsymbol{b}'_T\}$. In other words, $\boldsymbol{b}'$ and $\boldsymbol{b}''$ differ only in the t-th time-wise component.

Besides, $f(\boldsymbol{b}') <= f(\boldsymbol{b}'')$ since: (1) $f_t(\boldsymbol{b}'_t) <= f_t(\boldsymbol{b}''_t)$; (2) $f(\boldsymbol{b}') = f_t(\boldsymbol{b}'_t)$; (3) $f(\boldsymbol{b}'') = f_t(\boldsymbol{b}''_t)$. Hence, the intermediate values found by the algorithm are nonincreasing.

Since there are a finite number of feasible solutions and the algorithm terminates when the objective value does not improve after one cycle of iterations, it will eventually converge in finite time. ■

Even though the algorithm produces a solution that satisfies the necessary conditions for optimal solutions, it does not in general converge to an optimal solution. In essence, the combination of the piecewise linear relaxations of the integer subproblems does not result in a convex function of the relaxed variables, and enables the iteration to stop short of optimality. A simple counterexample to optimality is shown next:

Let number of agents be $M = 1$, number of locations be $K = 2$ and time horizon be $T = 2$. Assume that the agent can search both locations. Let the prior probability vector over locations be $\boldsymbol{\pi}_1 = [0.4, 0.6]$. Let the probability of detection vector over locations be $\boldsymbol{\alpha} = [0.75, 0.25]$. Furthermore, let the Markovian transition probability matrix be $P = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$.

For this problem, a single round of forward or backward pass in the integer FAB algorithm consists of only one iteration since $T = 2$. The algorithm runs as follows:

1) Initially,

$$\boldsymbol{b}_1 = [0, 0],\ \boldsymbol{b}_2 = [0, 0]$$

$$\boldsymbol{q}_R(1) = [0.4, 0.6],\ \boldsymbol{q}_R(2) = [0, 0]$$

$$\boldsymbol{q}_L(1) = [1, 1],\ \boldsymbol{q}_L(2) = [1, 1]$$

The objective value $f(\boldsymbol{b})$ is 1.

2) At the 1st iteration, $t = 1$. If the agent searches location 1, $\boldsymbol{b}_1 = [1, 0]$ and $f_1(\boldsymbol{b}_1) = (1 - 0.75) \times 0.4 + 1 \times 0.6 = 0.7$. If the agent searches location 2, $\boldsymbol{b}_1 = [0, 1]$ and $f_1(\boldsymbol{b}_1) = 1 \times 0.4 + (1 - 0.25) \times 0.6 = 0.85 > 0.7$. So the agent will search location 1 at $t = 1$.

After the iteration, we have

$$\boldsymbol{b}_1 = [1, 0],\ \boldsymbol{b}_2 = [0, 0]$$
$$\boldsymbol{q}_R(1) = [0.4, 0.6],\ \boldsymbol{q}_R(2) = [0.15, 0.55]$$
$$\boldsymbol{q}_L(1) = [1, 1],\ \boldsymbol{q}_L(2) = [1, 1]$$

The objective value $f(\boldsymbol{b})$ becomes 0.7.

3) At the 2nd iteration, $t = 2$. If the agent searches location 1, $\boldsymbol{b}_2 = [1, 0]$ and $f_2(\boldsymbol{b}_2) = (1 - 0.75) \times 0.15 + 1 \times 0.55 = 0.5875$. If the agent searches location 2, $\boldsymbol{b}_2 = [0, 1]$ and $f_2(\boldsymbol{b}_2) = 1 \times 0.15 + (1 - 0.25) \times 0.55 = 0.5625 < 0.5875$. So the agent will search location 2 at $t = 2$.

After the iteration, we have

$$\boldsymbol{b}_1 = [1, 0],\ \boldsymbol{b}_2 = [0, 1]$$
$$\boldsymbol{q}_R(1) = [0.4, 0.6],\ \boldsymbol{q}_R(2) = [0.15, 0.55]$$
$$\boldsymbol{q}_L(1) = [0.975, 0.775],\ \boldsymbol{q}_L(2) = [1, 1]$$

The objective value $f(\boldsymbol{b})$ becomes 0.5625.

4) We do another round of forward and backward passes. The search plan $\boldsymbol{b}$ does not change, and the algorithm converges.

The search plan returned by the FAB algorithm is to search location 1 at time 1, then search location 2 at time 2. It has an objective value of 0.5625. However, the optimal search plan for this example is to search location 2 at time 1, then search

location 1 at time 2. The optimal objective value is 0.5463. Thus, the FAB algorithm did not converge to a global minimum for this example.

4.3.2 Convex Relaxation

We can obtain a convex optimization problem if we relax the the integer constraints (4.15) with an exponential relaxation. We call it *the relaxed problem*:

$$\underset{\boldsymbol{b}}{\text{minimize}} \;\; f(\boldsymbol{b}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) \prod_{t=1}^{T} (1 - \alpha_{t,x_t})^{\sum_{m=1}^{M} b_{t,m,x_t}} \tag{4.16}$$

$$\text{subject to} \sum_{k:(m,k)\in\mathcal{A}_t} b_{t,m,k} = N_m, \;\; \forall t, m \tag{4.17}$$

$$b_{t,m,k} \in [0, N_m], \;\; \forall t, \forall (m,k) \in \mathcal{A}_t \tag{4.18}$$

$$b_{t,m,k} = 0, \;\; \forall t, \forall (m,k) \notin \mathcal{A}_t \tag{4.19}$$

The relaxed problem can be viewed as allowing agents to spend fractional search effort in each location. Correspondingly, we define *the t-th relaxed subproblem* as

$$RP(t) \equiv \underset{\boldsymbol{b}_t}{\text{minimize}} \;\; f_t(\boldsymbol{b}_t) = \boldsymbol{q}_L^T(t) I_t(\boldsymbol{b}_t) \boldsymbol{q}_R(t) \tag{4.20}$$

$$\text{subject to} \sum_{k:(m,k)\in\mathcal{A}} b_{t,m,k} = N_m, \;\; \forall m \tag{4.21}$$

$$b_{t,m,k} \in [0, N_m], \;\; \forall (m,k) \in \mathcal{A}_t \tag{4.22}$$

$$b_{t,m,k} = 0, \;\; \forall (m,k) \notin \mathcal{A}_t \tag{4.23}$$

The objectives of both of the relaxed problem, as well as the relaxed subproblems $RP(t)$, can be shown to be convex and continuously differentiable, since they are sums of exponentials in the decision variables.

Lemma 4.3.2. *$f(\boldsymbol{b})$ is a continuously differentiable convex function.*

Proof. We first prove that $f(\boldsymbol{b})$ is continuously differentiable over the feasible region of the relaxed problem. Each term in $f(\boldsymbol{b})$ corresponds to an object motion trajectory.

Consider a trajectory $\boldsymbol{x}$ and $\boldsymbol{b}(\boldsymbol{x}) = (b_{1,1,x_1}, \ldots, b_{1,M,x_1}, \ldots, b_{T,1,x_T}, \ldots, b_{T,M,x_T}) \in \mathcal{R}^{TM}$, the term in $f(\boldsymbol{b})$ that corresponds to $\boldsymbol{x}$ is $g(\boldsymbol{b}(\boldsymbol{x})) = p(\boldsymbol{x}) \prod_{t=1}^{T} \prod_{m=1}^{M} (1 - \alpha_{t,x_t})^{b_{t,m,x_t}}$. Since $g(\boldsymbol{b}(\boldsymbol{x}))$ is continuously differentiable over the feasible region of $\boldsymbol{b}(\boldsymbol{x})$, $f(\boldsymbol{b}) = \sum_{\boldsymbol{x}} g(\boldsymbol{b}(\boldsymbol{x}))$ is continuously differentiable over the feasible region of $\boldsymbol{b}$.

We next prove that $f(\boldsymbol{b})$ is convex over the feasible region of the relaxed problem. Consider term $g(\boldsymbol{b}(\boldsymbol{x}))$ of $f(\boldsymbol{b})$, for any $b_{t',m',x_{t'}}$,

$$\frac{\partial g(\boldsymbol{b}(\boldsymbol{x}))}{\partial b_{t',m',x_{t'}}} = \Big[p(\boldsymbol{x}) \prod_{t=1}^{T} \prod_{m=1}^{M} (1 - \alpha_{t,x_t})^{b_{t,m,x_t}} \Big] \ln(1 - \alpha_{t',x_{t'}})$$

Then, for any $b_{t',m',x_{t'}}$ and $b_{t'',m'',x_{t''}}$,

$$\frac{\partial^2 g(\boldsymbol{b}(\boldsymbol{x}))}{\partial b_{t',m',x_{t'}} \partial b_{t'',m'',x_{t''}}} = \Big[p(\boldsymbol{x}) \prod_{t=1}^{T} \prod_{m=1}^{M} (1 - \alpha_{x_t})^{b_{t,m,x_t}} \Big] \ln(1 - \alpha_{t',x_{t'}}) \ln(1 - \alpha_{t'',x_{t''}})$$

Define column vector $\boldsymbol{u} = \big(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_T\big) \in \mathcal{R}^{TM}$ where $\boldsymbol{u}_t = \big(\ln(1 - \alpha_{t,x_t}), \ldots, \ln(1 - \alpha_{t,x_t})\big) \in \mathcal{R}^{M}$ for all t. The Hessian matrix $\nabla^2 g(\boldsymbol{b}(\boldsymbol{x}))$ is equivalent to

$$\nabla^2 g(\boldsymbol{b}(\boldsymbol{x})) = \Big[p(\boldsymbol{x}) \prod_{t=1}^{T} \prod_{m=1}^{M} (1 - \alpha_{x_t})^{b_{t,m,x_t}} \Big] \boldsymbol{u}\boldsymbol{u}'$$

Then, for any $\boldsymbol{v} \in \mathcal{R}^{TM}$,

$$\boldsymbol{v}' \nabla^2 g(\boldsymbol{b}(\boldsymbol{x})) \boldsymbol{v} = \Big[p(\boldsymbol{x}) \prod_{t=1}^{T} \prod_{m=1}^{M} (1 - \alpha_{x_t})^{b_{t,m,x_t}} \Big] \|\boldsymbol{u}'\boldsymbol{v}\|_2^2 \geq 0$$

So the Hessian matrix of $g(\boldsymbol{b}(\boldsymbol{x}))$ is positive semidefinite, and $g(\boldsymbol{a}(\boldsymbol{x}))$ is a convex function. Since $f(\boldsymbol{a})$ is the sum of convex functions, it is also a convex function. ■

Also note that the constraints on the decision variables are linear, and the feasible set of decisions is convex and compact. As such, solutions can be obtained by standard convex optimization techniques, but the number of decision variables is large. Alternatively, one can use specialized algorithms such as the nonlinear auction techniques of [Bangla and Castañón, 2013] that exploit the inherent bipartite nature of the problem.

Instead, we use a variant of the integer FAB algorithm in Section 4.3.1 to solve the relaxed problem (4.16)-(4.19). The relaxed FAB algorithm also consists of forward and backward passes. Instead of solving the integer subproblems $IP(t)$, we now solve the relaxed subproblems $RP(t)$ in the forward and backward passes of the relaxed FAB algorithm.

We now show how to solve the relaxed subproblems $RP(t)$. The relaxed subproblems $RP(t)$ are to minimize a convex and continuously differentiable objective function over a convex set formed by linear constraints. We first show how to solve the single-agent relaxed subproblems, then generalize it to the multi-agent case via coordinate descent method.

Since there is only one agent, we can assume that it can search all locations. The single-agent relaxed subproblem is then

$$\begin{aligned} \underset{\boldsymbol{b}_t}{\text{minimize}} \quad & \sum_{k=1}^{K} q_{k,0}(1-\alpha_k)^{b_{t,1,k}} \\ \text{subject to} \quad & \sum_{k=1}^{K} b_{t,1,k} = N_1 \\ & b_{t,1,k} \geq 0, \quad \forall k \end{aligned}$$

The Lagrangian function is

$$L_t(\boldsymbol{b}_t, \lambda) = \sum_{k=1}^{K} q_{k,0}(1-\alpha_k)^{b_{t,1,k}} + \lambda(\sum_{k=1}^{K} b_{t,1,k} - N_1)$$

If $\boldsymbol{b}_t^*$ is a global minimum, then there exists a scalar λ^* such that

$$\boldsymbol{b}_t^* \in \arg\min_{\boldsymbol{b}_t \geq 0} L_t(\boldsymbol{b}_t, \lambda^*)$$

The optimality conditions for an orthant constraint yield

$$q_{k,0} \ln(1-\alpha_k)(1-\alpha_k)^{b_{t,1,k}^*} + \lambda^* \geq 0, \;\; k = 1, \ldots, K$$

$$q_{k,0}\ln(1-\alpha_k)(1-\alpha_k)^{b^*_{t,1,k}}+\lambda^*=0,\ \text{if } b^*_{t,1,k}>0$$

Thus, the single-agent relaxed subproblem can be solved using a scalar optimization technique to find λ^* and the corresponding $\boldsymbol{b}_t^*$ that satisfy the feasibility constraints and the optimality conditions.

For multiple agents with limited accessibilities, the relaxed subproblem $RP(t)$ is

$$\begin{aligned}
\underset{\boldsymbol{b}_t}{\text{minimize}}\ \ & f_t(\boldsymbol{b}_t)=\sum_{\boldsymbol{x}}p(\boldsymbol{x})\prod_{t=1}^{T}(1-\alpha_{t,x_t})^{\sum_{m=1}^{M}b_{t,m,x_t}}\\
\text{subject to}\ & \sum_{k:(m,k)\in\mathcal{A}}b_{t,m,k}=N_m,\ \ \forall m\\
& b_{t,m,k}\geq 0,\ \ \forall t,\forall(m,k)\in\mathcal{A}_t\\
& b_{t,m,k}=0,\ \ \forall t,\forall(m,k)\notin\mathcal{A}_t
\end{aligned}$$

The decision variables $\boldsymbol{b}_t$ can be divided into M blocks. The $\hat{m}$-th block $\boldsymbol{b}_t(\hat{m})=\{b_{t,\hat{m},1},\ldots,b_{t,\hat{m},K}\}$ corresponds to the $\hat{m}$-th agent. If we fix $M-1$ blocks of a feasible $\boldsymbol{b}_t$ and only optimize over its $\hat{m}$-th block, *the relaxed subproblem with respect to the $\hat{m}$-th block* is

$$\begin{aligned}
\underset{\boldsymbol{b}_t(\hat{m})}{\text{minimize}}\ \ & \sum_{k:(\hat{m},k)\in\mathcal{A}}\hat{q}_{k,0}(1-\alpha_k)^{b_{t,\hat{m},k}}\\
\text{subject to}\ & \sum_{k:(\hat{m},k)\in\mathcal{A}}b_{t,\hat{m},k}=N_{\hat{m}}\\
& b_{t,\hat{m},k}\geq 0,\ \ \forall k
\end{aligned}$$

where $\hat{q}_{k,0}=q_{k,0}(1-\alpha_k)^{\sum_{m\neq\hat{m}:(m,k)\in\mathcal{A}}b_{t,m,k}}$ are constants. These block-wise subproblems can be solved as single-agent relaxed subproblems.

By applying the block coordinate descent method, we can then solve the multi-

agent sparse relaxed subproblems $RP(t)$ to optimality.

Theorem 4.3.3. *The block coordinate descent method finds an optimal solution of the multi-agent relaxed subproblems $RP(t)$.*

Proof. The objective function $f_t(\boldsymbol{b}_t)$ is continuously differentiable and convex, minimizing over some $\boldsymbol{b}_t(\hat{m})$ at each step in its iterations. The feasible sets of $\boldsymbol{b}_t(\hat{m})$ are compact and convex for all $\hat{m}$, as is the feasible set of $\boldsymbol{b}_t$. By Proposition 6 of [Grippo and Sciandrone, 2000], the limit point of the block coordinate descent method is a local minimum of $f_t(\boldsymbol{b}_t)$, which is also a global minimum due to the convexity of $f_t(\boldsymbol{b}_t)$. ■

In Section 4.5.3, we compare the block coordinate descent method with the constrained nonlinear optimization algorithms available in `scipy.optimize`, a Python optimization package. Results in Table 4.7 show that our block coordinate descent method is much faster than the general algorithms.

We next show that the relaxed FAB algorithm finds an optimal solution of the entire relaxed problem:

Theorem 4.3.4 (Global Optimality of FAB Algorithm for the Relaxed Problem)**.** *The FAB algorithm for continuous search effort converges to a global optimum of the relaxed problem.*

Proof. The FAB algorithm is a block coordinate descent algorithm with a convex, continuously differentiable objective function $f(\boldsymbol{b})$, minimizing over some $\boldsymbol{b}_t$ at each step in its iterations. The feasible sets of $\boldsymbol{b}_t$ are compact and convex, as is the feasible set of $\boldsymbol{b}$. By Proposition 6 of [Grippo and Sciandrone, 2000], the limit point of the block coordinate descent method is a local minimum of $f(\boldsymbol{b})$, which is also a global minimum because $f(\boldsymbol{b})$ is convex. ■

4.3.3 A Two-Stage FAB Algorithm

To solve the integer allocation problem, we propose the use of a two-stage FAB algorithm. In the first stage, we solve the relaxed problem to optimality, using the FAB algorithm for continuous search effort. In the second stage, we randomly sample

a set of initial sensor allocation decisions using the optimal solution for the relaxed problem as probabilities. This sampling is done multiple times, to generate multiple integer sensor allocations as starting allocations for the integer FAB algorithm. For each of these starting allocations, we use the integer FAB algorithm to converge to an improved integer solution, corresponding to a block-coordinate local minimum. For the final sensor allocations, we select the local minimum solution with the best value.

Two-Stage FAB Algorithm

Input: $\{\alpha_k, k = 1, \ldots, K\}$, $\boldsymbol{\pi}_1$, N_{rep}.

Output: $\boldsymbol{b}$.

Initialization:

Let $\boldsymbol{q}_R(1) = \boldsymbol{\pi}_1$. Let $\boldsymbol{q}_L(t) = \mathbf{1}$ for all t.

Stage I:

Step 1 (Relaxed Forward Pass) Starting from $t = 1$ to $T - 1$:

1. Solve the relaxed subproblem $RP(t)$ in (4.20)–(4.23) to obtain $\boldsymbol{b}_t$.
2. Update $\boldsymbol{q}_R(t+1) = PI_t(\boldsymbol{b}_t)\boldsymbol{q}_R(t)$.

Step 2 (Relaxed Backward Pass) Starting from $t = T$ to 2:

1. Solve the relaxed subproblem $RP(t)$ in (4.20)–(4.23) to obtain $\boldsymbol{b}_t$.
2. Update $\boldsymbol{q}_L(t-1) = P^T I_t(\boldsymbol{b}_t)\boldsymbol{q}_L(t)$.

Step 3 (Repeat Relaxed Forward-Backward Passes) Repeat **Step 1–2** in Stage I until it converges, i.e., after one cycle of iterations $f(\boldsymbol{b})$ does not improve the solution significantly.

Stage II:

Step 4: Normalize $\bar{\boldsymbol{b}}$ obtained from Stage I to one. Randomly sample integer-valued $\boldsymbol{b}$ from a multinomial distribution whose probability mass function is the normalized $\bar{\boldsymbol{b}}$ for N_{rep} times. Compute compute $f(\boldsymbol{b})$. For each $\boldsymbol{b}$, compute $\boldsymbol{q}_L(t)$ for all t and let $\boldsymbol{q}_R(1) = \boldsymbol{\pi}_1$, and do **Step 5–7**.

Step 5 (Integer Forward Pass) Starting from $t = 1$ to $T - 1$:

1. Solve the integer subproblem $IP(t)$ in (4.12)–(4.15) to obtain $\boldsymbol{b}_t$.
2. Update $\boldsymbol{q}_R(t+1) = PI_t(\boldsymbol{b}_t)\boldsymbol{q}_R(t)$.

Step 6 (Integer Backward Pass) Starting from $t = T$ to 2:

1. Solve the integer subproblem $IP(t)$ in (4.12)–(4.15) to obtain $\boldsymbol{b}_t$.
2. Update $\boldsymbol{q}_L(t-1) = P^T I_t(\boldsymbol{b}_t)\boldsymbol{q}_L(t)$.

Step 7 (Repeat Integer Forward-Backward Passes): Repeat **Step 5–6** in Stage II until convergence, i.e., after one cycle of iterations $f(\boldsymbol{b})$ does not improve.

Step 8: Return the best one among N_{rep} solutions.

4.4 Heterogeneous Agent Search

When the homogeneous agent assumption of Section 4.3 is not satisfied, the integer subproblems are NP-hard, and the FAB algorithm becomes impractical to implement exactly. Note that the relaxed version of these multi-agent search problems will still

be convex, and an optimal solution can be obtained efficiently using the fractional FAB algorithm. However, we no longer have an efficient integer FAB algorithm to improve the solutions. Instead, we establish that the optimization problem of (4.2) - (4.5) (restated here):

$$\underset{\boldsymbol{b}}{\text{minimize}} \;\; f(\boldsymbol{b}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) \prod_{t=1}^{T} \prod_{m=1}^{M} (1 - \alpha_{t,m,x_t})^{b_{t,m,x_t}} \tag{4.2}$$

$$\text{subject to} \sum_{k:(m,k)\in\mathcal{A}_t} b_{t,m,k} \leq N_m, \;\; \forall t, m \tag{4.3$'$}$$

$$b_{t,m,k} \in \{0, \ldots, N_m\}, \;\; \forall t, \forall (m,k) \in \mathcal{A}_t \tag{4.4}$$

$$b_{t,m,k} = 0, \;\; \forall t, \forall (m,k) \notin \mathcal{A}_t \tag{4.5}$$

can be reduced to a problem of maximizing a monotone submodular function with matroid constraints. Note that due to the monotonicity of the objective function, the inequality constraint (4.3$'$) is equivalent to the equality constraint (4.3). We present two greedy algorithms with guaranteed performance for the approximate solution of this problem and a lower bound of the optimal objective value.

4.4.1 Set Based Formulation

Without loss of generality, we can assume that $N_m = 1$ for each agent m at any time period. This is because if $N_m > 1$, we can create N_m copies of the agent with the same visibility $\mathcal{A}_t$ at each time t, resulting in an equivalent problem with $N = \sum_{m=1}^{M} N_m$ micro agents, each of which has a budget of one unit of search effort.

Let $\mathcal{E}_{t,m} = \{(t,m,k) | (m,k) \in \mathcal{A}_t\}$ be the set of all feasible assignments for each micro agent m at time t, and let $\mathcal{E} = \cup_{t=1}^{T} \cup_{m=1}^{M} \mathcal{E}_{t,m}$. Note that $\mathcal{E}_{t,m} \cap \mathcal{E}_{t',m'} = \emptyset$ if $t \neq t'$ or $m \neq m'$, and $\{\mathcal{E}_{t,m}\}$ is a partition of $\mathcal{E}$. We transform the problem from minimizing the probability of no detection to maximizing the probability of a detection, and reformulate it in terms of set variables, as:

$$\underset{S}{\text{maximize}} \;\; F(S) = -\sum_{\boldsymbol{x}} p(\boldsymbol{x}) \prod_{t=1}^{T} \prod_{m=1}^{M} \prod_{(t,m,x_t)\in S} (1-\alpha_{t,m,x_t}) \tag{4.24}$$

$$\text{subject to} \;\; |S \cap \mathcal{E}_{t,m}| \leq 1, \;\; \forall t, m \tag{4.25}$$

$$S \subset \mathcal{E} \tag{4.26}$$

The correspondence between the set-based formulation and (4.2) - (4.5) is as follows: $(t, m, k) \in S$ is equivalent to $b_{t,m,k} = 1$. The cardinality constraints (4.25) restrict each agent to search at most one location at each time, which is the inequality version of (4.3) when $N_m = 1$. Switching to an inequality constraint is not a restriction, because the monotonicity property that will be shown subsequently implies that the optimal solution will be satisfied the constraint with equality. A feasible set S can be interpreted as a feasible assignment of agents to locations for multiple time periods. It is easy to see that the maximum cardinality of a feasible set is TM, and an optimal set has cardinality TM.

Let the collection of feasible sets (known as the independent sets) satisfying (4.25)–(4.26) be $\mathcal{I}$. We show that the objective function $F(S)$ is increasing and submodular, and the pair $(\mathcal{E}, \mathcal{I})$ is a matroid.

Lemma 4.4.1. *The objective function $F(S)$ is increasing and submodular in S.*

Proof. Consider two feasible assignments S, S' where there exists $(t', m', k') \in \mathcal{E}$ such that $(t', m', k') \notin S$ and $S' = S \cup \{(t', m', k')\}$. Then, from (4.25) we have $S \cap \mathcal{E}_{t',m'} = \emptyset$ and

$$\begin{aligned}
& F(S') \\
=& -\sum_{\boldsymbol{x}: x_{t'} \neq k'} p(\boldsymbol{x}) \prod_{t,m} \prod_{(t,m,x_t)\in S'} (1-\alpha_{t,m,x_t}) \\
& -\sum_{\boldsymbol{x}: x_{t'} = k'} p(\boldsymbol{x}) \Big[\prod_{t\neq t' \text{ or } m\neq m'} \prod_{(t,m,x_t)\in S'} (1-\alpha_{t,m,x_t}) \Big] (1-\alpha_{t',m',k'}) \\
>& -\sum_{\boldsymbol{x}: x_{t'} \neq k'} p(\boldsymbol{x}) \prod_{t,m} \prod_{(t,m,x_t)\in S} (1-\alpha_{t,m,x_t})
\end{aligned}$$

$$
\begin{aligned}
&- \sum_{\boldsymbol{x}: x_{t'}=k'} p(\boldsymbol{x}) \Big[\prod_{t \neq t' \text{ or } m \neq m'} \prod_{(t,m,x_t) \in S} (1 - \alpha_{t,m,x_t}) \Big] \\
=&F(S)
\end{aligned}
$$

Thus, $F(S)$ is increasing in S.

Next, consider $S \subset S'$ and $e = (t', m', k') \in \mathcal{E}$ such that $e \notin S'$, we have

$$
\begin{aligned}
&F(S \cup \{e\}) - F(S) \\
=& \sum_{\boldsymbol{x}: x_{t'}=k'} p(\boldsymbol{x}) \Big[\prod_{t \neq t' \text{ or } m \neq m'} \prod_{(t,m,x_t) \in S} (1 - \alpha_{t,m,x_t}) \Big] \alpha_{t',m',k'} \\
>& \sum_{\boldsymbol{x}: x_{t'}=k'} p(\boldsymbol{x}) \Big[\prod_{t \neq t' \text{ or } m \neq m'} \prod_{(t,m,x_t) \in S'} (1 - \alpha_{t,m,x_t}) \Big] \alpha_{t',m',k'} \\
=&F(S' \cup \{e\}) - F(S')
\end{aligned}
$$

Thus, $F(S)$ has the diminishing return property and is submodular in S [Fisher et al., 1978]. ■

Lemma 4.4.2. *The pair $(\mathcal{E}, \mathcal{I})$ is a matroid.*

Proof. A pair $(\mathcal{E}, \mathcal{I})$ where $\mathcal{E}$ is a finite set and $\mathcal{I}$ is a family of subsets of $\mathcal{E}$ is a matroid if [Welsh, 1976]:

1. $\emptyset \in \mathcal{I}$.
2. (Hereditary Property) For each $S' \subset S \subset \mathcal{E}$, if $S \in \mathcal{E}$ then $S' \in \mathcal{E}$.
3. (Augmentation Property) If $S \in \mathcal{I}$, $S' \in \mathcal{I}$ and $|S'| < |S|$, then there exists $e \in S \setminus S'$ such that $S' \cup \{e\} \in \mathcal{I}$.

Given a feasible assignment $S \in \mathcal{I}$, any $S' \subset S$ is also a feasible assignment, thus the hereditary property of independent sets required for a matroid is satisfied.

Furthermore, given two feasible assignments S and S' such that $|S'| > |S|$. Then, from (4.25) there must exist $e = (t', m', k') \in S'$ and $(t', m', k) \notin S$ for all k, implying that $S \cup \{e\}$ is also a feasible assignment. Therefore, the augmentation property of a matroid is also satisfied. ■

4.4.2 A Greedy Algorithm

We propose a greedy algorithm that exploits the structure of the problem and has a performance guarantee. It starts with no agent assignments (i.e., $S = \emptyset$), assigns search effort sequentially in time, and within each time, sequentially in agents. Within each time t, for each agent m, it adds an element $e^* \in \mathcal{E}_{t,m}$ to the set S which maximizes the marginal gain $F(S \cup \{e\}) - F(S)$. However, computation of $F(S \cup \{e\}) - F(S)$ would be very inefficient, even for Markov trajectories, as this would require updating the recursions (4.6)-(4.7) for every time t after each additional assignment.

Therefore, before presenting the greedy algorithm, we first show how to compute the marginal gain efficiently. Assume that we have assigned agents for the first $t-1$ time periods. For time period t, we have assigned the first $m-1$ agents. We have not assigned the rest. Denote the set representation for current agent assignments by S, and let S correspond to $\boldsymbol{b} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_T]$. Now we want to assign agent m for time t. We want to find k^* such that $\mathbf{1}^* = \{t, m, k^*\} \in \mathcal{E}_{t,m}$ maximizes the marginal gain $F(S \cup \{e\}) - F(S)$. The marginal gain can be easily represented as shown in Lemma 4.4.3.

Lemma 4.4.3. *Let S and $\boldsymbol{b}$ be the current agent assignments. Assume that for some t and m,*

$$|S \cap \mathcal{E}_{s,i}| = \begin{cases} 1, & \text{if } s < t \text{ or } s = t, i < m \\ 0, & \text{if } s = t, i \geq m \text{ or } s > t \end{cases}$$

and equivalently,

$$\sum_{j=1}^{K} b_{s,i,j} = \begin{cases} 1, & \text{if } s < t \text{ or } s = t, i < m \\ 0, & \text{if } s = t, i \geq m \text{ or } s > t \end{cases} \quad \text{and } b_{s,i,j} \in \{0,1\},\ \forall s, i, j$$

For a given k, let $e = \{t, m, k\} \in \mathcal{E}_{t,m}$. Then

$$F(S \cup \{e\}) - F(S) = \boldsymbol{q}_R(t)_{(k)} I_t(\boldsymbol{b}_t)_{(k,k)} \alpha_{t,m,k}$$

where $\boldsymbol{q}_R(t)_{(k)}$ is the k-th element of column vector $\boldsymbol{q}_R(t)$ and $I_t(\boldsymbol{b}_t)_{(k,k)}$ is the (k,k)-th entry of square matrix $I_t(\boldsymbol{b}_t)$ respectively.

Proof. Since

$$\sum_{j=1}^{K} b_{s,i,j} = \begin{cases} 1, & \text{if } s < t \text{ or } s = t, i < m \\ 0, & \text{if } s = t, i \geq m \text{ or } s > t \end{cases}$$

and $\mathbf{1}^T P = \mathbf{1}^T$, we have

$$\begin{aligned} & F(S) = -f(\boldsymbol{b}) \\ =& -\mathbf{1}^T \Big[\prod_{s=t+1}^{T} PI_s(\boldsymbol{b}_s) \Big] PI_t(\boldsymbol{b}_t) \Big[\prod_{s=1}^{t-1} PI_s(\boldsymbol{b}_s) \Big] \boldsymbol{\pi}_1 \\ =& -\mathbf{1}^T PI_t(\boldsymbol{b}_t) \Big[\prod_{s=1}^{t-1} PI_s(\boldsymbol{b}_s) \Big] \boldsymbol{\pi}_1 \\ =& -\mathbf{1}^T I_t(\boldsymbol{b}_t) \boldsymbol{q}_R(\boldsymbol{b}_t) \\ =& -\sum_{j=1}^{K} \boldsymbol{q}_R(\boldsymbol{b}_t)_{(j)} \prod_{i=1}^{M} (1 - \alpha_{t,i,j})^{b_{t,i,j}} \\ =& -\sum_{j=1}^{K} \boldsymbol{q}_R(\boldsymbol{b}_t)_{(j)} \prod_{i=1}^{m-1} (1 - \alpha_{t,i,j})^{b_{t,i,j}} \end{aligned}$$

Assume that adding $e = \{t, m, k\}$ to S is equivalent to letting $b_{t,m,k} = 1$ in $\boldsymbol{b}$ to form $\boldsymbol{b}_{new}$. Therefore,

$$\begin{aligned} & F(S \cup \{e\}) - F(S) = -f(\boldsymbol{b}_{new}) + f(\boldsymbol{b}) \\ =& -\sum_{j \neq k} \boldsymbol{q}_R(\boldsymbol{b}_t)_{(j)} \prod_{i=1}^{m-1} (1 - \alpha_{t,i,j})^{b_{t,i,j}} \\ & - \boldsymbol{q}_R(\boldsymbol{b}_t)_{(k)} \Big[\prod_{i=1}^{m-1} (1 - \alpha_{t,i,k})^{b_{t,i,k}} \Big] (1 - \alpha_{t,m,k}) \\ & - (-\sum_{j=1}^{K} \boldsymbol{q}_R(\hat{\boldsymbol{b}}_t)_{(j)} \prod_{i=1}^{m-1} (1 - \alpha_{t,i,j})^{b_{t,i,j}}) \\ =& \boldsymbol{q}_R(\boldsymbol{b}_t)_{(k)} \Big[\prod_{i=1}^{m-1} (1 - \alpha_{t,i,k})^{b_{t,i,k}} \Big] \alpha_{t,m,k} \end{aligned}$$

■

Now we present the greedy algorithm. It iterates through the time periods. For each time period, it assigns the agents one by one. The algorithm is as follows:

- **Initialization:** Let $i = 1$. Let $S_0 = \emptyset$ and $\boldsymbol{b} = 0$. Let $\boldsymbol{q}_R(1) = \boldsymbol{\pi}_1$.
- **Main Loop:** For $t = 1, \ldots, T$:
  - For $m = 1, \ldots, M$:
    - Find $e^* = \arg\max_{e \in \mathcal{E}_{t,m}} \left[F(S_{i-1} \cup \{e\}) - F(S_{i-1})\right]$. This is equivalent to finding

$$k^* = \underset{k:(m,k)\in\mathcal{A}}{\arg\max}\ \boldsymbol{q}_R(t)_{(k)} I_t(\boldsymbol{b}_t)_{(k,k)} \alpha_{t,m,k}$$

      Let $S_i = S_{i-1} \cup \{e^*\}$ and $b_{t,m,k^*} = 1$.
    - Update $I_t(\boldsymbol{b}_t)$ by letting $I_t(\boldsymbol{b}_t)_{(k^*,k^*)} = I_t(\boldsymbol{b}_t)_{(k^*,k^*)}(1 - \alpha_{t,m,k^*})$. If $m = M$, update $\boldsymbol{q}_R(t+1) = P I_t(\boldsymbol{b}_t) \boldsymbol{q}_R(t)$.
    - Increment $i = i + 1$.
- Return the greedy solution $S_{greedy} = S_{TM}$.

The algorithm produces a solution whose value is at least $\frac{1}{2}$ of the optimal value. In fact, this holds true for any greedy-style algorithm.

Theorem 4.4.4. *The value of the solution returned by the greedy algorithm above is at least $\frac{1}{2}$ of the value of the optimal solution S^*.*

More generally, consider an arbitrary ordering $\{\mathcal{E}_i, i = 1, \ldots, TM\}$ of sets $\mathcal{E}_{t,m}$, $\forall t, m$. A greedy-style algorithm which starts with $S_0 = \emptyset$ and adds $e_i^* \in \mathcal{E}_i$ that maximizes the marginal gain $\big(F(S_{i-1} \cap \{e_i\}) - F(S_{i-1})\big)$ to S_{i-1} at each iteration i, $i = 1, \ldots, TM$, can find a greedy solution with value at least $\frac{1}{2}$ of the optimal value.

Proof. We prove the general result for any greedy algorithm of the style described in the theorem. Denote the solution returned by the greedy algorithm by $S_{greedy} = S_{TM}$. Denote the optimal solution be S^*. Let $T^* = S^* \backslash S_{greedy}$ and denote the elements in

T^* by $\{e_1^*, \ldots, e_{|T^*|}^*\}$. Let $T_i = (S^* \cap \mathcal{E}_i) \backslash S_{greedy}$, $i = 1, \ldots, TM$. Since $\{\mathcal{E}_i\}$ is a partition of $\mathcal{E}$, $\{T_i\}$ is a partition of T^*. Then,

$$
\begin{aligned}
F(S^*) \leq & F(S_{greedy} \cup S^*) \\
= & F(S_{greedy} \cup T^*) \\
= & F(S_{greedy}) + \sum_{j=1}^{|T^*|} [F(S_{greedy} \cup \{e_1^*, \ldots, e_j^*\}) - F(S_{greedy} \cup \{e_1^*, \ldots, e_{j-1}^*\})] \\
\leq & F(S_{greedy}) + \sum_{j=1}^{|T^*|} [F(S_{greedy} \cup \{e_j^*\}) - F(S_{greedy})] \qquad (4.27) \\
= & F(S_{greedy}) + \sum_{e \in T^*} [F(S_{greedy} \cup \{e\}) - F(S_{greedy})] \\
= & F(S_{greedy}) + \sum_{i=1}^{TM} \sum_{e \in T_i} [F(S_{greedy} \cup \{e\}) - F(S_{greedy})] \\
\leq & F(S_{greedy}) + \sum_{i=1}^{TM} \sum_{e \in S^* \cap \mathcal{E}_i} [F(S_{greedy} \cup \{e\}) - F(S_{greedy})] \\
\leq & F(S_{greedy}) + \sum_{i=1}^{TM} \sum_{e \in S^* \cap \mathcal{E}_i} [F(S_{i-1} \cup \{e\}) - F(S_{i-1})] \qquad (4.28) \\
\leq & F(S_{greedy}) + \sum_{i=1}^{TM} [F(S_i) - F(S_{i-1})] \qquad (4.29) \\
= & 2F(S_{greedy})
\end{aligned}
$$

Inequalities (4.27) and (4.28) are due to the submodularity of $F(S)$. Inequality (4.29) holds since the greedy algorithm finds the best element in $\mathcal{E}_i$ that augments the previous set S_{i-1} and $|S^* \cap \mathcal{E}_{\langle m,j \rangle}| = 1$. ■

Note that the greedy algorithm above also achieves a $\frac{1}{2}$ approximation ratio for the special case where the probabilities of detection only depend on locations.

4.4.3 Another Greedy Algorithm

We present another greedy algorithm which allocates one agent for all the time periods, then allocates another agent. The algorithm iterates in opposite directions for

neighboring agents, and updates the temporary values involved in a forward-backward manner.

We have the following lemma for computing the marginal gain.

Lemma 4.4.5. *Let S and $\boldsymbol{b}$ be the current agent assignments. Assume that for some t and m,*

$$|S \cap \mathcal{E}_{s,i}| = \begin{cases} 1, & \text{if } i < m \text{ or } i = m, s < t \\ 0, & \text{if } i = m, s \geq t \text{ or } i > m \end{cases}$$

and equivalently,

$$\sum_{k=1}^{K} b_{s,i,j} = \begin{cases} 1, & \text{if } i < m \text{ or } i = m, s < t \\ 0, & \text{if } i = m, s \geq t \text{ or } i > m \end{cases} \quad \text{and } b_{s,i,j} \in \{0,1\},\ \forall s, \forall i, \forall j$$

For a given k, let $e = \{t, m, k\} \in \mathcal{E}_{t,m}$. Then

$$F(S \cup \{e\}) - F(S) = \boldsymbol{q}_L(t)_{(k)} \boldsymbol{q}_R(t)_{(k)} I_t(\boldsymbol{b}_t)_{(k,k)} \alpha_{t,m,k}$$

where $\boldsymbol{q}_L(t)_{(k)}$, $\boldsymbol{q}_R(t)_{(k)}$, and $I_t(\boldsymbol{b}_t)_{(k,k)}$ are the k-th element of column vector $\boldsymbol{q}_L(t)$, the k-th element of column vector $\boldsymbol{q}_R(t)$, and the (k,k)-th entry of square matrix $I_t(\boldsymbol{b}_t)$ respectively.

Proof. Assume that adding $e = \{t, m, k\}$ to S is equivalent to letting $b_{t,m,k} = 1$ in $\boldsymbol{b}$ to form $\boldsymbol{b}_{new}$. Then,

$$\begin{aligned} & F(S \cup \{e\}) - F(S) = F(\boldsymbol{b}_{new}) - F(\boldsymbol{b}) \\ =& -\sum_{j \neq k} \boldsymbol{q}_L(\boldsymbol{b}_t)_{(j)} \boldsymbol{q}_R(\boldsymbol{b}_t)_{(j)} I_t(\boldsymbol{b}_t)_{(j,j)} \\ & - \boldsymbol{q}_L(\boldsymbol{b}_t)_{(k)} \boldsymbol{q}_R(\boldsymbol{b}_t)_{(k)} I_t(\boldsymbol{b}_t)_{(k,k)} (1 - \alpha_{t,m,k}) \\ & - \Big(-\sum_{j=1}^{K} \boldsymbol{q}_L(\boldsymbol{b}_t)_{(j)} \boldsymbol{q}_R(\boldsymbol{b}_t)_{(j)} I_t(\boldsymbol{b}_t)_{(j,j)} \Big) \\ =& \boldsymbol{q}_L(\boldsymbol{b}_t)_{(k)} \boldsymbol{q}_R(\boldsymbol{b}_t)_{(k)} I_t(\boldsymbol{b}_t)_{(k,k)} \alpha_{t,m,k} \end{aligned}$$

■

The algorithm goes as follows:

- **Initialization:** Let $i = 0$. Let $S_0 = \emptyset$ and $\boldsymbol{b} = 0$. Let $\boldsymbol{q}_R(1) = \mathbf{1}$. Let $\boldsymbol{q}_L(t) = \boldsymbol{\pi}_1$ for all t. Let $I_t(\boldsymbol{b}_t)$ be an identity matrix for all t.

- **Main Loop:** For $m = 1, \ldots, M$:

  - If m is odd, iterate from $t = 1$ to $t = T$. Otherwise, iterate from $t = T$ to $t = 1$:

    * Find $e^* = \arg\max_{e \in \mathcal{E}_{t,m}} \left[F(S_{i-1} \cup \{e\}) - F(S_{i-1})\right]$. This is equivalent to finding

$$k^* = \underset{k:(m,k)\in\mathcal{A}}{\arg\max}\, \boldsymbol{q}_L(t)_{(k)} \boldsymbol{q}_R(t)_{(k)} I_t(\boldsymbol{b}_t)_{(k,k)} \alpha_{t,m,k}$$

      Let $S_i = S_{i-1} \cup \{e^*\}$ and $b_{t,m,k^*} = 1$.

    * Update $I_t(\boldsymbol{b}_t)$ by letting $I_t(\boldsymbol{b}_t)_{(k^*,k^*)} = I_t(\boldsymbol{b}_t)_{(k^*,k^*)}(1 - \alpha_{t,m,k^*})$. If m is odd and $t < T$, update $\boldsymbol{q}_R(t+1) = P I_t(t) \boldsymbol{q}_R(t)$. If m is even and $t > 1$, update $\boldsymbol{q}_L(t-1) = P' I_t(\boldsymbol{b}_t) \boldsymbol{q}_L(t)$.

    * Increment $i = i + 1$.

- Return the greedy solution $S_{greedy} = S_{TM}$.

By Theorem 4.4.4, the algorithm finds a value that is at least $\frac{1}{2}$ of the optimal value.

4.4.4 Lower Bound of the Optimal Value

Consider the following nonlinear program which is a continuous relaxation of the heterogeneous agent search program (4.2) - (4.5):

$$\underset{\boldsymbol{b}}{\text{minimize}} \;\; f(\boldsymbol{b}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) \prod_{t=1}^{T} \prod_{m=1}^{M} (1 - \alpha_{t,m,x_t})^{b_{t,m,x_t}} \tag{4.30}$$

$$\text{subject to} \sum_{k:(m,k)\in\mathcal{A}_t} b_{t,m,k} = N_m, \;\; \forall t, m \tag{4.31}$$

$$b_{t,m,k} \in [0, N_m],\ \forall t, \forall (m,k) \in \mathcal{A}_t \tag{4.32}$$

$$b_{t,m,k} = 0,\ \forall t, \forall (m,k) \notin \mathcal{A}_t \tag{4.33}$$

It is easy to see that the optimal solution of the realxed problem (4.30)–(4.33) is a lower bound of the optimal solution of the integer nonlinear problem (4.2)–(4.5). The objective function $f(\boldsymbol{b})$ in (4.30) is a continuously differentiable convex function, which can be shown by replacing the probabilies of detection $\alpha_{t,k}$ with $\alpha_{t,m,k}$ in the proof of Lemma 4.4.1.

The relaxed FAB algorithm developed in Section 4.3.2 finds an optimal solution of the relaxed heterogeneous agent search problem (4.30)–(4.33). This is because the relaxed FAB algorithm is a block coordinate descent algorithm with a continuously differentiable convex objective function $f(\boldsymbol{b})$, minimizing over some $\boldsymbol{b}_t$ at each iteration. The feasible sets of $\boldsymbol{b}_t$ are compact and convex, as is the feasible set of $\boldsymbol{b}$. By Proposition 6 of [Grippo and Sciandrone, 2000] and the convexity of $f(\boldsymbol{b})$, the limit point of the relaxed FAB algorithm is a global minimum.

4.5 Experiments

4.5.1 Experiments for Homogeneous Agent Search

In this section, we do experiments to demonstrate the performance of our two-stage FAB algorithm for the problem of sparse search of Markovian moving object using multiple agents with location-dependent probabilities of detection. All experiments are conducted using Python 3 on a laptop computer with Intel i7-4600M processor and 8GB RAM.

First, we show that our algorithm finds solutions that are close to the best potential solution. We perform the same experiments on Algorithm 2 in [Royset and Sato, 2010], referred to as the RS algorithm. The RS algorithm is an exact algorithm

based on the cutting plane method. We use the Gurobi v7.5 Python package to solve the mixed-integer linear problems generated in the algorithm.

The RS algorithm computes a lower bound f_{lo}, which is the best potential objective value. It should be emphasized that this lower bound may not be achieved by any feasible solution. Denote the objective value found by our algorithm by f_{our}. The relative optimality gap is computed as $\delta = \frac{f_{our} - f_{lo}}{f_{lo}}$. The performance of our algorithm can therefore be evaluated by the relative optimality gap. The RS algorithm also maintains an upper bound f_{hi}, which is the best objective value achieved by it so far.

The RS algorithm starts with no agent assignments for all time periods, i.e., $\boldsymbol{b} = 0$. Our experiments show that the RS algorithm converges much slower than our algorithm and the lower bound stays at 0 even after many iterations. To make it converge faster, we initialize the RS algorithm with cuts computed using the N_{rep} solutions obtained from our two-stage FAB algorithm. Thus, the RS algorithm is guaranteed to obtain a search plan that is at least as good as the one obtained by our algorithm.

We randomly generate 5 models for 3 search agents, 225 locations, and 10 time periods. For each agent m and each location k, we let the probability of $(m, k) \in \mathcal{A}$ be 0.6 (unless otherwise specified, the probability of generating arcs is also 0.6 for the models hereafter). We also made sure that each agent can access at least one location and each location is accessible to at least one agent. For each model, we sample $N_{rep} = 20$ initial starting points for stage II in our algorithm. We run the RS algorithm for 900 seconds to obtain the lower bounds and compute the relative optimality gaps of our solutions.

The results are shown in Table 4.1. For all models, our algorithm has found solutions of relative optimality gap within 5% in no more than 8 seconds. Besides, the solutions found by our algorithm were not improved by the RS algorithm.

Table 4.1: Relative optimality gaps and run times by our algorithm. The models have 3 agents, 225 locations and 10 time periods. Number of random initial points for stage II in our algorithm is $N_{rep} = 20$. The RS algorithm is initialized with our solution instead of $\boldsymbol{b} = 0$. Lower bounds f_{lo} are obtained by running the RS algorithm for 900 seconds. Relative optimality gaps are computed as $\delta = \frac{f_{our} - f_{lo}}{f_{lo}}$.

Model No.	Our value f_{our}	RS lower bound f_{lo}	Relative opt. gap δ	Our run time	RS upper bound f_{hi}
1	0.8564	0.8181	4.69%	6.35s	0.8564
2	0.8588	0.8186	4.91%	6.60s	0.8588
3	0.8585	0.8224	4.40%	7.15s	0.8585
4	0.8613	0.8236	4.57%	6.60s	0.8613
5	0.8598	0.8211	4.71%	6.93s	0.8598

Second, we compare four variants of our two-stage FAB algorithm. The 1st variant, *one-stage direct*, has only stage II and is initialized with $\boldsymbol{b} = 0$. It is equivalent to the algorithm presented in Section 4.3.1. The 2nd variant, *two-stage direct*, has both stages, and stage II uses the fractional point found by stage I as the initial point. The 3rd variant, *one-stage randomized*, has only stage II. The initial points for stage II are randomly sampled from a uniform distribution. The last variant, *two-stage randomized*, is our two-stage FAB algorithm presented in Section 4.3.2. In addition, we compare with the greedy myopic algorithm. It starts with no agent assignments. Then from time period 1 to T, it chooses the agent assignment for that time period that minimizes the objective value. The myopic algorithm is in fact equivalent to one sweep of the forward recursion in the one-stage direct FAB algorithm and therefore is guaranteed to find a worse solution than the one-stage direct FAB algorithm.

We randomly generate 5 models for 10 search agents, 100 locations, and 15 time periods. In the two randomized variants, we randomly generate 20 initial points for stage II. The results are shown in Table 4.2. For all the models, the two-stage FAB algorithm in Section 4.3.2 finds better objective values than its other three variants and the myopic algorithm.

Next, we study how our two-stage FAB algorithm scales with problem size. We

Table 4.2: Comparison of four variants of our algorithm and the myopic algorithm. The models have 10 agents, 100 locations and 15 time periods. Number of random initial points for stage II in the randomized variants is $N_{rep} = 20$. The asterisk (*) indicates the best value found.

Model No.	Myopic	One-stage direct	One-stage randomized	Two-stage direct	Two-stage randomized
1	0.17740	0.17732	0.17728	0.17731	0.17727*
2	0.18678	0.18646	0.18648	0.18644	0.18643*
3	0.19541	0.19496	0.19491	0.19492	0.19490*
4	0.19576	0.19547	0.19540	0.19541	0.19537*
5	0.19643	0.19546	0.19540	0.19539	0.19538*

run our two-stage FAB algorithm with $N_{rep} = 20$ random initial points. We also run the RS algorithm to obtain lower bounds and relative optimality gaps. As before, we initialize the RS algorithm with cuts computed using our solutions and run it for 900 seconds.

We do three experiments. The first experiment fixes the number of agents to be 10 and the number of locations to be 100, and generates 5 models by varying the time horizon from 5 to 25. The results are in Table 4.3. The second experiment fixes the number of locations to be 100 and the time horizon to be 10, and generates 5 models by varying the number of agents from 4 to 20. The results are in Table 4.4. The third experiment fixes the number of locations to be 100 and the time horizon to be 10, and generates 5 models by varying the number of agents from 4 to 20. The results are in Table 4.5.

The results show that our algorithm scales well with respect to different problem sizes. We observe that the run time for different numbers of locations and fixed number of agents and time horizon is approximately constant, and the run time increases approximately linearly as time horizon or number of agents grows. Besides, our algorithm has found solutions that could not be further improved by the RS algorithm.

Table 4.3: Objective values and run times by our two-stage FAB algorithm for different time horizons. The model has 10 agents and 100 locations. Time horizon ranges from 5 to 25. The RS algorithm is initialized with N_{rep} cuts computed using the solutions obtained from the N_{rep} repetitions of stage II of our algorithm. Lower bounds are obtained by running the RS algorithm for 900 seconds. $N_{rep} = 20$.

Time horizon	Our value f_{our}	RS lower bound f_{lo}	Relative opt. gap δ	Our run time	RS upper bound f_{hi}
5	0.5349	0.4351	22.94%	1.45s	0.5349
10	0.3079	0.0601	412.31%	3.60s	0.3079
15	0.1773	0	N/A	6.22s	0.1773
20	0.1021	0	N/A	8.89s	0.1021
25	0.0588	0	N/A	10.09s	0.0588

Table 4.4: Objective values and run times by our two-stage FAB algorithm for different numbers of agents. The model has 100 locations and 10 time periods. Number of agents ranges from 4 to 20. The RS algorithm is initialized with N_{rep} cuts computed using the solutions obtained from the N_{rep} repetitions of stage II of our algorithm. Lower bounds are obtained by running the RS algorithm for 900 seconds. $N_{rep} = 20$.

Number of agents	Our value f_{our}	RS lower bound f_{lo}	Relative opt. gap δ	Our run time	RS upper bound f_{hi}
4	0.6184	0.5216	18.6%	0.60s	0.6184
8	0.3878	0.1826	112.38%	1.53s	0.3878
12	0.2439	0.0128	1805.47%	4.65s	0.2439
16	0.1515	0	N/A	8.31s	0.1515
20	0.0932	0	N/A	11.60s	0.0932

4.5.2 Experiments for Heterogeneous Agent Search

In this section, we compare different algorithms for the heterogeneous agent search problem (4.2) - (4.5). We generate 5 models to compare the algorithms. Each model has 20 agents, 400 locations and 20 time periods.

We refer to the greedy algorithm in Section 4.4.2 as the *G1 direct* algorithm, and the greedy algorithm in Section 4.4.3 as the *G2 direct* algorithm. We refer to the randomized version of G1 direct (and G2 direct) as *G1 random* (and *G2 random*). The randomized algorithms randomly initialize a set of $\boldsymbol{q}_R(\boldsymbol{b}_1)$, run the main loop to obtain a greedy solution for each $\boldsymbol{q}_R(\boldsymbol{b}_1)$, then keep the best solution. For the G1/G2

Table 4.5: Objective values and run times by our two-stage FAB algorithm for different numbers of locations. The model has 10 agents and 10 time periods. Number of locations ranges from 64 to 361. The RS algorithm is initialized with N_{rep} cuts computed using the solutions obtained from the N_{rep} repetitions of stage II of our algorithm. Lower bounds are obtained by running the RS algorithm for 900 seconds. $N_{rep} = 20$.

Number of locations	Our value f_{our}	RS lower bound f_{lo}	Relative opt. gap δ	Our run time	RS upper bound f_{hi}
64	0.1541	0	N/A	1.50s	0.1541
100	0.3709	0.0601	412.31%	3.60s	0.3079
169	0.5113	0.3733	36.97%	4.12s	0.5113
225	0.6016	0.5094	18.10%	4.21s	0.5094
361	0.7388	0.6929	6.62%	4.51s	0.7388

random algorithms, let the number of initial $\boldsymbol{q}_R(\boldsymbol{b}_1)$ be 20.

We also compare with the RS algorithm (it is initialized with $\boldsymbol{b} = 0$ and runs for 900 seconds) and a purely random approach, which randomly generates 1000 feasible agent assignments $\boldsymbol{b}$ and picks the best value. In addition, we compute the lower bounds on the optimal objective values by solving the relaxed heterogeneous agent search problem (4.30)–(4.33).

The results are in Table 4.6. The values are the final probability of not detecting the objects. All four greedy algorithms have found solutions better than the RS algorithm and the purely random approach. The G2 random algorithm, which assigns one agent for all the time periods followed by assigning another agent for all the time periods, wins all 5 cases. Between the direct and random variants, the random ones have found better solutions.

4.5.3 Experiments for Relaxed Subproblems

We compare our algorithm presented in Appendix B for solving the relaxed subproblem with the SLSQP algorithm provided in `scipy.optimize`, a Python optimization package. The SLSQP algorithm in `scipy.optimize`, which was originally imple-

Table 4.6: Comparison of objective values by different algorithms for sparse search with multiple heterogeneous agents. The models have 20 agents, 400 locations and 20 time periods. Number of random initial $\boldsymbol{q}_R(\boldsymbol{b}_1)$ for G1/G2 random is 20. The RS algorithm is initialized with $\boldsymbol{b} = 0$ and runs for 900 seconds. The purely random approach chooses the best among 1000 random solutions. The asterisk (*) indicates the best value found.

Model No.	G1 direct	G1 random	G2 direct	G2 random	RS	Purely random	Lower bound
1	0.3312	0.3302	0.3314	0.3292*	0.3529	0.4562	0.1820
2	0.3298	0.3324	0.3298	0.3286*	0.3559	0.4568	0.1842
3	0.3291	0.3291	0.3292	0.3275*	0.3546	0.4567	0.1826
4	0.3319	0.3310	0.3318	0.3292*	0.3525	0.4575	0.1788
5	0.3319	0.3310	0.3320	0.3298*	0.3519	0.4539	0.1808

mented by Kraft [Kraft, 1988], is a sequential quadratic programming algorithm.

We generate 5 models. Each model has 10 agents and 100 locations. The run times of two algorithms are given in Table 4.7. It can be seen that our specialized algorithm is two orders of magnitude faster than the SLSQP algorithm for general constrained nonlinear optimization problem.

Table 4.7: Compasion of run times by our algorithm and the SLSQP algorithm in Python `scipy.optimize` package for solving the relaxed subproblems. The models have 10 agents and 100 locations.

Model No.	Our Algorithm Run Time	`scipy.optimize` SLSQP Run Time
1	0.15s	24.47s
2	0.22s	20.43s
3	0.18s	31.57s
4	0.17s	22.47s
5	0.23s	21.62s

Chapter 5

Multi-Object Graph Search with Motion and Switching Cost Constraints

In this chapter, we develop a different class of fast algorithms for the problem of multi-object graph search with motion and switching cost constraints, which is formulated as a single orienteering problem (SOP) or a team orienteering problem (TOP) depending on the number of search agents, based on a decomposition of the orienteering problem into a knapsack problem (assignment of locations with rewards to agents with limited switching cost budgets) and a subsequent traveling salesperson problem (selection of most efficient route for assigned locations). Fast knapsack algorithms are based on selecting locations in terms of their marginal reward per additional resource cost. For orienteering problems, the resource cost is hard to evaluate because it requires the solution of the subsequent traveling salesperson problem. We develop a novel approach based on spanning trees which allow for estimation of increased resource costs, leading to a fast algorithm for selecting locations to be searched by each agent. We subsequently determine paths for the selected locations using fast approximations for the traveling salesperson problem, such as the Lin-Kernighan-Helsgaun algorithm [Helsgaun, 2000]. The resulting tours for each agent can be refined further by exploring additional local search techniques.

The chapter is organized as follows: Section 5.1 describes our algorithm for the single orienteering problem. Section 5.2 describes our algorithm for the team orienteering problem. Section 5.3 contains experimental results comparing our algorithms

with available algorithms in the literature.

5.1 The Single Orienteering Problem (SOP)

We first discuss the problem for a single agent. We model the location of potential objects of interest in terms of an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_N\}$ is the *node* set and $E = \{(v_i, v_j) : 1 \leq i < j \leq N\}$ is the *edge* set. Each node represents a location with a potential object of interest, and edges represent potential paths between locations. Assume that $(v_i, v_j) \in E$ if and only if $(v_j, v_i) \in E$. Let v_1 denote the *home* node where the tour starts and ends.

Each node v_k has a positive *reward* r_k that can be collected by having the agent visit the node. We assume that the reward at a node can be collected only once. Each edge (v_i, v_j) has an *edge cost* c_{ij}, which represents the amount of resource required to traverse the edge. Assume that $c_{ij} = c_{ji}$. We assume that the edge costs satisfy the triangle inequality: $c_{ik} + c_{kj} \geq c_{ij}$ for all distinct i, j, k. For many of the applications we are interested in, the cost of an edge is the Euclidean distance between its two nodes.

Let B denote the *budget*, in units of cost as above. The goal of the orienteering problem [Golden et al., 1987, Vansteenwegen et al., 2011] is to design a closed path, or a *tour*, to visit a subset of nodes that maximizes the total reward collected, without exceeding the budget constraint. This problem has ingredients of both the *traveling salesperson problem* (TSP) and the *knapsack problem* and is NP-Hard. A node subset that will be visited must be selected; one would like to select a subset with as high reward as possible, while respecting budget constraints. Evaluating the required budget for a given node subset requires solution of a traveling salesperson problem. This coupling makes it difficult to use fast approximate algorithms for either knapsack problems or TSP problems.

The above problem can be formulated as an integer programming problem [Tsiligirides, 1984]. Define binary variables $\beta_k \in \{0,1\}$ $(k = 2,\ldots,N)$ to be 1 if node v_k is visited and 0 otherwise. Define binary variables $e_{ij} \in \{0,1\}$ $(1 \le i < j \le N)$ to be 1 if edge (v_i, v_j) is travelled and 0 otherwise. To handle the case where the solution contains only one edge – from v_1 to some v_j, e_{1j} is also allowed to take value 2 in addition to 0 and 1. Then we have the following integer linear programming formulation:

$$
\begin{aligned}
\underset{\{\beta_k\}}{\text{maximize}} \quad & \sum_{k=2}^{N} r_k \beta_k \\
\text{subject to} \quad & \sum_{j=2}^{N} e_{1j} = 2; \quad \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} e_{ij} \le B \\
& \sum_{i=1}^{k-1} e_{ik} + \sum_{j=k+1}^{N} e_{kj} = 2\beta_k, \quad k = 2,\ldots,N \\
& 2 \sum_{v_k \in S} \beta_k \le |S| \Big(\sum_{v_i \in S, v_j \notin S} e_{ij} + \sum_{v_i \notin S, v_j \in S} e_{ij} \Big) \\
& \qquad\qquad S \subset V \setminus \{v_1\},\ |S| \ge 2 \qquad (5.1) \\
& e_{1j} \in \{0,1,2\}, \quad j = 2,\ldots,N \\
& e_{ij} \in \{0,1\}, \qquad 2 \le i < j \le N \\
& \beta_k \in \{0,1\}, \qquad k = 2,\ldots,N
\end{aligned}
$$

The above formulation modifies the standard TSP by introducing the node selection variables β_k and the budget constraint. In addition, the objective function is to maximize the total reward of the nodes being visited. The above formulation has an exponential number of subtour elimination constraints (5.1), limiting the applicability of direct integer programming approaches. Next, we describe our proposed alternative approximate algorithms for obtaining fast solutions to the above problem.

The optimization procedure of the orienteering problem involves selecting a node

subset. In order to efficiently utilize the budget, the ideal way is to travel along an optimal TSP tour in the induced subgraph of the selected node subset.

Given a subset of nodes $S \subset V$ that includes the origin v_1, we want to estimate the resource cost to visit those nodes. A modern technique for finding an approximate TSP tour is the *Lin-Kernighan-Helsgaun* (LKH) algorithm [Helsgaun, 2000], which we use to estimate the resource cost. Let $c_{LKH}(S)$ denote the cost of the Hamiltonian tour on S found by the LKH algorithm.

Our algorithm exploits the following facts from graph optimization problems: Given any spanning tree T on S, the cost of an optimal tour on S is no greater than twice the cost of the edges of the tree, $c_{tree}(T)$. Thus, $2 \cdot c_{tree}(T)$ provides an estimate of the resource cost without requiring a tour solution. Furthermore, the tree provides a topological order on the nodes in the tree induced by a depth-first traversal. The cost of the Hamiltonian tour traversing those nodes in topological order is a better estimate of the resource cost, $c_{tree-tour}(T)$. Better estimates of tour length are available, but these require solution of optimization problems that are time-consuming.

Our algorithm consists of three major steps and a refinement step, which will be explained next.

Algorithm for the Single Orienteering Problem

Step 1 (Tree Growing)

Starting from node v_1, we grow a tree using a greedy knapsack approach. Suppose the current tree T spans node subset S where $v_1 \in S$ and has tour cost $c(T)$ (as estimated by the weight $2 \cdot c_{tree}(S)$). For each unselected node $v_k \notin S$, compute the

minimum cost to connect this node as a new leaf to tree T, denoted c_k, as follows:

$$c_k = \min_{v_j \in T} c_{jk} \tag{5.2}$$

Denote by j^* the index of the closest node to v_k in T. The question is whether connecting v_k as a new leaf in T results in a shorter tree than inserting v_k as the parent node to j^*, denoted as $par(j^*)$. This is easily answered by considering whether $c_{par(j^*)j^*} > c_{par(j^*)k}$. If so, it is best to insert node v_k as the parent of node v_{j*}. The incremental increase in tree length is

$$s_k = \min\{c_k, c_k + c_{par(j^*)k} - c_{par(j^*)j^*}\} \tag{5.3}$$

Define the *Reward-to-Connection-Cost Ratio*

$$RCCR(k,T) = \frac{r_k}{s_k} \tag{5.4}$$

Select the node v_k with the biggest RCCR among the nodes such that $c(T \cup \{v_k\}) \leq B$ and add it to tree T as computed previously. This continues until none of remaining nodes outside of S satisfies $c(T \cup \{v_k\}) \leq B$.

Step 2 (Tree Improvement):

Denote the cost of a tour using the topological order imposed from a depth-first traversal of the tree as $c_{tree-tour}(T)$. $c_{tree-tour}(T)$ is generally smaller than $2 \cdot c_{tree}(T)$, and thus is a better estimate of the resource cost.

Select node v_k with the biggest RCCR computed as in (5.4) among the nodes such that $c_{tree-tour}(T \cup \{v_k\}) \leq B$ and add it to the tree. This continues until none of remaining nodes outside of S satisfies $c_{tree-tour}(T \cup \{v_k\}) \leq B$.

Step 3 (Tour Construction)

The resulting tree from the first two stage identifies a set of nodes S with locations that should be searched. To find the shortest route to visit and search these locations,

we run the LKH algorithm on the selected node subset S to find a tour, denoted by I_S, with cost $c_{LKH}(S)$.

Step 4 (Tour Improvement)

The tour I_S constructed by the LKH algorithm in the last step may not exhaust the budget, so its cost $c(I_S) < B$. If so, we may be able to insert some of the remaining nodes into the tour. For each node $v_k \notin S$, define its *incremental cost* given l_S as $d_k(I_S) = \min_{(v_i,v_j)\in I_S} c_{ik} + c_{jk} - c_{ij}$. If $c(I_S) + d_k(I_S) \leq B$, then node v_k is feasible for adding to the tour I_S. Find the feasible node v_k with highest *Reward-to-Incremental-Cost Ratio* (RICR), $\frac{r_k}{d_k(I_S)}$ and insert it in the tour I_s at the location with minimal incremental cost. Repeat incrementally until no more nodes can be added in this way with the given budget.

Step 5 (Tour Refinement — Optional)

After more nodes are inserted into the tour, the tour we currently have may not be an optimal TSP tour on this enlarged node subset. In this case, we find a new tour using the LKH algorithm on the current node subset and repeat the tour improvement step until no further improvements can be made. This step is not needed in most cases, but it is a simple step.

5.2 The Team Orienteering Problem (TOP)

The above algorithm can be extended to problem with multiple agents that work as a team. For M ($M \geq 2$) agents, M tours are planned that search disjoint sets of locations. For simplicity, we assume that the same reward will be collected if the same location is searched by any agent, and all agents have the same budget B. We also assume that all agents start from a common base location, at node v_1. The goal

of the team orienteering problem [Chao et al., 1996b, Vansteenwegen et al., 2011] is to plan M tours, all starting and ending at v_1, to collect maximal reward without exceeding the budget limit of any agent.

An integer programming formulation of the team orienteering problem is as follows. Define binary variables $\beta_{km} \in \{0,1\}$ ($k = 2,\ldots,N$, $m = 1,\ldots,M$) to be 1 if node v_k is visited by tour m and 0 otherwise. Define binary variables $e_{ijm} \in \{0,1\}$ ($1 \le i < j \le N$, $m = 1,\ldots,M$) to be 1 if edge (v_i, v_j) is in tour m and 0 otherwise. Since tour m may visit only one node other than v_1, e_{1jm} is also allowed to take value 2 in addition to 0 and 1. Then we have the following integer linear programming formulation:

$$
\begin{aligned}
\underset{\{\beta_{km}\}}{\text{maximize}} \quad & \sum_{m=1}^{M}\sum_{k=2}^{N} r_k \beta_{km} \\
\text{subject to} \quad & \sum_{m=1}^{M} \beta_{km} \le 1,\ k = 2,\ldots,N \qquad (5.5) \\
& \sum_{j=2}^{N} e_{1jm} = 2;\ \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} e_{ijm} \le B,\ \forall m \\
& \sum_{i=1}^{k-1} e_{ikm} + \sum_{j=k+1}^{N} e_{kjm} = 2\beta_{km},\ k = 2,\ldots,N,\ \forall m \\
& 2\sum_{v_k \in S} \beta_{km} \le |S| \Big(\sum_{v_i \in S, v_j \notin S} e_{ijm} + \sum_{v_i \notin S, v_j \in S} e_{ijm} \Big) \\
& \qquad\qquad S \subset V \setminus \{v_1\},\ |S| \ge 2,\ \forall m \\
& e_{1jm} \in \{0,1,2\},\ \ j = 2,\ldots,N,\ \forall m \\
& e_{ijm} \in \{0,1\}, \qquad 2 \le i < j \le N,\ \forall m \\
& \beta_{km} \in \{0,1\}, \qquad k = 2,\ldots,N,\ \forall m
\end{aligned}
$$

Compared to the previous integer programming formulation for the single tour case, a new set of constraints (5.5) is imposed so that each node is only visited

once across tours. The objective function becomes the total reward collected by all tours, and the rest of the constraints in the orienteering problem formulation are now extended to M tours. As before, instead of trying to solve the integer linear programming problem, we propose an approximate algorithm based on knapsack heuristics to effectively find solutions. This algorithm generalizes the approach described in the previous section to multiple agents.

For simplicity of exposition, we limit our description to the case where $M = 3$ agents. The approach generalizes readily to other number of agents.

Algorithm for the Team Orienteering Problem

Step 1 (Initialization)

One of the hardest parts of our algorithm is initialization, as we must initialize simultaneously $M = 3$ tours. To do so, we restrict our attention to all locations that are within round-trip distance of the source node v_1. Partition the domain circularly into 12 equal-angle sectors centered at v_1. For each sector, compute the total reward of locations present in that sector. Determine the 3 sectors with the highest rewards, and select the node nearest to v_1 in each of the 3 sectors to connect to v_1 and be part of the tour for each agent. The goal is to start the trees with some geographic diversity, with anticipation of directions where the higher value locations exist.

Step 2 (Tree Growing)

Denote each of the $M = 3$ trees rooted at v_1 as T_1, T_2, T_3, and let S_i be the node subset spanned by T_i. Denote the cost of the edges in tree T_m by $c_{tree}(T_m)$. For each node $v_k \notin S_1 \cup S_2 \cup S_3$, compute the incremental distance to insert in tree T_m, denoted as s_k^m using (5.2), (5.3). We now compute the Reward-to-Connection-Cost Ratio (RCCR) of v_k with respect to each tree T_m as before, as in (5.4):

$$RCCR(k, T_m) = \frac{r_k}{s_k^m}$$

Determine the biggest $RCCR(k^*, T_{m^*})$ satisfying $c_{tree}(T_{m^*} \cup \{v_{k*}\}) \leq \frac{B}{2}$, and insert v_{k*} into tree T_{m^*} with minimal insertion cost. Update the values $RCCR(k, T_m)$ and continue inserting the largest RCCRs until no more insertions into any of the trees are feasible.

Step 3 (Tree Improvement)

We refine the estimated cost of a tour for each tree by using the topological order imposed from a depth-first traversal of the tree. Let $c_{tree-tour}(T_m \cup \{v_k\})$ denote the resource cost of the topological tour of tree T_m after adding v_k to it. Add node $v_k \notin S$ to its best T_m in order of biggest RCCR as long as $c_{tree-tour}(T_m \cup \{k\}) \leq B$.

Step 4 (Tour Construction)

Given the trees T_m, we run the LKH algorithm on the nodes S_m in each tree T_m to obtain tours I_{S_m}. If the cost of a tour exceeds B, delete the last node added to the corresponding tree and rerun the LKH algorithm until feasible tours I_{S_m} are found.

Step 5 (Tour Improvement)

Compute the Reward-to-Incremental Cost Ratio (RICR) of each unselected node with respect to the three tours, and insert node in order of biggest RICR as long as the cost of the tour meets the budget constraint.

Step 6 (Tour Refinement — Optional)

Find new tours for each agent using the LKH algorithm on the current node subsets and repeat the tour improvement step until no further improvements can be made.

5.3 Experiments

Now we present some numerical results to demonstrate the effectiveness of our new orienteering algorithms. All experiments are conducted in MATLAB 2014 on a laptop computer with Intel i7-4600M processor and 8GB RAM.

5.3.1 Experiments for SOP

In this section, we test our SOP algorithm on the data used in [Ding et al., 2016]. The data in [Ding et al., 2016] was generated using the Statlog LANDSAT dataset from the UCI Machine Learning Repository [Frank and Asuncion, 2010]. This LANDSAT dataset has 4435 training examples. Each example is an image of very low-resolution (3×3) representing one of six object types, such as cotton crop, red soil, etc. It has a tag indicating which object type it belongs to.

We are interested in confirming the presence of cotton crops in an area. We have 10 test areas to look for cotton crops. Each area has 100 potential locations, for which low resolution LANDSAT imagery is available, and contains between 3 and 5 locations with cotton crops. Using a binary classifier as in [Wang et al., 2014], we estimate the probability that each location in each area contains a cotton crop using the LANDSAT data, and assign a reward for visiting and searching that location, consisting of the entropy of the Bernoulli distribution with that probability. For each of the 10 areas, the goal for the agent is to visit locations in the area collect as much reward as possible given limits on total distance traveled in its tour.

We compare our algorithm (hereinafter referred to as "New") with a greedy baseline approach (referred to as "Bsl"): We grow the tour incrementally by growing a path from the home node, and adding nodes not already in the path using an algorithm similar to the tour improvement step: We compute the reward per incremental cost of connecting to the most recently added node for nodes not already in the path,

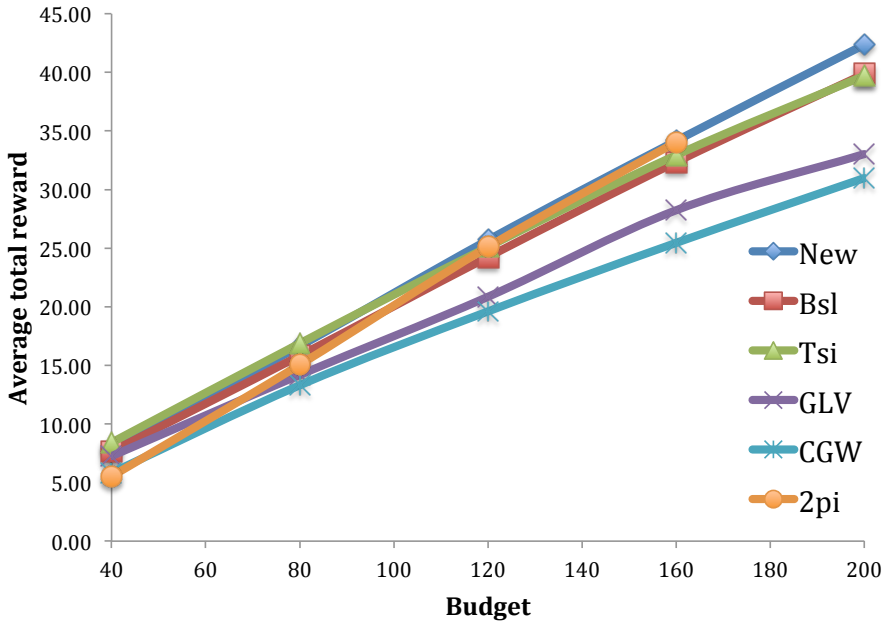


Figure 5·1: Average reward collected over 10 areas for different algorithms and budgets.

and choose to extend the path by selecting the node with highest marginal reward among those nodes that can be added while leaving enough budget to return to the home node. This construction maintains a path. If the given budget does not allow us to visit one more node before returning to the original node, then finish and use this as the final tour. In addition, we compare our algorithm with the algorithms proposed by Tsiligirides in [Tsiligirides, 1984] (referred to as "Tsi"), Golden-Levy-Vohra in [Golden et al., 1987] (referred to as "GLV"), Chao-Golden-Wasil in [Chao et al., 1996b] (referred to as "CGW"), and Silberholz-Golden in [Silberholz and Golden, 2010] (named "2-Parameter Iterative" algorithm; referred to as "2pi" here).

To evaluate tours, we use 3 criteria: average reward achieved across the 10 areas, average computation time, and average remaining travel time budget for the agent. We use the ten areas with 100 locations discussed previously, where the average total reward present in those areas is 61.37. We vary the budget for the agent ranging

from 40 to 200 distance units. Fig. 5·1 shows the average rewards collected by the different algorithms. Table 5.1 describes the computation time as well as the average unused budget by each algorithm. Note that 40 units of budget is a tight budget where the agent can collect only a small fraction of the total reward available, and 200 is a budget that allows the agent to collect over half of the total reward available.

The results in Fig. 5·1 and Table 5.1 show that, in terms of reward collected, our algorithm is better than the baseline algorithm, the Golden-Levy-Vohra algorithm, the Chao-Golden-Wasil algorithm and the 2-Parameter Iterative algorithm universally, and also outperforms the Tsiligirides algorithm once enough budget is available. Note that our algorithm is between 23 and 70 times faster than the Tsiligirides algorithm, and has more unused budget, which implies more robustness against unexpected resource consumptions. Furthermore, the results indicate that the computation time of our algorithm grows linearly with the budget, and remains efficient even in a MATLAB implementation. In contrast, the computation algorithm of Golden-Levy-Vohra algorithm explodes nonlinearly as the number of locations increases. The 2-Parameter Iterative algorithm has excessive computation requirements, failing to complete the search for a tour when the budget increased to 200. The Chao-Golden-Wasil algorithm had reasonable computation times, but it achieved under 75% of the total reward that our algorithm collected with comparable budgets. Surprisingly, the second best performing algorithm was our baseline greedy algorithm, which used only the tour improvement step. The algorithm was fastest among all the algorithms considered, and achieved over 92% of the value that our algorithm collected for different budgets.

5.3.2 Experiments for TOP

In this section, we present experiments of our algorithm for TOP. We first test our algorithm on the same benchmark instances that were used in prior algorithm evalu-

ations [Chao et al., 1996b], which are now available online [Chao et al., 1996c]. We assume that there are 3 agents, so $M = 3$, and our algorithm will develop solutions with three tours. There are four datasets (p4.3, p5.3, p6.3 and p7.3) available for three-tour team orienteering problem. Each dataset contains a fixed physical layout of the nodes, but changes the available budget for the agents over nearly 20 levels. Dataset p4.3 has 100 nodes, with budget levels per agent from 25 to 80 units. Dataset p5.3 has 66 nodes, with budgets ranging from 1.7 to 43.3 units per agent. Dataset p6.3 has 64 nodes, with budget levels from 5 to 26.7 units per agent. Dataset p7.3 has 102 nodes, with budgets from 6.7 to 133 units per agent.

We compare our results with the algorithm of Chao-Golden-Wasil described in [Chao et al., 1996b], which was originally proposed for the path team orienteering problem. For each dataset, Figure 5·2 plots the reward that our algorithm obtains as a function of the agent budget, and compares it with the reward obtained by the Chao-Golden-Wasil TOP algorithm (referred to as "CGW-T"). The most important feature of these results is the steady monotone increase in performance of our algorithm as budgets increase. In contrast, the performance of the CGW-T algorithm is somewhat erratic, with significant deviations from monotonicity. These deviations are most pronounced in the larger scenarios (p4.3 and p7.3). An example of the three tours found by our algorithm is shown in Fig. 5·3, for dataset 4.3 with budget 76.7.

To provide a more quantitative comparison, we computed the average team reward (Avg. Team Rwd.) and the average computation time (Time) across the different budget levels in each dataset, and summarize the results in Table 5.2. As the table illustrates, both algorithms have comparable average computation times. However, our algorithm achieves higher average rewards in 3 out of the 4 data sets. The performance difference in dataset 5 is due to a tour improvement heuristic in the CGW-T algorithm that swaps nodes among tours. We chose to leave that step out of

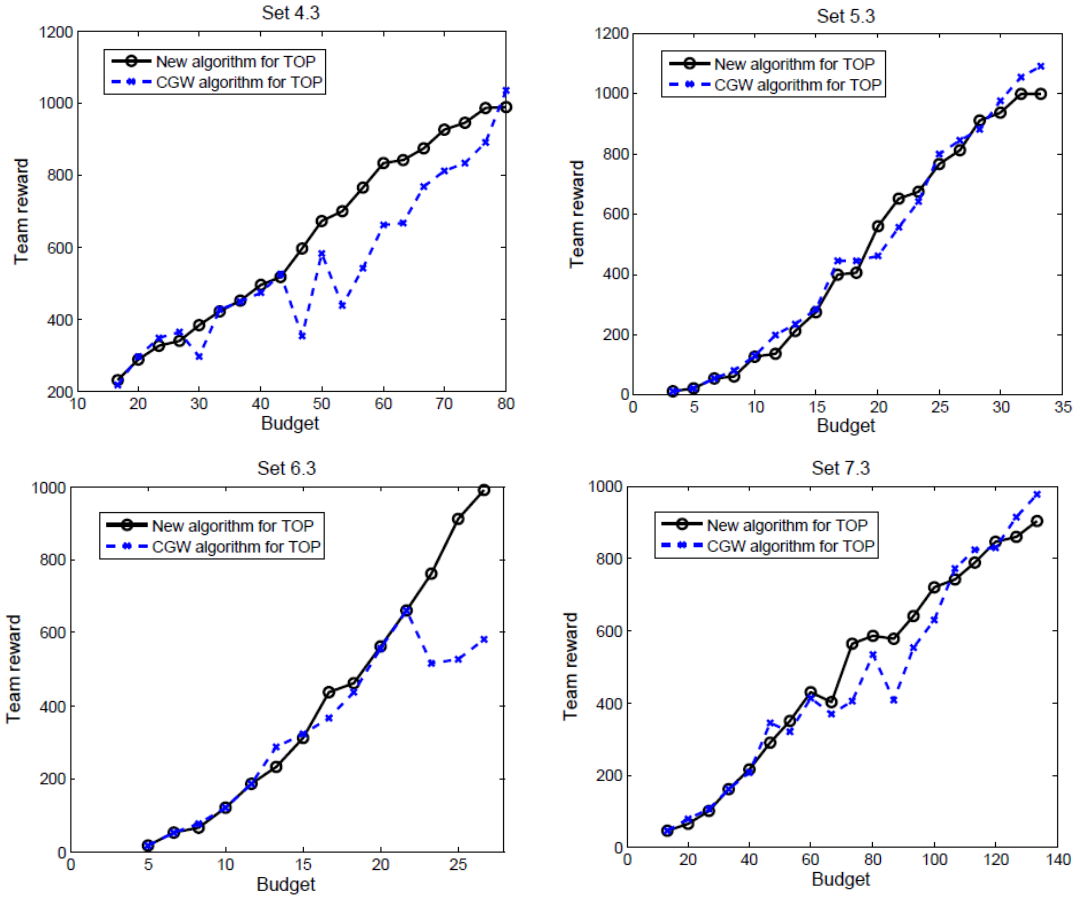


Figure 5·2: Comparisons of our new algorithm with the modified Chao-Golden-Wasil (CGW) algorithm for the orienteering on different datasets.

our algorithm, in the expectation that starting from a better set of node assignments to the individual agents would provide sufficient performance using our simpler tour improvement approach.

As a second set of experiments, we compared both algorithms on the ten LANDSAT scenarios discussed in Section 5.3.1, where we varied the budget per agent from 20 units to 120 units. The results are shown in Table 5.3. As the results illustrate, our algorithm collects more reward on average, and has more resources in reserve, than the CGW-T algorithm. Furthermore, the computation time of our algorithm grows linearly with the amount of resources, whereas the computation time of the CGW-T

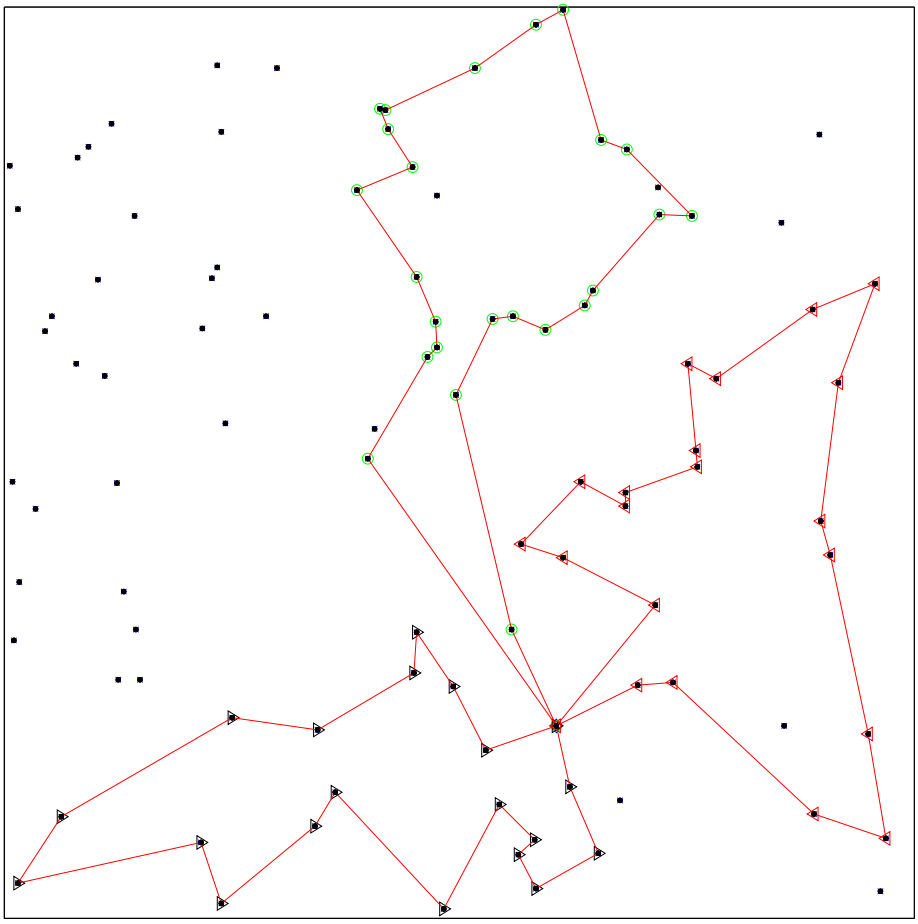


Figure 5·3: An example of the tour found by the new algorithm, for dataset 4.3 with budget 76.7.

algorithm grows nonlinearly, becoming significantly slower than our algorithm for the resource level of 120 units per agent.

Budget		40	80	120	160	200
Time (seconds)	New	5.24	7.12	10.27	12.03	16.41
	Bsl	0.05	0.10	0.14	0.18	0.24
	Tsi	201.9	426.1	625.4	800.2	990.7
	GLV	0.11	0.29	10.50	16.02	135.0
	CGW	11.26	21.39	24.64	28.11	33.26
	2pi	2239	3941	12606	48951	N/A
Bgt. Lft.	New	1.415	1.532	1.148	1.372	1.045
	Bsl	1.233	1.160	1.760	1.859	1.200
	Tsi	0.407	0.478	0.603	0.787	1.063
	GLV	1.602	0.783	1.258	0.976	1.550
	CGW	1.022	0.711	0.526	1.443	0.981
	2pi	1.130	0.570	0.250	0.560	N/A

Table 5.1: Average computation time (Time) and average unused budget (Bgt. Lft.) over ten areas for different algorithms.

Dataset		4.3	5.3	6.3	7.3
Avg. Team Rwd.	New-T	629.6	640.2	412.7	488.1
	CGW-T	594.3	677.0	336.9	467.3
Time (seconds)	New-T	3.13	2.20	0.95	1.83
	CGW-T	2.33	0.79	0.32	1.48

Table 5.2: Average team reward (Avg. Team Rwd.) and average computation time (Time) for our new algorithm and the modified CGW algorithm.

Individual budget		20	40	80	120
Avg. Team Rwd.	New-T	9.47	20.06	31.73	42.35
	CGW-T	8.60	13.63	21.19	35.25
Time (seconds)	New-T	9.40	11.41	13.37	15.92
	CGW-T	1.54	5.78	15.40	21.80
Idv. Bgt. Lft.	New-T	1.999	2.732	4.734	7.235
	CGW-T	0.930	1.213	0.818	0.710

Table 5.3: Average team reward (Avg. Team Rwd.), average computation time (Time) and average unused individual budget (Idv. Bgt. Lft.) over the ten scenarios of the LANDSAT data, for both the new algorithm and the Chao-Golden-Wasil algorithm.

Chapter 6

Multi-agent Adaptive Search with Complex Error Models

In this chapter, we generalize the single-agent adaptive search problem of [Jedynak et al., 2011] to multiple agents with correlated error models. We pose this as an equivalent multi-region single-agent search problem where the agent can partition the object space into multiple regions and inquire as to which region the object is located in. We adopt a Bayes formulation similar to that in [Jedynak et al., 2011] with the goal of reducing the final entropy of the conditional probability density of the object location after a fixed number of observations. We solve the resulting stochastic control problem. We provide a constructive algorithm for computing the optimal strategies based on convex optimization. We derive a lower bound on the performance of the minimum mean-square error estimator, and provide upper bounds on the estimation error for special cases. Our results extend the results of [Tsiligkaridis et al., 2014] to correlated error models with general measurement spaces.

For the case of multi-agent search with independent error models, we extend the results of [Tsiligkaridis et al., 2014], to general asymmetric error models. We show that the optimal sensing strategies can be obtained in terms of the solution of decoupled scalar convex optimization problems, followed by a joint region selection procedure. We describe a generalized symmetry condition for non-binary error models that enables the analytic solution of the decoupled scalar optimization problems, and provide a constructive solution for generating the optimal adaptive sensing strategies.

We also consider the case where each agent is also allowed to choose among sensing modes with different error models for their observations at different costs, extending the results of [Sznitman et al., 2013] to the multi-agent case. We derive the optimal policies for this problem and show that the optimal strategies can again be computed in terms of the solution of single-agent problems followed by a joint region selection procedure.

We further consider adaptive search of multiple objects of interest using multiple agents. We derive an optimal sensing policy. For search of two objects in an interval using two agents, we provide an explicit construction for determining adaptively the sensing actions for each agent. We illustrate the performance of our proposed policy in simulation, comparing to alternative policies based on spatial domain partition for each agent.

The chapter is structured as follows: Section 6.1 contains the formulation and results for the multi-agent search with correlated error models, and the equivalent multi-region single-agent search problem. Section 6.2 contains the results for the multi-agent search problem with independent error models. Section 6.3 contains the results for multi-agent search problem where agents can control both the choice of sensing area as well as the precision mode of that search. In Section 6.4, we study the case where multiple objects of interest are present. Section 6.5 contains simulation results that illustrate the performance of our approaches.

6.1 Correlated Error Models

6.1.1 Problem Formulation

Consider the problem of localizing a stationary point object whose position is denoted by X, a continuous-valued random vector in a compact subset $\mathcal{X}$ of $\mathbb{R}^d$ ($d \leq 3$ for our purposes) with prior probability distribution that is absolutely continuous with

respect to Lebesgue measure, with density $p_0(x)$. We assume this initial density has finite differential entropy. We have M ($M \geq 2$) agents that can collect measurements of the object location at discrete times $n = 1, \ldots, N$. The agent measurement model is similar to that in [Jedynak et al., 2011], motivated by problems in group testing [Atia and Saligrama, 2012]: at each time n, each agent m selects a region A_n^m, which is a Lebesgue-measurable subset of $\mathcal{X}$, and collects a measurement Y_n^m that depends on whether the object X is contained in the sensed region A_n^m. The measurements Y_n^m take values in a discrete set $\mathcal{Y}$. Extensions to continuous-valued measurements are straightforward provided one places restrictions on the admissible sets that can be queried by agents to ensure Borel measurability of the resulting random variables. Such restrictions are not needed when the measurements are discrete-valued.

A team decision at time n corresponds to a collection of observation regions $(A_n^1, \ldots, A_n^M)$ for all M agents, where $A_n^1, \ldots, A_n^M \subset \mathcal{X}$. Define the random variables

$$Z_n^m = \mathbb{1}_{\{X \in A_n^m\}}, \quad \text{for agent } m \text{ at stage } n$$

Let $\boldsymbol{A}_n = (A_n^1, \ldots, A_n^M)$ denote the collection of sensing areas observed at stage n, and $\boldsymbol{Y}_n = (Y_n^1, \ldots, Y_n^M)$ denote the collection of noisy observations collected at stage n, where $\boldsymbol{Y}_n \in \mathcal{Y}^M$. The measurement process is described by conditional probability distributions that represent correlated measurement errors across agents, of the form

$$P(\boldsymbol{Y}_n = \boldsymbol{y} | X, \boldsymbol{A}_n) = P(\boldsymbol{Y}_n = \boldsymbol{y} | Z_n^1, \ldots, Z_n^M) \tag{6.1}$$

where (6.1) highlights the assumption that the measurement distribution depends on whether the object is present in the observed region, and not on the specific regions being observed. In addition, the measurements $\boldsymbol{Y}_n, \boldsymbol{Y}_k$ are assumed to be conditionally independent for $k \neq n$ given the respective indicators $Z_n^1, \ldots, Z_n^M$ and

$Z_k^1, \ldots, Z_k^M$. The use of correlated errors across the multiple agents represents errors created by fluctuations in the signatures associated with X, which can result in low signal-to-noise measurements simultaneously in all agents observing a region that contains X.

The assumption that the distribution of the measurements $\boldsymbol{Y}_n$ depends only on whether the object is included in the observation regions $(A_n^1, \ldots, A_n^M)$ is a generalization of the measurement model used in [Jedynak et al., 2011, Tsiligkaridis et al., 2014]. It corresponds to agents that combine signals from multiple locations for single processing. Examples of such agents are single pixel detectors that use wide-area patterned illumination on a sparse field with only one reflector to detect the presence of the reflector in the illuminated areas; localization is accomplished by varying the illumination pattern rather than by increasing the resolution of the illumination beam [Sun et al., 2014]. This model also arises in transmission of information over correlated memoryless binary channels [Cover and Thomas, 2012], and in group testing for chemical sensing [Atia and Saligrama, 2012], where samples from different regions are mixed to detect the presence of chemical agents. These agents exploit the sparsity of the signal to obtain rapid localization with greatly reduced number of measurements when compared with agents that measure small areas, similar to the manner in which compressive sensing [Donoho, 2006] reconstructs the support of a signal with a small number of measurements.

Our goal is to obtain measurements sequentially over N times to improve our knowledge of the object location X. We wish to do this adaptively, exploiting the past information collected by the agents. Denote the information history collected by the agents after the measurements at time n have been obtained by:

$$D_n = \{\boldsymbol{A}_1, \boldsymbol{Y}_1, \cdots, \boldsymbol{A}_n, \boldsymbol{Y}_n\}$$

Denote by $p_n(x)$ to be the conditional probability density of the object location X given the information history D_n, so

$$p_n(x) \equiv p(x|D_n)$$

We refer to this quantity as the information state at time n. The evolution of this information state across stage is derived using Bayesian reasoning, as follows. Assume that, at time $n+1$, we know $p_n(x)$, and obtain a measurement $\boldsymbol{Y}_n$ from observing sensing areas $\boldsymbol{A}_n$. Then,

$$p_{n+1}(x) = p_n(x) \frac{P(\boldsymbol{Y}_{n+1} = \boldsymbol{y}|\boldsymbol{A}_{n+1}, X = x)}{\int_{\mathcal{X}} p_n(\sigma) P(\boldsymbol{Y}_{n+1} = \boldsymbol{y}|\boldsymbol{A}_{n+1}, X = \sigma) d\sigma} \tag{6.2}$$

The above evolution can be viewed as a stochastic dynamical system for the measure-valued information state $p_{n+1}(x)$, where the evolution depends on the finite-valued random "disturbance" $\boldsymbol{y}$, with conditional probability distribution

$$\eta(\boldsymbol{y}) = \int_{\mathcal{X}} p_n(\sigma) P(\boldsymbol{Y}_{n+1} = \boldsymbol{y}|\boldsymbol{A}_{n+1}, X = \sigma) d\sigma$$

that depends on the current information state $p_n(x)$ and the control action $\boldsymbol{A}_{n+1}$. As long as $\eta(\boldsymbol{y}) > 0$, the resulting information state p_{n+1} is well-defined, and will represent a probability density on $\mathcal{X}$. For $\eta(\boldsymbol{y}) = 0$, we arbitrarily define $p_{n+1}(x) = p_n(x)$.

To complete the formulation, we define admissible strategies and objectives for the stochastic control problem. Let $\mathcal{S}$ denote the space of probability densities $p(x)$ over $\mathcal{X}$. Let $\Gamma(\mathcal{X})$ denote the set of all Lebesgue-measurable subsets of $\mathcal{X}$. We define an adaptive joint sensing policy $\pi = (\pi_1, \pi_2, \cdots, \pi_N)$ to be a sequence of functions where $\pi_n : \mathcal{S} \to \Gamma(\mathcal{X})^M$ will map the posterior density $p_{n-1}(x)$ into admissible batch sensing regions $\boldsymbol{A}_n = (A_n^1, \ldots, A_n^M)$. Let Π denote the space of all adaptive joint

sensing policies.

In terms of objective, we will evaluate the quality of our knowledge of X after collecting information D_n by its *posterior differential entropy* $H(p_n)$ defined as

$$H(p_n) = -\int_{\mathcal{X}} p_n(x) \log_2 p_n(x) dx$$

Our objective is to minimize $H(p_N)$ — the posterior differential entropy after N stages of joint sensing:

$$\inf_{\pi \in \Pi} E[H(p_N)|p_0] \tag{6.3}$$

Note that this objective is a nonlinear functional of the information state, unlike the standard models for partially observed Markov decision processes [Bertsekas, 2005] where the final objective is a linear functional of the information state.

The above dynamic decision problem can be viewed as a perfectly observed Markov decision problem with infinite-dimensional state space $\mathcal{S}$, stochastic dynamics with discrete-valued disturbances (6.2), and terminal cost objective (6.3). Given the discrete nature of the observations, concerns about measurability of strategies are simplified; our problem is a stochastic optimal control problem with countable disturbances described in Chapter 3 of [Bertsekas and Shreve, 1978], which allows us to apply dynamic programming techniques to characterize optimal strategies. An alternative approach used in [Jedynak et al., 2011, Tsiligkaridis et al., 2014] is to restrict the regions A^m to unions of rectangular areas, enabling the use of stochastic control results for Borel models.

We define the optimal value function $V(p_n, n)$ at stage n to be:

$$V(p_n, n) = \inf_{\pi_{n+1}, \ldots, \pi_N} E[H(p_N)|p_n]$$

The optimal value function has to satisfy the Bellman equation [Bertsekas and Shreve, 1978]:

$$V(p_n, n) = \inf_{\boldsymbol{A}_{n+1}} E_{\boldsymbol{Y}_{n+1}}[V(p_{n+1}, n+1)|\boldsymbol{A}_{n+1}, p_n] \tag{6.4}$$

Furthermore, if a policy π^* satisfies

$$E_{\boldsymbol{Y}_{n+1}}[V(p_{n+1}, n+1)|\pi^*_{n+1}(p_n), p_n] = V(p_n, n)$$

for all p_n, then the policy is optimal.

To obtain a solution, we will exploit the following equivalence: A set of observation regions by the collection of M agents, denoted by $\boldsymbol{A} = (A^1, \ldots, A^M)$ induces a partition of the region $\mathcal{X}$ into 2^M subsets. For each $k \in \{0, \ldots, 2^M - 1\}$, denote the dyadic expansion of k as $i_M i_{M-1} \cdots i_1$. Then, the subset B^k in the partition can be identified as

$$B^k = \cap_{m=1}^{M} (A^m)^{i_m}$$

where we use the set notation $(A)^0 \equiv A^c$ and $(A)^1 \equiv A$.

Thus, the sets B^k are defined as intersections of observation regions or their complements, one for each agent. Selecting a set of observation regions $\boldsymbol{A}$ corresponds to a unique partition of $\mathcal{X}$ into 2^M Lebesgue measurable subsets. Similarly, given any partition $\{B^k, k = 0, \ldots, 2^{M-1}\}$ of $\mathcal{X}$ into Lebesgue measurable subsets, let $i_M i_{M-1} \cdots i_1$ denote the dyadic expansion of k. Then, define the sets

$$A^m = \cup_{\{k: i_m = 1\}} B^k$$

Then, the partition $\{B^k\}$ uniquely defines a set of M measurable regions $\boldsymbol{A} = (A^1, \ldots, A^M)$ to be observed by the M agents.

To solve the multi-agent problem for the optimal adaptive sensing regions $\boldsymbol{A} =$

$(A^1, \ldots, A^M)$, we will solve the single-agent, multi-region problem for the optimal adaptive partition $\{B^0, \ldots, B^K\}$, where $K = 2^M - 1$. Note the following correspondence: The object location $X \in B^k$ if and only if the dyadic expansion of k is given by $Z_M Z_{M-1} \cdots Z_1$, where $Z_i = \mathbb{1}_{\{X \in A^i\}}$.

To simplify notation, define

$$f_k(\boldsymbol{y}) = P(\boldsymbol{Y} = \boldsymbol{y} | X \in B^k), \quad k = 0, \ldots, K$$

which, by our previous assumptions, depends only on whether X is in the region B^k. A useful quantity in our development is the *joint operating point* at time $n+1$, $\boldsymbol{u}_{n+1} = \{u_{n+1}^k, k = 0, \ldots, K = 2^M - 1\}$ for a collection of observed regions $\boldsymbol{A}$ resulting in a partition $\boldsymbol{B} = \{B^0, \ldots, B^K\}$, given information state $p_n(x)$. This joint operating point is defined as

$$u_{n+1}^k = \int_{B^k} p_n(\sigma) d\sigma \ \geq 0 \tag{6.5}$$

Since $\boldsymbol{B}$ is a partition of $\mathcal{X}$, we have $\sum_{k=0}^K u_k = 1$. With this notation, the denominator in (6.2) simplifies to

$$\eta(\boldsymbol{y}) = \sum_{k=0}^K f_k(\boldsymbol{y}) u_k$$

6.1.2 Optimal Policies for Multi-Region Search

To derive the optimal policy, we consider the reduction in expected posterior differential entropy $H(p_n) - E[H(p_{n+1}) | \boldsymbol{B}_{n+1}, p_n]$ that results from collecting and processing measurements at time $n+1$ using a sensing partition $\boldsymbol{B}_{n+1}$ based on information state $p_n(x)$. The following theorem summarizes our result:

Theorem 6.1.1. *The expected reduction in posterior differential entropy from processing measurements collected by a sensing partition $\boldsymbol{B}_{n+1}$ is given in terms of the*

joint operating point $\boldsymbol{u}_{n+1} = (u^0_{n+1}, \ldots, u^K_{n+1})$ *in* (6.5)*, as*

$$\begin{aligned}\varphi(\boldsymbol{u}_{n+1}) &\equiv H(p_n) - E_{\boldsymbol{Y}_{n+1}}[H(p_{n+1})|\boldsymbol{B}_{n+1}, p_n] \\ &= \mathcal{H}(\sum_{k=0}^{K} f_k(\boldsymbol{y}) u^k_{n+1}) - \sum_{k=0}^{K} u^k_{n+1} \mathcal{H}(f_k(\boldsymbol{y}))\end{aligned}$$

where $\mathcal{H}$ *is the standard Shannon entropy for discrete-valued distributions.*

Proof. Let $\boldsymbol{B}_{n+1}$ denote the partition induced by $\boldsymbol{A}_{n+1}$. Define $\eta_0(\boldsymbol{y}, x) = P(\boldsymbol{Y}_{n+1} = \boldsymbol{y}|\boldsymbol{A}_{n+1}, X = x)$. Using Bayes' rule as in (6.2), we get

$$\begin{aligned}&E_{\boldsymbol{Y}_{n+1}}[H(p_{n+1})|\boldsymbol{B}_{n+1}, p_n] \\ =&-\sum_{\boldsymbol{y}\in\mathcal{Y}^M} \eta(\boldsymbol{y}) \Big(\int_{\mathcal{X}} [p_n(x)\frac{\eta_0(\boldsymbol{y}, x)}{\eta(\boldsymbol{y})}] \log[p_n(x)\frac{\eta_0(\boldsymbol{y}, x)}{\eta(\boldsymbol{y})}] dx \Big)\end{aligned}$$

$$\begin{aligned}=&-\int_{\mathcal{X}} p_n(x) \log p_n(x) dx \\ &+\sum_{\boldsymbol{y}\in\mathcal{Y}^M} \int_{\mathcal{X}} p_n(x)\eta_0(\boldsymbol{y}, x) \log \eta(\boldsymbol{y}) dx \\ &-\int_{\mathcal{X}} p_n(x) \sum_{\boldsymbol{y}\in\mathcal{Y}^M} \eta_0(\boldsymbol{y}, x) \log \eta_0(\boldsymbol{y}, x) dx \\ =&H(p_n) - \Big[\mathcal{H}(\eta(\boldsymbol{y})) - \sum_{k=1}^{K} \int_{\mathcal{X}} p_n(x)\mathcal{H}(f_k)\mathbb{1}_{\{x\in A^k_{n+1}\}} dx\Big] \\ =&H(p_n) - \Big[\mathcal{H}(\sum_{k=1}^{K} u^k_{n+1} f_k) - \sum_{k=1}^{K} u^k_{n+1}\mathcal{H}(f_k)\Big]\end{aligned}$$

■

One way of interpreting $\varphi(\boldsymbol{u}_{n+1})$ is to consider $\boldsymbol{u}_{n+1}$ as a probability distribution for the values of a discrete-valued random variable Z_{n+1}, with $P(Z_{n+1} = k) = u^k_{n+1}$. Then,

$$\varphi(\boldsymbol{u}_{n+1}) = I(\boldsymbol{Y}_{n+1}; Z_{n+1})$$

where the mutual information for two discrete-valued random variables is defined in terms of the Shannon entropy as $I(Y;Z) = \mathcal{H}(Y) - \mathcal{H}(Y|Z)$. This is readily established as

$$\begin{aligned} I(\boldsymbol{Y};Z) &= -\sum_{\boldsymbol{y}\in\mathcal{Y}^M}\sum_{z=0}^{K} P(\boldsymbol{y}|z)P(z)\log\Big[\sum_{z=0}^{K} P(\boldsymbol{y}|z)P(z)\Big] \\ &\quad + \sum_{z=0}^{K} P(z)\sum_{\boldsymbol{y}\in\mathcal{Y}^M} P(\boldsymbol{y}|z)\log P(\boldsymbol{y}|z) \\ &= -\sum_{\boldsymbol{y}\in\mathcal{Y}^M}\sum_{k=0}^{K} u^k f_k(\boldsymbol{y})\log\Big[\sum_{k=0}^{K} u^k f_k(\boldsymbol{y})\Big] \\ &\quad + \sum_{k=0}^{K} u^k \sum_{\boldsymbol{y}\in\mathcal{Y}^M} f_k(\boldsymbol{y})\log f_k(\boldsymbol{y}) \\ &= \mathcal{H}(\sum_{k=0}^{K} f_k(\boldsymbol{y})u^k) - \sum_{k=0}^{K} u^k \mathcal{H}(f_k(\boldsymbol{y})) \end{aligned}$$

Note that $\varphi(\boldsymbol{u}) = \varphi(u^0,\cdots,u^K)$ as defined in Theorem 6.1.1 is strictly concave over the simplex $\sum_{k=0}^{K} u^k = 1$, for $u^k \geq 0$, $k = 0,\cdots,K$. This follows from the strict concavity of the Shannon entropy $\mathcal{H}(f)$. Thus, it has a unique maximum value achieved at a unique point $\boldsymbol{u}^* = (u^{0*},\cdots,u^{K*})$. Any partition $\boldsymbol{B}_{n+1}$ for which the statistics in (6.5) are equal to $\boldsymbol{u}^*$ achieves the maximal differential entropy reduction at stage $n+1$. Note that the optimal operating point $\boldsymbol{u}^*$ does not depend on the posterior density $p_n(x)$ or the partition $\boldsymbol{B}_{n+1}$.

Next, we show that, for any operating point $\boldsymbol{u}^*$ and information state $p_n(x)$, there exists a sensing partition $\boldsymbol{B}_{n+1}$ for which $\boldsymbol{u}(\boldsymbol{B}_{n+1}, p_n) = \boldsymbol{u}^*$. Let d denote the dimension of the Euclidean space containing $\mathcal{X}$, and let $\boldsymbol{e}$ denote the d-dimensional vector of all ones. Since $p_n(x)$ corresponds to a distribution that is absolutely continuous, the cumulative distribution function $P_n(x) = \int_{-\infty}^{x}\cdots\int_{-\infty}^{x} p_n(x')dx'$ is continuous, and monotone non-decreasing on the diagonal $x = \alpha\boldsymbol{e}$, starting at 0 for $\alpha <= -C$, and increasing to 1 for $\alpha >= C$ for some C because of the compactness of

$\mathcal{X}$. Hence, for any u^{0*}, we can find a value a_0 so that $P_n(a_0\boldsymbol{e}) = u^{0*}$, and we can set $B^0_{n+1} = \{x \leq a_0\boldsymbol{e}\} \cap \mathcal{X}$, where the inequality is interpreted element wise. Similarly, for any u^{1*} such that $u^{0*}+u^{1*} \leq 1$, we can find $a_1 \geq a_0$ such that $P_n(a_1\boldsymbol{e}) - P_n(a_0\boldsymbol{e}) = u^{1*}$, and set $B^1_{n+1} = \{a_0\boldsymbol{e} < x \leq a_1\boldsymbol{e}\} \cap \mathcal{X}$. We continue this construction to obtain the final $a_K = C$, because $\sum_{k=0}^{K} u^{k*} = 1$. The final partition $\boldsymbol{B}_{n+1}$ so constructed satisfies $\boldsymbol{u}(\boldsymbol{B}_{n+1}, p_n) = \boldsymbol{u}^*$. Note that there are many other partitions that would also satisfy this equality, which implies that the optimal partition is not unique. This approach results in elements in the partition that are unions of a small number of rectangular regions, rather than arbitrary measurable subsets.

What remains is to show the optimal solution to the multi-stage policy optimization problem (6.3) can be constructed in terms of the above adaptive sensing policy.

Theorem 6.1.2. *Let* $(u^{0*}, \cdots, u^{K*}) = \arg\max_{\boldsymbol{u}=(u^0,\cdots,u^K)} \varphi(\boldsymbol{u})$ *for* $\varphi(\boldsymbol{u})$ *as defined in Theorem 6.1.1. For each stage* $n+1$*, select a sensing partition* $\boldsymbol{B}_{n+1}$ *that satisfies* $\boldsymbol{u}(\boldsymbol{B}_{n+1}, p_n) = \boldsymbol{u}^*$*. Based on this partition, select observation regions for each agent as* $A^m_{n+1} = \cup_{k:i_m=1} B^k_{n+1}$*, where* i_m *is the* $m-th$ *digit in the dyadic expansion of* k*. Then, this adaptive set of observation regions is optimal for problem* (6.3)*. Furthermore, the optimal value function is given by*

$$V(p_n, n) = H(p_n) - (N-n)\varphi^* \tag{6.6}$$

where the constant $\varphi^* = \varphi(u^{0*}, \cdots, u^{K*})$*.*

Proof. To establish this, we show that (6.6) satisfies the Bellman equation (6.4) and the above policy is a minimizing policy. The optimal value function is correct at stage N, as $V(p_N, N) = H(p_N)$. Assume by induction that the optimal value function satisfies (6.6) for all $k \geq n+1$. Then,

$$\begin{aligned}
V(p_n, n) &= \inf_{\boldsymbol{A}} E_{\boldsymbol{Y}_{n+1}}[V(p_{n+1}, n+1) | \boldsymbol{A}_{n+1} = \boldsymbol{A}, p_n] \\
&= \inf_{\boldsymbol{A}} E_{\boldsymbol{Y}_{n+1}}[H(p_{n+1}) - (N-n-1)\varphi^*) | \boldsymbol{A}_{n+1} = \boldsymbol{A}, p_n] \\
&= \inf_{\boldsymbol{A}} E_{\boldsymbol{Y}_{n+1}}[H(p_{n+1}) | \boldsymbol{A}_{n+1} = \boldsymbol{A}, p_n] - (N-n-1)\varphi^* \\
&= \inf_{\boldsymbol{B}} E_{\boldsymbol{Y}_{n+1}}[H(p_{n+1}) | \boldsymbol{B}_{n+1} = \boldsymbol{B}, p_n] - (N-n-1)\varphi^*
\end{aligned}$$

$$= H(p_n) - \sup_{\boldsymbol{B}} \left[\mathcal{H}(\sum_{k=1}^{K} u_{n+1}^k f_k) - \sum_{k=1}^{K} u_{n+1}^k \mathcal{H}(f_k) \right] - (N-n-1)\varphi^*$$

because of the equivalence of the partition $\boldsymbol{B}$ and the sensing areas $\boldsymbol{A}$, and the results of Theorem 6.1.1. Furthermore, we know that

$$\sup_{\boldsymbol{B}} \left[\mathcal{H}(\sum_{k=1}^{K} u_{n+1}^k f_k) - \sum_{k=1}^{K} u_{n+1}^k \mathcal{H}(f_k) \right] = \varphi^*$$

because, given p_n, we have provided a construction for choosing a partition $\boldsymbol{B}_{n+1}$ such that $p_n(B_{n+1}^k) = u^{k*}, k = 1, \ldots, K$. Thus,

$$V(p_n, n) = H(p_n) - (N-n)\varphi^*$$

and the supremum is achieved by the feedback strategy $\pi_{n+1}^*(p_n) = \boldsymbol{B}_{n+1}$. ■

We note at this point that the optimal single stage entropy reduction φ^* is equal to the information-theoretic channel capacity C of a memoryless communication channel with input the discrete variables Z and output the observations $\boldsymbol{Y}$: both quantities are defined by the same optimization problem.

Note the computational complexity of the solution: one must solve a strictly concave maximization problem in 2^M variables (which may be done off-line) to determine the joint operating point $\boldsymbol{u}^*$. The real-time computation requirements are to update $p_n(x)$, and to determine at time $n+1$, the partition $\boldsymbol{B}$ based on $p_n(x)$ so that $\boldsymbol{u}(\boldsymbol{B}_{n+1}, p_n) = \boldsymbol{u}^*$. The number of elements in the partition, 2^M, grows exponentially with the number of agents. In addition, a sampled form for the conditional density $p_n(x)$ will require a number of samples that grow exponentially with the underlying dimension of the search space.

An important property of the above solution is that the optimal feedback strategy does not depend on the length of the planning horizon N. Thus, the resulting strate-

gies are optimal for any duration of the planning horizon, yielding search algorithms that are anytime-optimal no matter when the search terminates.

The above results exploit several special structures of our adaptive control problem, as discussed below:

- The object location must have a prior distribution over a continuous region that is absolutely continuous with respect to Lebesgue measure. This leads to conditional cumulative probability distributions that are continuous, and enable us to construct strategies that satisfy the optimality conditions. This would not be the case if the potential object locations had distributions that were not absolutely continuous.

- The differential entropy objective function allows for separability of the contribution of new information from past information, a critical step in the development of optimality conditions. Replacing the objective by functions such as Rényi entropy or other similar divergence measures requires additional conditions to guarantee concavity as well as existence of minimizing strategies.

- The measurement error models do not depend on the size of the regions used in the partitions at each stage, and depend on X only through the indicator that X is in particular regions.

There are special cases where the optimal operating point is known explicitly. One such case is when the measurement error model satisfies a special symmetry condition. The error model from Z to $\boldsymbol{Y}$ is modeled as a noisy discrete memoryless channel because of the conditional independence across time. Such channels are said to be quasi-symmetric when the set of outputs $\mathcal{Y}^M$ can be partitioned into subsets $\mathcal{W}^\ell$ such that, for each subset, the sub-transition probability matrices $P(\boldsymbol{y}|z)$ for $y \in \mathcal{W}^\ell, z \in \{0, \ldots, K\}$ satisfy the property that each row is a permutation of every

other row, and each column sums up to the same subset-dependent constant. When this channel has the property of quasi-symmetry [Alajaji and Chen, 2015], or the property of symmetry as defined in [Gallager, 1968], the optimal operating point satisfies $(u^{0*}, \cdots, u^{K*}) = (\frac{1}{K+1}, \ldots, \frac{1}{K+1})$ ([Gallager, 1968], Thm 4.5.2).

6.1.3 Bounds on Mean-Square Error

From Theorem 6.1.2, the maximal expected posterior entropy reduction is $n\varphi^*$ after n sensing stages are completed, where φ^* is defined in Theorem 6.1.2. This allows us to give a lower bound on the performance of the minimum mean-square error estimator, following the framework established in Theorem 4 in [Tsiligkaridis et al., 2014]:

Theorem 6.1.3. *Assume $H(p_0)$ is finite. Then, the minimum mean-square error estimator at stage n $\hat{X}_n = \int_{\mathcal{X}} x p_n(x) dx$ under any admissible policy has the following mean-square error lower bound:*

$$E[||X - \hat{X}_n||_2^2] \geq \frac{d\sqrt[d]{C_0}}{2\pi e} 2^{-\frac{2n\varphi^*}{d}}$$

where d is the dimension of the object space and $C_0 = 2^{2H(p_0)}$, and φ^* is defined in Theorem 6.1.2.

Proof. Let $\hat{X}_n = \int_{\mathcal{X}} x p_n(x) dx$ and $\Sigma_n = E[(X - \hat{X}_n)(X - \hat{X}_n)^T]$. By Theorem 17.2.3 in [Cover and Thomas, 2012] and Jensen's inequality, under any policy ζ, we have

$$\begin{aligned} E_\zeta[H(p_n)] \leq & E_\zeta\Big[\frac{1}{2}\log((2\pi e)^d \det(\Sigma_n))\Big] \\ \leq & \frac{1}{2}\log(2\pi e)^d + \frac{1}{2}\log(\det(E_\zeta[\Sigma_n])) \\ = & \frac{1}{2}\log((2\pi e)^d \det(E_\zeta[\Sigma_n])) \end{aligned}$$

where $\det(\cdot)$ denotes the matrix determinant. From Theorem 6.1.2, under any policy ζ, we have $E_\zeta[H(p_n)] \geq H(p_0) - n\varphi^*$. By letting $C_0 = 2^{2H(p_0)}$, we have

$$\frac{C_0 2^{-2n\varphi^*}}{(2\pi e)^d} \leq \frac{2^{2E_\zeta[H(p_n)]}}{(2\pi e)^d} \leq \det(E_\zeta[\Sigma_n])$$

Since the determinant and the trace of a square matrix can be written as the product and the sum of the eigenvalues of the matrix respectively, using the inequality of arithmetic and geometric means we have

$$\det(E_\zeta[\Sigma_n]) \leq \Big(\frac{E_\zeta[\mathrm{tr}(\Sigma_n)]}{d}\Big)^d$$

where $\mathrm{tr}(\cdot)$ denotes the matrix trace.

Combining and rewriting the inequalities, we get

$$E[||X - \hat{X}_n||_2^2] = E_\zeta[\mathrm{tr}(\Sigma_n)] \geq \frac{d\sqrt[d]{C_0}}{2\pi e} 2^{-\frac{2n\varphi^*}{d}}$$

■

This bound is tighter than the bound in [Tsiligkaridis et al., 2014] for the independent error case, as it uses smaller exponents and constants. The lower bound decays exponentially with the number of stages, at a rate that is proportional to the maximal one-stage expected entropy reduction φ^*.

This lower bound applies to any search strategy, including optimal search strategies that satisfy the conditions of Theorem 6.1.2. Obtaining an upper bound will depend on the specific choice of optimal strategy. Finding such upper bounds is closely related to finding bounds on the mean squared error of communication channels with perfect feedback. Unfortunately, most of the available bounds apply to coding schemes that do not satisfy the optimality conditions in our problem. The results of [Tsiligkaridis et al., 2014] cite the bound for the algorithm of [Burnashev and Zigangirov, 1974], as described in [Castro and Nowak, 2008]. However, that algorithm is a heuristic that is not optimal for our problem, or for any of the problems discussed in [Tsiligkaridis et al., 2014].

To our knowledge, the only upper bound on the expected error of an optimal algorithm was published recently in [Waeber et al., 2011], where they derived a bound on the probability of error of Horstein's original probabilistic bisection algorithm

[Horstein, 1963]. This algorithm corresponds to the single-agent case, searching for an object in a one-dimensional interval, with binary measurements and symmetric error model. For this case, the symmetry of the error model results in $u^{0*} = u^{1*} = 0.5$, and our construction selects an observation region A that is delineated by the median of the distribution defined by the density $p_{n-1}(x)$, which is the probabilistic bisection algorithm. We summarize the bound in Theorem 5.1 in [Waeber et al., 2011] below:

Theorem 6.1.4. *([Waeber et al., 2011]) There exists a constant $c(p) > 1$ such that*

$$E[|X - X_n|] \in o(c(p)^{-n})$$

where X_n is the median of the conditional distribution defined by $p_{n-1}(x)$ and $1 - p$ is the probability of error in the binary symmetric channel.

Although the above result is for the median estimator and the error magnitude, a similar exponential bound is easily derived for the mean-square error, as the proof shows that the conditional probability density is concentrating on the point X. To illustrate these bounds, we simulate the performance of a single agent with binary symmetric errors, with probability of error 0.1, using 1000 Monte Carlo runs, for an object X with uniform initial distribution on the unit interval $[0, 1]$. Figure 6·1 shows the decay of the upper and lower bounds, compared with the simulated mean squared errors computed using the optimal algorithm. While both bounds decay exponentially, the lower bound is a closer approximation of the actual mean squared error.

It is straightforward to extend the upper bound to multiple agents acting in sequential order, so that only one agent collects a measurement at each time. However, extension of the upper bound of [Waeber et al., 2011] to single agents with asymmetric channels or to multiple agents with simultaneous measurements remains an unsolved problem. In our experiment results later in the paper, we compute the empirical mean-square error and the lower bound to illustrate the tightness of the bound.

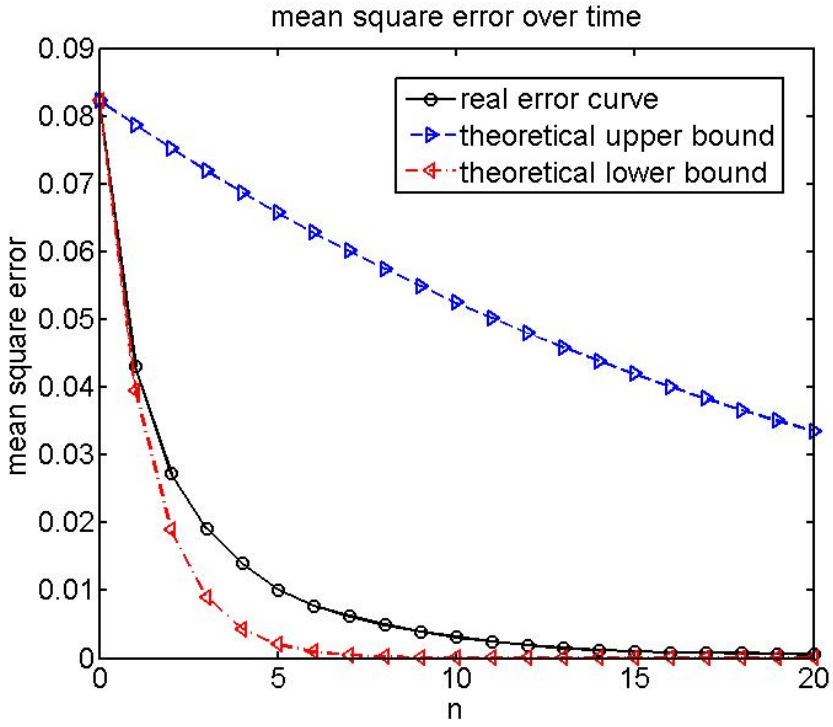


Figure 6·1: Empirical mean squared error for single agent case, with upper and lower bounds

6.2 Independent Error Models

As a special case of the results in Section 6.1, we consider the case where the noisy measurements collected by the M agents are conditionally independent across agents and across times given the true object location X and the sensing regions $A_n^m, n = 1, \ldots, N, m = 1, \ldots, M$. This makes the error channels from the true indicators Z_n^m to the measurements Y_n^m independent memoryless channels. The conditional independence implies the following statistical model for our measurements, as in (6.1):

$$P(\boldsymbol{Y}_n = \boldsymbol{y}|X, \boldsymbol{A}_n) = \prod_{m=1}^{M} P(Y_n^m = y^m|Z_n^m) \tag{6.7}$$

To simplify the exposition, define the measurement probability distributions

$$P(Y_n^m = y|Z_n^m = k) = f_k^m(y),\ y \in \mathcal{Y},\ k \in \{0, 1\}$$

Based on the conditional independence assumptions, the conditional density of the joint measurements given the indicator variables $Z^1 = i_1, \ldots, Z^M = i_M$ associated with the sensing areas $A^1, \ldots, A^M$ and the state X, is given by

$$P(\boldsymbol{Y}_n = \boldsymbol{y}|X, \boldsymbol{A}_n) \equiv q_{i_{M:1}}(\boldsymbol{y}) = \prod_{m=1}^{M} f_{i_m}^m(y^m) \tag{6.8}$$

where we use $i_{M:1}$ as a shorthand for $(i_M, \ldots, i_1)$.

This simplifies the dynamics for the conditional density $p_n(x)$ as defined by Bayes' rule (6.2) as:

$$\begin{aligned} p_{n+1}(x) &= p_n(x) \frac{P(\boldsymbol{Y}_{n+1} = \boldsymbol{y}|\boldsymbol{A}_{n+1}, X = x)}{\int_{\mathcal{X}} p_n(\sigma) P(\boldsymbol{Y}_{n+1} = \boldsymbol{y}|\boldsymbol{A}_{n+1}, X = \sigma) d\sigma} \\ &= \frac{p_n(x) \sum_{i_{M:1} \in \{0,1\}^M} q_{i_{M:1}}(\boldsymbol{y}) \mathbb{1}_{\{x \in \cap_{m=1}^M (A_n^m)^{i_m}\}}}{\int_{\mathcal{X}} p_n(\sigma) \sum_{i_{M:1} \in \{0,1\}^M} q_{i_{M:1}}(\boldsymbol{y}) \mathbb{1}_{\{\sigma \in \cap_{m=1}^M (A_n^m)^{i_m}\}} d\sigma} \end{aligned} \tag{6.9}$$

where we use the notation $(A)^0 \equiv A^c$ and $(A)^1 \equiv A$ if A is a subset of $\mathcal{X}$.

In this case, the joint operating point of (6.5), denoted as $\boldsymbol{u} = \{u_{i_{M:1}}, i_1, \ldots, i_M \in \{0,1\}\}$, is

$$u_{i_{M:1}} = \int\limits_{\cap_{m=1}^{M}(A^m)^{i_m}} p_n(\sigma)d\sigma \ \geq 0 \tag{6.10}$$

Then, $P(\boldsymbol{Y}_{n+1} = \boldsymbol{y}|\boldsymbol{A}_{n+1}, p_n) \equiv \eta(\boldsymbol{y})$ is given by

$$\eta(\boldsymbol{y}) = \sum_{i_{M:1}\in\{0,1\}^M} q_{i_{M:1}}(y^1, \ldots, y^M)u_{i_{M:1}}$$

The expected entropy reduction of a collection of observation areas $\boldsymbol{A}_{n+1}$ is now given by:

$$\begin{aligned}\varphi(\boldsymbol{u}) &\equiv H(p_n) - E[H(p_{n+1})|\boldsymbol{A}_{n+1}, p_n] \\ &= \mathcal{H}(\sum_{i_{M:1}\in\{0,1\}^M} q_{i_{M:1}}u_{i_{M:1}}) - \sum_{i_{M:1}\in\{0,1\}^M} u_{i_{M:1}}\mathcal{H}(q_{i_{M:1}})\end{aligned} \tag{6.11}$$

where $q_{i_{M:1}}(y^1, \ldots, y^M), \boldsymbol{u}$ are defined in (6.8), (6.10).

The results of Theorem 6.1.2 guarantee that there is a unique joint operating point $\boldsymbol{u}^* = \{u^*_{i_{M:1}}\}$ that maximizes $\varphi(\boldsymbol{u})$. Furthermore, we can always find a joint sensing strategy that achieves this optimal value. This implies the optimal value function is given by

$$V(p_n, n) = H(p_n) - (N-n)\varphi^* \tag{6.12}$$

and the optimal decisions at stage n could be computed via a joint strictly concave maximization problem for $\boldsymbol{u}^*$, followed by a region allocation problem to determine $\{A_n^{1*}, \ldots, A_n^{M*}\}$ based on $p_{n-1}(x)$. We show next that, for the special case of independent error models considered in this section, the optimization problem for $\boldsymbol{u}^*$ can be decoupled to the solution of M scalar optimization problems, greatly reducing the

computation complexity.

Consider the problem when there is only one agent present, as in [Jedynak et al., 2011]. Define $\varphi^m(u)$ to be the single agent expected differential entropy reduction that agent m can achieve on its own by selecting its sensing area A^m such that $p_n(A^m) = u$, as

$$\begin{aligned}\varphi^m(u) = &\mathcal{H}(uf_1^m + (1-u)f_0^m) \\ &- u\mathcal{H}(f_1^m) - (1-u)\mathcal{H}(f_0^m)\end{aligned}$$

and define the maximum expected entropy reduction as

$$\varphi^{m*} = \max_u \varphi^m(u)$$

Theorem 6.2.1. *Consider general discrete-output agent error models. Denote the optimal operating points of each individual agent m as $u^{m*} = \arg\max_{u^m} \varphi^m(u^m)$, $m = 1, \ldots, M$. Then the optimal operating point for joint sensing, i.e., $\boldsymbol{u}^* = \{u^*_{i_{M:1}}\} = \arg\max_{\boldsymbol{u}} \varphi(\boldsymbol{u})$, is given by*

$$u^*_{i_{M:1}} = \prod_{m=1}^{M} (u^{m*})^{i_m} (1-u^{m*})^{1-i_m} \tag{6.13}$$

In addition, $\varphi^* = \sum_{m=1}^{M} \varphi^{m*}$.

Proof. We first prove that $\varphi^* \leq \sum_{m=1}^{M} \varphi^{m*}$:

$$\varphi^* = \mathcal{H}\Big(\sum_{i_{M:1} \in \{0,1\}^M} u^*_{i_{M:1}} q_{i_{M:1}}\Big) - \sum_{i_{M:1} \in \{0,1\}^M} u^*_{i_{M:1}} \mathcal{H}(q_{i_{M:1}})$$

From the additivity property of the Shannon entropy, we have:

$$\mathcal{H}(q_{i_{M:1}}) = \sum_{m=1}^{M} (\mathcal{H}(f_1^m)\mathbb{1}_{\{i_m=1\}} + \mathcal{H}(f_0^m)\mathbb{1}_{\{i_m=0\}})$$

$$\sum_{i_{M:1}\in\{0,1\}^M} u^*_{i_{M:1}}\mathcal{H}(q_{i_{M:1}}) = \sum_{m=1}^{M}\Big[\sum_{i_{M:1}:i_m=1} u^*_{i_{M:1}}\mathcal{H}(f_1^m) + \sum_{i_{M:1}:i_m=0} u^*_{i_{M:1}}\mathcal{H}(f_0^m)\Big]$$

Similarly, note that the term $\sum_{i_{M:1}\in\{0,1\}^M} u^*_{i_{M:1}} q_{i_{M:1}}$ specifies a joint probability distribution for the variables $Y^1,\ldots,Y^M$, with marginal probability distribution for each variable Y^m given by

$$g^m(y) = \sum_{i_{M:1}:i_m=1} u^*_{i_{M:1}} f_1^m(y) + \sum_{i_{M:1}:i_m=0} u^*_{i_{M:1}} f_0^m(y)$$

Combining these relations and using the subadditivity property of the Shannon entropy, we obtain

$$\begin{aligned}\varphi^* \le &\sum_{m=1}^{M}\mathcal{H}(g^m) - \sum_{m=1}^{M}\Big[\sum_{i_{M:1}:i_m=1} u^*_{i_{M:1}}\mathcal{H}(f_1^m)\\ &+ \sum_{i_{M:1}:i_m=0} u^*_{i_{M:1}}\mathcal{H}(f_0^m)\Big]\end{aligned}$$

Note that the numbers $a^m = \sum_{i_{M:1}:i_m=1} u^*_{i_{M:1}}$ and $b^m = \sum_{i_{M:1}:i_m=0} u^*_{i_{M:1}}$ are non-negative and sum up to 1, and thus represent a possible operating point for agent m. Since $(u^{m*}, 1-u^{m*})$ is the optimal operating point that maximizes $\varphi^m(u)$, we have

$$\begin{aligned}\varphi^* \le &\sum_{m=1}^{M}\mathcal{H}(g^m) - \sum_{m=1}^{M}\Big[\sum_{i_{M:1}:i_m=1} u^*_{i_{M:1}}\mathcal{H}(f_1^m)\\ &+ \sum_{i_{M:1}:i_m=0} u^*_{i_{M:1}}\mathcal{H}(f_0^m)\Big]\\ = &\sum_{m=1}^{M}\varphi^m(a^m) \le \sum_{m=1}^{M}\varphi^{m*}\end{aligned}$$

Given u^{m*}, define

$$\overline{u}_{i_{M:1}} = \prod_{m=1}^{M}(u^{m*})^{i_m}(1-u^{m*})^{1-i_m}$$

Note that $\sum_{i_{M:1}\in\{0,1\}^M}\overline{u}_{i_{M:1}} = 1$, so this is a valid joint operating point $\boldsymbol{u}$ for the multi-agent problem. Then,

$$\varphi(\overline{\boldsymbol{u}}) = \mathcal{H}(\sum_{i_{M:1}\in\{0,1\}^M} \overline{u}_{i_{M:1}} q_{i_{M:1}}) - \sum_{i_{M:1}\in\{0,1\}^M} \overline{u}_{i_{M:1}} \mathcal{H}(q_{i_{M:1}})$$

$$= \mathcal{H}(\sum_{i_{M:1}\in\{0,1\}^M} \prod_{m=1}^{M} (u^{m*})^{i_m}(1-u^{m*})^{1-i_m} q_{i_{M:1}})$$

$$- \sum_{i_{M:1}\in\{0,1\}^M} (\prod_{m=1}^{M} (u^{m*})^{i_m}(1-u^{m*})^{1-i_m}) \mathcal{H}(q_{i_{M:1}})$$

Since

$$\sum_{i_{M:1}\in\{0,1\}^M} (\prod_{m=1}^{M} (u^{m*})^{i_m}(1-u^{m*})^{1-i_m}) q_{i_{M:1}}$$

$$= \sum_{i_{M:1}\in\{0,1\}^M} \prod_{m=1}^{M} [(u^{m*})^{i_m}(1-u^{m*})^{1-i_m} f_{i_m}^m]$$

$$= \prod_{m=1}^{M} [u^{m*} f_1^m + (1-u^{m*}) f_0^m]$$

$$\sum_{i_{M:1}\in\{0,1\}^M} (\prod_{m=1}^{M} (u^{m*})^{i_m}(1-u^{m*})^{1-i_m}) \mathcal{H}(q_{i_{M:1}})$$

$$= \sum_{i_{M:1}\in\{0,1\}^M} (\prod_{m=1}^{M} (u^{m*})^{i_m}(1-u^{m*})^{1-i_m}) \sum_{k=1}^{M} \mathcal{H}(f_{i_k}^k)$$

$$= \sum_{k=1}^{M} \sum_{i_{M:1}\in\{0,1\}^M} (\prod_{m=1}^{M} (u^{m*})^{i_m}(1-u^{m*})^{1-i_m}) \mathcal{H}(f_{i_k}^k)$$

$$= \sum_{k=1}^{M} [u^{k*} \mathcal{H}(f_1^k) + (1-u^{k*}) \mathcal{H}(f_0^k)]$$

Thus,

$$\varphi(\overline{\boldsymbol{u}}) = \mathcal{H}\Big(\prod_{m=1}^{M} [\sum_{j=0}^{1} f_j^m (u^{m*})^j (1-u^{m*})^{1-j}] \Big)$$

$$- \sum_{m=1}^{M} \sum_{j=0}^{1} (u^{m*})^j (1-u^{m*})^{1-j} \mathcal{H}(f_j^m)$$

$$
\begin{aligned}
&= \sum_{m=1}^{M} \Big[\mathcal{H}\Big(u^{m*} f_1^m + (1-u^{m*}) f_0^m \Big) \\
&\quad - u^{m*} \mathcal{H}(f_1^m) - (1-u^{m*}) \mathcal{H}(f_0^m) \Big] \\
&= \sum_{m=1}^{M} \varphi^{m*}
\end{aligned}
$$

Since $\varphi^* = \max_{\boldsymbol{u}} \varphi(\boldsymbol{u}) \leq \sum_{m=1}^{M} \varphi^{m*}$ and $\varphi(\boldsymbol{u})$ has a unique optimal point, selecting $\boldsymbol{u}$ as (6.13) will give us the optimal operating point for $\varphi(\boldsymbol{u})$ and we have $\varphi^* = \sum_{m=1}^{M} \varphi^{m*}$. ■

Thus, the optimal joint operating point for the multiple Boolean agent case with discrete measurements can be obtained from the optimal single-agent operating points. Furthermore, we can now use the construction of Section 6.1.2 to obtain partitions of the region $\mathcal{X}$ that achieve the probabilities required by the joint operating point, and combine them to obtain the joint optimal sensing areas for each agent $m \in \{1, \ldots, M\}$.

One way to understand the results of Theorem 6.2.1 is to connect the maximal entropy reduction at each stage to the concept of channel capacity, as was done in [Tsiligkaridis et al., 2014]. Each agent m can be viewed as a discrete memoryless stationary channel whose input is $Z^m \in \{0, 1\}$, output is $Y^m \in \mathcal{Y}$, and transition probabilities are specified by f_1^m and f_0^m. Furthermore, we can regard $q_{i_{M:1}}(y^1, \ldots, y^M)$ defined in (6.8) as the transition probabilities of a "mixed" vector (product) channel for all the M agents used in joint sensing, shown in Figure 6·2. The inputs of this mixed channel are vector $(Z^1, \ldots, Z^M) \in \{0, 1\}^M$, and the outputs are $(Y^1, \ldots, Y^M) \in \mathcal{Y}^M$. The capacity of this channel is equal to φ^*, the solution of our optimization problem in Theorem 6.2.1.

The results of Theorem 6.2.1 can also be used to identify an equivalence between joint sensing and sequential sensing for Boolean agents with discrete measurements, similar to the results in [Tsiligkaridis et al., 2014] for agents with binary measure-

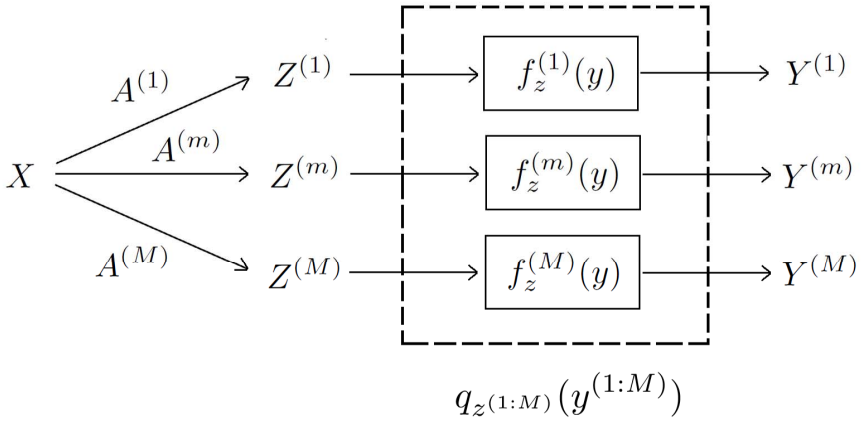


Figure 6·2: The "mixed" channel.

ments. As in [Tsiligkaridis et al., 2014], a sequential sensing scheme divides each stage into M sub-stages. At each substage m, the m-th agent selects a sensing area based on information state $p_{n,m-1}$, where $p_{n,0} \equiv p_n(x)$, and collects its noisy measurement. This measurement is processed to obtain an updated probability density for the object location $p_{n,m}(x)$. This information is made available to the next agent $m+1$, which in turn selects its query based on $p_{n,m}(x)$. The stage completes when the M-th agent collects its measurement, and uses it to update $p_{n,M-1}(x)$ to produce $p_{n+1}(x)$.

From [Jedynak et al., 2011] we know that the optimal policies are the ones that select a sensing area to reduce maximally the expected posterior entropy in each single substage. Thus, the optimal expected differential entropy reduction for the sequential policy at the end of one cycle is precisely $\sum_{m=1}^{M} \varphi^m(u^m)$, which is the same value φ^* that would be achieved by the joint sensing scheme in Theorem 6.2.1. This establishes the following lemma for general discrete observation Boolean agents.

Lemma 6.2.2. *Consider general discrete-output agent error models for both the sequential and joint sensing models described above. Then, the optimal expected posterior entropy achievable at the end of a stage is the same for both sequential and joint sensing.*

There is a further simplification where the optimal operating points for the joint sensing problem can be computed analytically. The results in [Tsiligkaridis et al., 2014] show that, when each of the M agents has a binary symmetric error model, the optimal operating points are $u^*_{i_{M:1}} = \frac{1}{2^M}$, and the optimal individual agent operating points are $u^{m*} = 1/2$. We extend this further to non-binary error models where $|\mathcal{Y}| \geq 2$. Specifically, we consider the case of Boolean agents with error models that correspond to quasi-symmetric memoryless channels [Alajaji and Chen, 2015] (also called symmetric discrete memoryless channels in [Gallager, 1968]). For Boolean agents, quasi-symmetric error models satisfy the following condition:

Definition 6.2.3 (Symmetry condition)**.** *A Boolean agent has a quasi-symmetric error model if there exists a permutation $\chi() : \mathcal{Y} \to \mathcal{Y}$ such that, for all $y \in \mathcal{Y}$, $f_1(y) = f_0(\chi(y))$ and $f_0(y) = f_1(\chi(y))$.*

The implications of having quasi-symmetric error models are:

$$\mathcal{H}(f_1) = \mathcal{H}(f_0)$$
$$\sum_{y\in\mathcal{Y}} f_1(y) \log \frac{f_1(y) + f_0(y)}{2} = \sum_{y\in\mathcal{Y}} f_0(y) \log \frac{f_1(y) + f_0(y)}{2}$$

These lead to the well-known result [Alajaji and Chen, 2015, Gallager, 1968] that the optimal capacity for quasi-symmetric discrete memoryless channels is achieved by a uniform input distribution. For Boolean agents, this means that the optimal sensing area at stage n with information state p_{n-1} is to pick a subset $A_n \in \mathcal{X}$ such that $\int_{x\in A_n} p_{n-1}(x)dx = 1/2$. This establishes the following result as a direct application of Theorem 6.2.1:

Corollary 6.2.4. *If we have M Boolean agents with quasi-symmetric (albeit different) error models, then the maximum value of $\varphi(\boldsymbol{u})$ occurs at $u^*_{i_{M:1}} = 2^{-M}$.*

When measurement values $\mathcal{Y}$ are a subset of the real line, $f_0(y), f_1(y)$ are given probability densities. A sufficient condition for a single Boolean agent to achieve its optimal capacity with uniform input distribution $u = 0.5, 1-u = 0.5$ is that there exist some constant α such that $f_0(y) = f_1(\alpha - y)$ for all y. These conditions are satisfied when the error models correspond to additive white Gaussian channels, so that

$$Y_n = h(Z_n) + w_n$$

where $h(Z_n)$ is a function of the Bernoulli variable Z_n and w_n is zero-mean with Gaussian distribution.

6.3 Independent Error Models with Precision Mode Selection

In [Sznitman et al., 2013], Sznitman et al. generalized [Jedynak et al., 2011] to the setting where at each sensing stage, the Boolean agent is allowed to choose a *precision mode* from a finite number of precision modes, in addition to its observed subset. Different precision modes will trigger different agent error models, but there are also costs associated with the precision modes. Precision modes with better error models will cost more to use, but may result in greater reduction in differential entropy. In this section, we extend our results to the Boolean multi-agent joint search problem with independent errors, allowing each agent to choose its precision mode at each stage.

Assume that for the m-th agent, the set of its possible precision modes is $\mathcal{L}^m = \{1, \ldots, L^m\}$. A joint search decision at stage n consists of selecting both sensing areas and precision modes for all M agents $(A_n^1, l_n^1, \ldots, A_n^M, l_n^M)$, where $A_n^m \subset \mathcal{X}$ and

$l_n^m \in \mathcal{L}^m$ for $m = 1, \ldots, M$.

The corresponding error models of the m-th agent under precision mode $l_n^m = l$ is characterized by $f_1^{m,l}$ and $f_0^{m,l}$, defined as:

$$P(Y_n^m = y | A_n^m = A, l_n^m = l, X) = \begin{cases} f_1^{m,l}(y) & X \in A \\ f_0^{m,l}(y) & X \notin A \end{cases}$$

When agent m selects a precision mode $l_n^m = l$, it incurs a cost $W^m(l)$. Note that this cost depends on the agent index m and the precision mode l, but does not depend on the sensing area A or the time instance n (although this last restriction can be easily relaxed).

Define the information state $p_n(x) = p(x|(\boldsymbol{A}_1, \boldsymbol{l}_1), \boldsymbol{Y}_1, \ldots, (\boldsymbol{A}_n, \boldsymbol{l}_n), \boldsymbol{Y}_n)$. The state evolves according to Bayes' rule as before, where the choices of precision modes are used to select the appropriate likelihoods for interpreting the observed measurements $\boldsymbol{Y}$. A joint sensing policy $\pi = (\pi_1, \ldots, \pi_N)$ with precision modes selection is a sequence of functions that map the information state $p_{n-1}(x)$ to admissible batch sensing and precision modes $(\boldsymbol{A}_n, \boldsymbol{l}_n) = (A_n^1, l_n^1, \ldots, A_n^M, l_n^M)$ at each stage n. Denote the policy space by Π.

Our joint sensing objective is to minimize the sum of the final-stage expected posterior differential entropy and the total cost of all agents discounted by a factor γ. The resulting optimization problem is :

$$\inf_{\pi \in \Pi} E[H(p_N) + \gamma \sum_{t=1}^{N} \sum_{m=1}^{M} W^m(l_t^m) | p_0]$$

This stochastic control problem fits the countable disturbance model of [Bertsekas and Shreve, 1978]. Define the optimal value function $V(p_n, n)$ at stage n to be:

$$V(p_n, n) = \inf_{\pi_{n+1}, \ldots, \pi_N} E[H(p_N) +$$

$$\gamma \sum_{t=n+1}^{N} \sum_{m=1}^{M} W^m(l_t^m)|p_n]$$

The Bellman equation for this problem is:

$$V(p_n, n) = \inf_{\boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1}} E_{\mathbf{Y}_{n+1}}[V(p_{n+1}, n+1) + \gamma \sum_{m=1}^{M} W^m(l_{n+1}^m)|(\boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1}), p_n]$$

If a policy $\pi \in \Pi$ achieves equality in the Bellman equation, it is an optimal policy.

Following our previous approach, consider a set of sensing decisions $(\boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1})$ at stage $n+1$. Define the joint operating point $\boldsymbol{u} = \{u_{i_{M:1}}, i_1, \ldots, i_M \in \{0,1\}\}$ as

$$u_{i_{M:1}} = \int_{\cap_{m=1}^{M} (A^m)^{i_m}} p_n(\sigma) d\sigma \geq 0 \tag{6.14}$$

For a given set of sensing decisions at stage $n+1$, consider the one-stage gain to be the expected reduction in differential entropy minus the cost of the precision modes by the agents, as

$$\hat{G}(p_n, \boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1}) \equiv \mathcal{H}(p_n) - E_{\mathbf{Y}_{n+1}}[\mathcal{H}(p_{n+1}) + \gamma \sum_{m=1}^{M} W^m(l_{n+1}^m)|(\boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1}), p_n]$$

We have the following characterization:

Lemma 6.3.1. *Define the function*

$$G(\boldsymbol{u}, \boldsymbol{l}) = \mathcal{H}(\sum_{i_{M:1} \in \{0,1\}^M} q_{i_{M:1}}^{\boldsymbol{l}} u_{i_{M:1}}) - \sum_{i_{M:1} \in \{0,1\}^M} u_{i_{M:1}} \mathcal{H}(q_{i_{M:1}}^{\boldsymbol{l}}) - \gamma \sum_{m=1}^{M} W^m(l^m) \tag{6.15}$$

where $\boldsymbol{u}(\cdot)$ *is defined from* p_n *and* $\boldsymbol{A}$ *as in* (6.14) *and*

$$q_{i_{M:1}}^{\boldsymbol{l}}(y^1, \ldots, y^M) = \prod_{m=1}^{M} f_{i_m}^{m,l^m}(y^m), y^m \in \mathcal{Y}$$

Then, for all sensing decisions, the one-stage gain can be expressed as

$$\hat{G}(p_n, \boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1}) \equiv G(\boldsymbol{u}_{n+1}, \boldsymbol{l}_{n+1})$$

The proof is straightforward, and follows the same arguments as Theorem 6.1.1. The dependence of the one stage gain on the information state and the selected sensing areas is summarized by the joint operating point $\boldsymbol{u}$ and the selected precision modes $\boldsymbol{l}$. This structure is exploited to derive the main result:

Theorem 6.3.2. *Consider the problem of finding the optimal sensing areas and precision modes for the Boolean multiagent problem with independent errors. Define* $(\boldsymbol{u}^*, \boldsymbol{l}^*) \in \arg\max_{\boldsymbol{u},\boldsymbol{l}} G(\boldsymbol{u}, \boldsymbol{l})$*. Then, at stage n, any policy that chooses batch sensing and precision modes* $(\boldsymbol{A}_n, \boldsymbol{l}_n)$ *such that* $\boldsymbol{l}_n = \boldsymbol{l}^* = (l^{1*}, \ldots, l^{M*})$ *and* $\boldsymbol{u}(\boldsymbol{A}_n, p_{n-1}) = \boldsymbol{u}^*$ *is optimal.*

Furthermore, the optimal value function has the following closed-form expression:

$$V(p_n, n) = H(p_n) - (N - n)G^*$$

where $G^* = G(\boldsymbol{u}^*, \boldsymbol{l}^*)$*.*

Proof. The optimal value function $V(p_N, N) = H(p_N) - (N - N)G^* = H(p_N)$ satisfies the hypothesized form at the terminal time N. For $n < N$, we have:

$$\begin{aligned} V(p_n, n) = \inf_{\boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1}} & E_{\mathbf{Y}_{n+1}}[V(p_{n+1}, n+1) \\ & + \gamma \sum_{m=1}^{M} W^m(l^m_{n+1}) | (\boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1}), p_n] \end{aligned}$$

Assume inductively that $V(p_{n+1}, n+1) = H(p_{n+1}) + (N - n - 1)G^*$; then,

$$\begin{aligned} & E_{\mathbf{Y}_{n+1}}[V(p_{n+1}, n+1) + \\ & \gamma \sum_{m=1}^{M} W^m(l^m_{n+1}) | (\boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1}), p_n] \\ = & E_{\mathbf{Y}_{n+1}}[H(p_{n+1}) + (N - n - 1)G^* \\ & + \gamma \sum_{m=1}^{M} W^m(l^m_{n+1}) | (\boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1}), p_n] \end{aligned}$$

$$= H(p_n) - G(\boldsymbol{u}_{n+1}, \boldsymbol{l}_{n+1}) + (N-n-1)G^*$$

by Lemma 6.3.1.

For fixed $\boldsymbol{l}$, $G(\boldsymbol{u}, \boldsymbol{l})$ is strictly concave over $\sum_{i_1=0}^{1} \cdots \sum_{i_M=0}^{1} u_{i_{M:1}} = 1$ due to the strict concavity of the Shannon entropy. Thus, $G^* = \sup_{\boldsymbol{u},\boldsymbol{l}} G(\boldsymbol{u}, \boldsymbol{l})$ exists. Furthermore, we know from the discussion after Theorem 6.1.1 that, for any density $p_n(x)$ and desired joint operating point $\boldsymbol{u}$, there exists joint sensing areas $\boldsymbol{A}_{n+1}$ such that the probabilities $\boldsymbol{u}(\boldsymbol{A}_{n+1}, p_n) = \boldsymbol{u}$. Thus,

$$\begin{aligned} V(p_n, n) &= \inf_{\boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1}} [H(p_n) - G(\boldsymbol{u}_{n+1}, \boldsymbol{l}_{n+1}) \\ &\quad + (N-n-1)G^*] \\ &= H(p_n) - \sup_{\boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1}} G(\boldsymbol{u}_{n+1}, \boldsymbol{l}_{n+1}) - (N-n-1)G^* \\ &= H(p_n) - (N-n)G^* \end{aligned}$$

Note that $\sup_{\boldsymbol{u},\boldsymbol{l}} G(\boldsymbol{u}, \boldsymbol{l}) = \max_{\boldsymbol{u},\boldsymbol{l}} G(\boldsymbol{u}, \boldsymbol{l})$ is achieved at some $(\boldsymbol{u}^*, \boldsymbol{l}^*)$ because it is the maximum of a finite number of strictly concave functions defined over the compact M-dimensional simplex. Thus, the optimal strategies are given by any $(\boldsymbol{A}_{n+1}, \boldsymbol{l}_{n+1})$ such that $\boldsymbol{l}_{n+1} = \boldsymbol{l}^*$, and $\boldsymbol{u}(\boldsymbol{A}_{n+1}, p_n) = \boldsymbol{u}^*$. ■

For each $\boldsymbol{l}$, the function $G(\boldsymbol{u}, \boldsymbol{l})$ is strictly concave in $\boldsymbol{u}$. Thus, we can find the maximum value and maximizing argument for each $\boldsymbol{l}$, and then select the maximum value among the possible choices of $\boldsymbol{l}$ to obtain $(\boldsymbol{u}^*, \boldsymbol{l}^*)$. The maximizing $\boldsymbol{l}^*$ may not be unique, as there can be multiple precision modes with the same maximum value. If the agent error models are stage-invariant, there is an optimal strategy where each agent will choose the same precision mode at every sensing stage.

We now show that optimal strategies can be computed from the solution of single agent problems: Let

$$\begin{aligned} G^m(u, l) &= \mathcal{H}(u f_1^{m,l}(y) + (1-u) f_0^{m,l}(y)) - \\ &\quad u\mathcal{H}(f_1^{m,l}(y)) - (1-u)\mathcal{H}(f_0^{m,l}(y)) - \gamma W^m(l) \end{aligned} \tag{6.16}$$

and let $(u^{m*}, l^{m*}) = \arg\max_{u,l} G^m(u,l)$ denote an optimal operating point and choice of precision mode for agent m when it searches by itself.

Theorem 6.3.3. *Let $(u^{m*}, l^{m*}) = \arg\max_{u,l} G^m(u,l)$ denote an optimal operating point and choice of precision mode for each agent $m = 1, \ldots, M$ when searching as a single agent. Then, an optimal joint operating point and precision modes for joint sensing $(\boldsymbol{u}^*, \boldsymbol{l}^*)$ can be obtained as $\boldsymbol{l}^* = (l^{1*}, \ldots, l^{M*})$, and*

$$u^*_{i_{M:1}} = \prod_{m=1}^{M} (u^{m*})^{i_m} (1 - u^{m*})^{1-i_m} \tag{6.17}$$

In addition, $G^* = \sum_{m=1}^{M} G^{m*}$, where $G^{m*} = G^m(u^{m*}, l^{m*})$.

Proof. We first prove that $G^* \leq \sum_{m=1}^{M} G^{m*}$:

$$\begin{aligned}
&G^* + \gamma \sum_{m=1}^{M} W^m(l^{m*}) = \mathcal{H}\Big(\sum_{i_{M:1} \in \{0,1\}^M} u^*_{i_{M:1}} q^{\boldsymbol{l}^*}_{i_{M:1}} \Big) \\
&- \sum_{i_{M:1} \in \{0,1\}^M} u^*_{i_{M:1}} \mathcal{H}(q^{\boldsymbol{l}^*}_{i_{M:1}}) \\
\leq &\sum_{m=1}^{M} [\mathcal{H}\Big(\sum_{i_{M:1}: i_m = 1} u^*_{i_{M:1}} f_1^{m,l^{m*}} + \sum_{i_{M:1}: i_m = 0} u^*_{i_{M:1}} f_0^{m,l^{m*}} \Big) \\
&- \sum_{i_{M:1}: i_m = 1} u^*_{i_{M:1}} \mathcal{H}(f_1^{m,l^{m*}}) - \sum_{i_{M:1}: i_m = 0} u^*_{i_{M:1}} \mathcal{H}(f_0^{m,l^{m*}})]
\end{aligned}$$

The inequality results from the subadditivity and additivity properties of the Shannon entropy. Then, since (u^{m*}, l^{m*}), $m = 1, \ldots, M$, are the optimal points of $G^m(u^m, l^m)$, $m = 1, \ldots, M$,

$$\begin{aligned}
&G^* + \gamma \sum_{m=1}^{M} W^m(l^{m*}) \\
\leq &\sum_{m=1}^{M} \Big[\mathcal{H}\Big(u^{m*} f_1^{m,l^{m*}} + (1 - u^{m*}) f_0^{m,l^{m*}} \Big) \\
&- \Big(u^{m*} \mathcal{H}(f_1^{m,l^{m*}}) + (1 - u^{m*}) \mathcal{H}(f_0^{m,l^{m*}}) \Big) \Big]
\end{aligned}$$

$$= \sum_{m=1}^{M} G^{m*} + \gamma \sum_{m=1}^{M} W^m(l^{m*})$$

Let $\bar{\boldsymbol{l}} = (l^{1*}, \ldots, l^{M*})$, and $\overline{u}_{i_{M:1}} = \prod_{m=1}^{M} (u^{m*})^{i_m} (1 - u^{m*})^{1-i_m}$. Substituting into $G(\boldsymbol{u}, \boldsymbol{l})$, and following an argument similar to the proof of Theorem 6.2.1, we have

$$\begin{aligned}
& G(\overline{\boldsymbol{u}}, \bar{\boldsymbol{l}}) + \gamma \sum_{m=1}^{M} W^m(l^{m*}) \\
&= \mathcal{H}\Big(\sum_{i_{M:1} \in \{0,1\}^M} \overline{u}_{i_{M:1}} q^{\bar{\boldsymbol{l}}}_{i_{M:1}} \Big) - \sum_{i_{M:1} \in \{0,1\}^M} \overline{u}_{i_{M:1}} \mathcal{H}(q^{\bar{\boldsymbol{l}}}_{i_{M:1}}) \\
&= \mathcal{H}\Big(\prod_{m=1}^{M} [\sum_{j=0}^{1} f_j^{m,l^{m*}} (u^{m*})^j (1 - u^{m*})^{1-j}] \Big) \\
&\quad - \sum_{m=1}^{M} \sum_{j=0}^{1} (u^{m*})^j (1 - u^{m*})^{1-j} \mathcal{H}(f_j^{m,l^{m*}}) \\
&= \sum_{m=1}^{M} \Big[\mathcal{H}\Big(\sum_{j=0}^{1} f_j^{m,l^{m*}} (u^{m*})^j (1 - u^{m*})^{1-j} \Big) \\
&\quad - \sum_{j=0}^{1} (u^{m*})^j (1 - u^{m*})^{1-j} \mathcal{H}(f_j^{m,l^{m*}}) \Big] \\
&= \sum_{m=1}^{M} G^{m*} + \gamma \sum_{m=1}^{M} W^m(l^{m*})
\end{aligned}$$

Since $G^* = \max_{\boldsymbol{u},\boldsymbol{l}} G(\boldsymbol{u}, \boldsymbol{l}) \leq \sum_{m=1}^{M} G^{m*}$, selecting $(\boldsymbol{u}, \boldsymbol{l})$ as (6.17) will give us an optimal operating point for $G(\boldsymbol{u}, \boldsymbol{l})$ and we have $G^* = \sum_{m=1}^{M} G^{m*}$. ■

Note the resulting computation complexity for the algorithm. For each agent m, it must solve L^m scalar concave maximization problems to obtain the maximum in (6.16). Given the operating points of all the agents, the joint operating point $u^*_{i_{M:1}}$ is computed as in (6.17). Using this operating point, and the statistic $p_{n-1}(x)$, 2^M regions B^k are constructed such that:

$$u^*_{i_{M:1}} = \int_{B^k} p_{n-1}(x) dx$$

Finally, the observation regions for each agent A_n^m are constructed as

$$A_n^m = \cup_{k:i_m=1} B^k$$

where $i_M i_{M-1} \cdots i_1$ is the dyadic expansion of k.

The results for joint precision mode and sensing area optimization can be readily extended to the correlated noise case in Section 6.1. However, the complexity of the resulting optimization algorithm increases greatly. In essence, one must solve $L^1 \times L^2 \cdots \times L^M$ strictly convex maximization problems in 2^M variables to determine the best combination of joint operating modes and precision mode selections. This makes the results difficult to apply in situations where the parameters of the sensing problem are determined in real-time, requiring that these optimizations be part of real-time computations.

6.4 Search of Multiple Objects

In this section, we extend our results to the search of multiple objects. Consider searching for J stationary objects $X_1, \ldots, X_J$ in a compact subset $\mathcal{X}$ of $\mathbb{R}^d$ using M search agents. We write $X_1, \ldots, X_J$ in a compact way as $X_{J:1}$. $X_{J:1}$ takes values in the *expanded search space* $\Omega = \mathcal{X}^J$. Assume that the number of objects J is known, and that we have prior knowledge of $X_{J:1}$, represented by a joint density $p_0(x_{J:1})$ over Ω, which is absolutely continuous with respect to Lebesgue measure.

At each stage, the m-th agent can select a Lebesgue measurable subset of $\mathcal{X}$, denoted by A_n^m, to search. Define an indicator variable $Z_n^m = \mathbb{1}_{\{\exists j: X_j \in A_n^m\}}$ which indicates if any object lies in the sensing region A_n^m. Assume that the measurement Y_n^m takes values from a discrete set $\mathcal{Y}$ and is subject to the following error model:

$$Pr(Y_n^m = y|X_{J:1}, A_n^m) = \begin{cases} f_1^m(y), & \text{if } Z_n^m = 1 \\ f_0^m(y), & \text{if } Z_n^m = 0 \end{cases}$$

Thus, the measurement error of Y_n^m depends on the true object locations $X_{J:1}$ and the sensing region A_n^m only through the indicator Z_n^m. This measurement error model is similar to the one used in group testing [Du and Hwang, 1993], but different from the one used in [Rajan et al., 2015], where the measurement errors depend on the number of objects in the sensing region A_n^m.

Assume that the measurements of all agents are conditionally independent across agents and across stages given the true object locations $X_{J:1}$ and the sensing decisions $\boldsymbol{A}_n = (A_n^1, \ldots, A_n^M)$. Given $Z_n^1 = i_1, \ldots, Z_n^M = i_M$ associated with the sensing regions $A_n^1, \ldots, A_n^M$ and the true object locations $X_{J:1}$, the conditional density of the joint measurements is:

$$Pr(\boldsymbol{Y}_n = \boldsymbol{y}|X_{J:1}, \boldsymbol{A}_n) = \prod_{m=1}^{M} f_{i_m}^m(y^m) \equiv q_{i_{M:1}}(\boldsymbol{y})$$

where we use $i_{M:1}$ as a shorthand for $(i_M, \ldots, i_1)$.

Define the information history D_n as the collection of the past sensing decisions and measurements up to time n:

$$D_n = \{\boldsymbol{A}_1, \boldsymbol{Y}_1, \ldots, \boldsymbol{A}_n, \boldsymbol{Y}_n\}$$

and define the information state of our search problem as the posterior density of $X_{J:1}$ in the expanded search space Ω given the history D_n, denoted by $p_n(x_{J:1})$:

$$p_n(x_{J:1}) = p(x_{J:1}|D_n)$$

This information state evolves according to Bayes' rule when the observed values for $\boldsymbol{Y}_{n+1} = \boldsymbol{y}$, as follows:

$$p_{n+1}(x_{J:1}) = p_n(x_{J:1}) \frac{\eta_0(\boldsymbol{y}, x_{J:1})}{\eta(\boldsymbol{y})} \tag{6.18}$$

where

$$\begin{aligned}\eta_0(\boldsymbol{y}, x_{J:1}) &\equiv P(\boldsymbol{Y}_{n+1} = \boldsymbol{y} | X_{J:1} = x_{J:1}, \boldsymbol{A}_{n+1}) \\ &= \sum_{i_{M:1} \in \{0,1\}^M} q_{i_{M:1}}(\boldsymbol{y}) \mathbb{1}_{\{\exists x_j : x_j \in \cap_{m=1}^M (A_n^m)^{i_m}\}}\end{aligned}$$

and

$$\begin{aligned}\eta(\boldsymbol{y}) \equiv & P(\boldsymbol{Y}_{n+1} = \boldsymbol{y} | \boldsymbol{A}_{n+1}, D_n) \\ = & \int_\Omega p_n(\omega_{J:1}) \sum_{i_{M:1} \in \{0,1\}^M} q_{i_{M:1}}(\boldsymbol{y}) \mathbb{1}_{\{\exists \omega_j : \omega_j \in \cap_{m=1}^M (A_n^m)^{i_m}\}} d\omega_{J:1}\end{aligned}$$

Define the joint operating point $\boldsymbol{u} = \{u_{i_{M:1}}, i_1, \ldots, i_M \in \{0,1\}\}$ as

$$u_{i_{M:1}} = \int_\Omega p_n(\omega_{J:1}) \mathbb{1}_{\{\exists \omega_j : \omega_j \in \cap_{m=1}^M (A_n^m)^{i_m}\}} d\omega_{J:1}$$

Then, $\eta(\boldsymbol{y}) = \sum_{i_1, \ldots, i_M = 0}^1 q_{i_{M:1}}(\boldsymbol{y}) u_{i_{M:1}}$.

Define a joint sensing policy $\pi = (\pi_1, \pi_2, \cdots, \pi_N)$ to be a sequence of functions which maps the information state p_{n-1} to admissible sensing regions $\boldsymbol{A}_n$ at each stage n.

Define the *posterior differential entropy* $H(p_n)$ as

$$H(p_n) = -\int_\Omega p_n(x_{J:1}) \log_2 p_n(x_{J:1}) dx_{J:1}$$

As before, our objective is to find a policy π to minimize the expected value of $H(p_N)$ after N search stages:

$$\inf_{\pi \in \Pi} E[H(p_N) | p_0]$$

To solve the problem in a stochastic control framework, define the optimal value function $V(p_n, n)$ at stage n to be

$$V(p_n, n) = \inf_{\pi_{n+1},\ldots,\pi_N} E[H(p_N)|p_n] \tag{6.19}$$

A policy π is optimal if it satisfies the Bellman equation at each stage:

$$V(p_n, n) = \inf_{\boldsymbol{A}_{n+1}=\pi_n(p_n)} E_{\boldsymbol{Y}_{n+1}}[V(p_{n+1}, n+1)|\boldsymbol{A}_{n+1}, p_n]$$

We first show that the expected one-stage reduction in posterior differential entropy is a function of the joint operating point $\boldsymbol{u}$.

Lemma 6.4.1. *Given the search decision at time $n+1$ to be $\boldsymbol{A}_{n+1} = (A_n^1, \ldots, A_n^M)$, the expected one-stage reduction in posterior differential entropy is*

$$\begin{aligned}\varphi(\boldsymbol{u}) \equiv& H(p_n) - E[H(p_{n+1})|\boldsymbol{A}_{n+1} = (A_n^1, \ldots, A_n^M), p_n] \\ =& \mathcal{H}\Big(\sum_{i_{M:1}\in\{0,1\}^M} q_{i_{M:1}} u_{i_{M:1}}\Big) - \sum_{i_{M:1}\in\{0,1\}^M} u_{i_{M:1}} \mathcal{H}(q_{i_{M:1}})\end{aligned}$$

where $\mathcal{H}$ is the standard Shannon entropy for discrete-valued distributions.

Proof.

$$\begin{aligned}&E_{Y_{n+1}}[H(p_{n+1})|\boldsymbol{A}_{n+1}, p_n] \\ =& \sum_{\boldsymbol{y}\in\mathcal{Y}^M} H(p_{n+1}) P(\boldsymbol{Y}_{n+1} = \boldsymbol{y}|\boldsymbol{A}_{n+1}, p_n) \\ =& H(p_n) + \sum_{\boldsymbol{y}\in\mathcal{Y}^M} \int_\Omega p_n(x_{J:1}) \eta_0(\boldsymbol{y}, x_{J:1}) \log \eta(\boldsymbol{y}) dx_{J:1} \\ &- \int_\Omega p_n(x_{J:1}) \sum_{\boldsymbol{y}\in\mathcal{Y}^M} \eta_0(\boldsymbol{y}, x_{J:1}) \log \eta_0(\boldsymbol{y}, x_{J:1}) dx_{J:1} \\ =& H(p_n) - \Big[\mathcal{H}(\eta(\boldsymbol{y})) - \int_\Omega p_n(x_{J:1}) \mathcal{H}(\eta_0(\boldsymbol{y}, x_{J:1})) dx_{J:1}\Big] \\ =& H(p_n) - \varphi(\boldsymbol{u})\end{aligned}$$

■

Note that $\varphi(\boldsymbol{u})$ depends on the information state $p_n(x_{J:1})$ and the search deci-

sion $\boldsymbol{A}_{n+1}$ only through the joint operating point $\boldsymbol{u}$. Following the strict concavity of the Shannon entropy $\mathcal{H}(\cdot)$, $\varphi(\boldsymbol{u})$ is a strictly concave function over the simplex $\sum_{i_{M:1}\in\{0,1\}^M} i_{i_{M:1}} = 1$ and thus it has a unique maximum achieved at the optimal joint operating point $\boldsymbol{u}$.

We now provide a sufficient condition for optimal search policies.

Theorem 6.4.2. *Let $\boldsymbol{u}^* \in \arg\min_{\boldsymbol{u}} \varphi(\boldsymbol{u})$ be the optimal joint operating point for $\varphi(\boldsymbol{u})$ as defined in Lemma 6.4.1. If for the given $p_{n-1}(x_{J:1})$ at each stage $n = 1, \ldots, N$, there exists decision $\boldsymbol{A}_n = (A_n^1, \ldots, A_n^M)$ such that $\boldsymbol{u}(\boldsymbol{A}_n, p_{n-1}) = \boldsymbol{u}^*$, then selecting such search decisions $\boldsymbol{A}_n$ adaptively from stage 1 to N constitutes an optimal policy.*

Moreover, the optimal value function in (6.19) *has a closed-form expression:*

$$V(p_n, n) = H(p_n) - (N - n)\varphi^*$$

where the constant $\varphi^* = \varphi(\boldsymbol{u}^*)$.

Proof. We prove the result by induction. At time N, the value function is $V(p_N, N) = H(p_N)$. Assume that for $k \geq n + 1$, the value function has the proposed form. Then by Lemma 6.4.1,

$$\begin{aligned} V(p_n, n) &= \inf_{\boldsymbol{A}_{n+1}} E[V(p_{n+1}, n+1)|\boldsymbol{A}_{n+1}, p_n] \\ &= \inf_{\boldsymbol{A}_{n+1}} E[H(p_{n+1}) - (N-n-1)\varphi^*|\boldsymbol{A}_{n+1}, p_n] \\ &= H(p_n) - \sup_{\boldsymbol{A}_{n+1}} \varphi(\boldsymbol{u}(\boldsymbol{A}_{n+1}, p_n)) - (N-n-1)\varphi^* \\ &= H(p_n) - (N-n)\varphi^* \end{aligned}$$

Hence, from Lemma 6.4.1 we can see that if for the given posterior density p_n, there exists $\boldsymbol{A}_{n+1}$ that satisfies $\boldsymbol{u}(\boldsymbol{A}_{n+1}, p_n) = \boldsymbol{u}^*$, then selecting such $\boldsymbol{A}_{n+1}$ will satisfy the Bellman equation and the resulting policy is optimal. ■

We now show how to construct the decision $\boldsymbol{A}_{n+1}$ for two objects x_1 and x_2 in $\mathcal{X} = [0, 1]$ using two agents as in Figure 6·3. The first agent searches $A_{n+1}^1 = [a_1, b_1] \subset \mathcal{X}$ and the second agent searches $A_{n+1}^2 = [a_2, b_2] \subset \mathcal{X}$. The expanded search space is $\Omega = [0, 1]^2$.

For ease of exposition, define the joint operating point $\boldsymbol{u} = (u_{11}, u_{10}, u_{01}, u_{00})$ as

$$u_{11} \equiv \int_{\Omega} \mathbb{1}_{\{x_1 \in A^1_{n+1} \text{ or } x_2 \in A^1_{n+1}\}} \mathbb{1}_{\{x_1 \in A^2_{n+1} \text{ or } x_2 \in A^2_{n+1}\}} p_n(x_1, x_2) dx_1 dx_2$$

$$u_{10} \equiv \int_{\Omega} \mathbb{1}_{\{x_1 \in A^1_{n+1} \text{ or } x_2 \in A^1_{n+1}\}} \mathbb{1}_{\{x_1 \notin A^2_{n+1} \text{ and } x_2 \notin A^2_{n+1}\}} p_n(x_1, x_2) dx_1 dx_2$$

$$u_{01} \equiv \int_{\Omega} \mathbb{1}_{\{x_1 \notin A^1_{n+1} \text{ and } x_2 \notin A^1_{n+1}\}} \mathbb{1}_{\{x_1 \in A^2_{n+1} \text{ or } x_2 \in A^2_{n+1}\}} p_n(x_1, x_2) dx_1 dx_2$$

$$u_{00} \equiv \int_{\Omega} \mathbb{1}_{\{x_1 \notin A^1_{n+1} \text{ and } x_2 \notin A^1_{n+1}\}} \mathbb{1}_{\{x_1 \notin A^2_{n+1} \text{ and } x_2 \notin A^2_{n+1}\}} p_n(x_1, x_2) dx_1 dx_2$$

Let $a_1 = 0$. Next, select b_2 such that

$$\int_{b_2}^{1} \int_{b_2}^{1} p_n(x_1, x_2) dx_1 dx_2 = u_{00}$$

This integral corresponds to the area of the top right square in orange. Then select b_1 such that

$$\int_{b_1}^{1} \int_{b_1}^{1} p_n(x_1, x_2) dx_1 dx_2 = u_{00} + u_{01}$$

This corresponds to both the orange and the blue areas. The above steps are realizable since the related integrals are continuous monotone functions.

Finally, given b_2 already fixed, we can select a_2 such that

$$\int_{0}^{a_2} \int_{0}^{a_2} p_n(x_1, x_2) dx_1 dx_2 + \int_{b_2}^{1} \int_{0}^{a_2} p_n(x_1, x_2) dx_1 dx_2 + \int_{0}^{a_2} \int_{b_2}^{1} p_n(x_1, x_2) dx_1 dx_2 = u_{10}$$

It corresponds to the yellow area. This can be done if

$$\int_{0}^{b_1} \int_{0}^{b_1} p_n(x_1, x_2) dx_1 dx_2 + \int_{b_2}^{1} \int_{0}^{b_1} p_n(x_1, x_2) dx_1 dx_2 + \int_{0}^{b_1} \int_{b_2}^{1} p_n(x_1, x_2) dx_1 dx_2 \geq u_{10}$$

The resulting partition is shown in Figure 6·3.

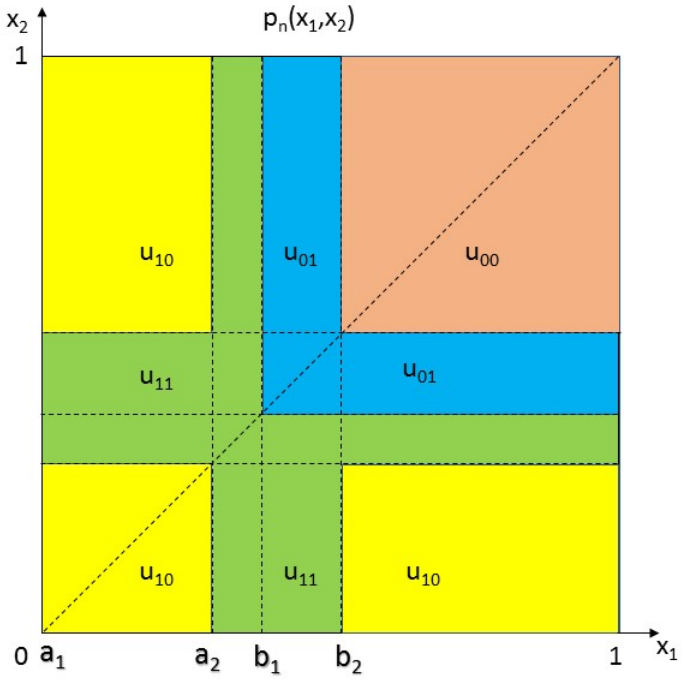


Figure 6·3: One way of realizing $(u_{11}, u_{10}, u_{01}, u_{00})$ given $p_n(x_1, x_2)$. $A^1_{n+1} = [a_1, b_1]$ and $A^2_{n+1} = [a_2, b_2]$ partition the domain of $p_n(x_1, x_2)$ into four disjoint regions, shown with different colors.

Note that the above construction results in overlapping intervals of observation for the two agents, where the amount of overlap is controlled adaptively based on the posterior probability density $p_n(x_1, x_2)$. The construction is suitable for two agents. When there are more than two agents in use, constructing a search decision for which the joint operating point $\boldsymbol{u} = (u_{11}, u_{10}, u_{01}, u_{00})$ achieves its optimal value is more complex. Furthermore, it is impossible to have search decisions consisting of single intervals in most cases.

6.5 Experiments

6.5.1 Experiments for Single Object

In this section, we illustrate the performance of our algorithms using simulated scenarios. We consider scenarios where the object location is in the unit square $\mathcal{X} = [0, 1]^2$, with initial density uniformly distributed in the area. The scenarios include two agents, denoted as 1 and 2, that will cooperate and share their sensed measurements of the object position X. We assume that the agents only have one precision mode. Denote the areas sensed at time n by each of the agents as A_n^1, A_n^2.

In terms of sensing errors, we first consider a correlated error model. We assume that each agent returns a binary measurement $y \in \{0, 1\}$, which is a noisy version of the indicator function that the object X is in the interrogated region. This binary measurement corresponds to a detector algorithm, where each agent detects the presence of a signal among background noise in the area of observation. The signal strength is random, and contributes to the correlated errors among the agents; the performance statistics of the agent were computed using the probabilities of error from an unknown-signal-in-noise model, as in [Van Trees, 2004]. The resulting correlated error model is given by Table 6.1 a). The second case uses agents with independent error models, as the signal strength is modeled as deterministic, but in an unknown location. We specify the error distribution in terms of the marginal distributions $f_k^m, m = 1, 2, k = 0, 1$ in Table 6.1 b). Note that, in the correlated error model, the agents have greater probability of agreeing on a measurement, which corresponds to a significant part of the error being created by randomness in the signature of the object of interest.

For these problems, computation of a greedy solution to minimize expected mean squared error at each stage is a formidable task, as it requires searching over all possible combinations of sensing regions for each agent. With the posterior differential

Table 6.1: The agent specifications for two types of Boolean agents: correlated and independent errors

	$\boldsymbol{y}$ =(0,0)	$\boldsymbol{y}$ =(0,1)	$\boldsymbol{y}$ =(1,0)	$\boldsymbol{y}$ =(1,1)
(0,0)	0.62	0.17	0.17	0.04
(0,1)	0.21	0.57	0.06	0.16
(1,0)	0.11	0.03	0.68	0.18
(1,1)	0.11	0.02	0.16	0.71

a) Correlated

	$y=0$	$y=1$
f_0^1	0.79	0.21
f_1^1	0.14	0.86
f_0^2	0.79	0.21
f_1^2	0.27	0.73

b) Independent

entropy objective, we have optimal strategies characterized in terms of computed operating points. For the independent case, the optimal operating points computed using the MATLAB function **fmincon** are $u^{1*} = 0.511, u^{2*} = 0.494$. Agent 1 is more accurate, and thus seeks to include more probability in its search area. The expected one-stage reduction in differential entropy for this case is 0.54 bits. For the correlated case, the joint operating point is $u^{11*} = 0.288, u^{10*} = 0.25, u^{01*} = 0.25, u^{00*} = 0.212$, with expected differential entropy reduction of 0.58 bits per stage. The correlated channel case leads to greater reduction in differential entropy, as the agents exploit the correlation in the signal to enhance information extraction. When compared with the optimal strategies for independent agents, the correlated agents increase the probability of the overlap area (u^{11*} vs. $u^{1*} \times u^{2*}$) where both agents query as to the presence of the object.

The optimal strategies at each stage n, based on $p_{n-1}(x)$, are to find regions $A_n^1, A_n^2 \subset [0,1]^2$ so that

$$\int_{x \in A_n^1 \cap A_n^2} p_{n-1}(x)dx = u^{11*}$$

$$\int_{x \in (A_n^1)^c \cap A_n^2} p_{n-1}(x)dx = u^{01*}$$

$$\int_{x \in A_n^1 \cap (A_n^2)^c} p_{n-1}(x)dx = u^{10*}$$

$$\int_{x \in (A_n^1)^c \cap (A_n^2)^c} p_{n-1}(x)dx = u^{00*}$$

Note that there are many sensing strategies that will satisfy the above equalities. We exploit this degree of freedom to select sensing strategies that can be implemented by agents with field of view constraints, and that aim to reduce mean squared error as well as achieving optimal reduction in differential entropy. Thus, we choose our subsets to be rectangular intervals, so that agents will observe connected regions. In addition, we choose our sensing strategies to alternate between partitions of the x-axis at odd times n, and partitions of the y-axis at even times n, dividing each axis into intervals with probabilities corresponding to $u^{10*}, u^{11*}, u^{01*}, u^{00*}$, and then we aggregate the appropriate regions to compute the sensing areas A_n^1, A_n^2. This construction is illustrated in Figure 6·4. By alternating between axes for partition at different times, we ensure that the errors in both dimensions are reduced as the differential entropy decreases.

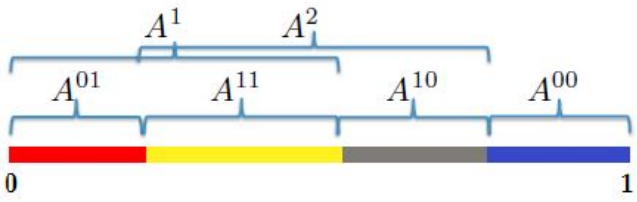


Figure 6·4: Partition of a line segment into four disjoint subsets at each stage.

For each of the agent measurement models, we conducted 2000 Monte Carlo experiments. In each experiment, we randomly generate a object position $X \in \mathcal{X} = [0,1]^2$ using a uniform distribution. We initialize our prior density for X, $p_0(x)$, as a uniform distribution; therefore, the initial differential entropy $H(p_0) = -\int_0^1 \int_0^1 \log_2 1 dx\, dy = 0$. At each stage $n > 0$, given the density $p_{n-1}(x)$, sensing areas A_n^1, A_n^2 are selected, and random measurements (y_n^1, y_n^2) are generated according to the agent error mod-

els. These measurements are used to update the conditional density from $p_{n-1}(x)$ to $p_n(x)$ as indicated in (6.2). We continue this process until $n = 20$ sensing stages are completed.

For each experiment, we plot the average differential entropy $H(p_n)$ and the average mean-square error as a function of n. Figure 6·5(a) contains the average differential entropy results for both the correlated and independent measurement error models. As expected, the average differential entropy for the correlated case decays faster than that for the independent case as the number of stages increases. Figure 6·5(b) contains the graph of the mean squared error of the estimated object location as a function of the number of stages, as well as the lower bounds on the errors. We note the near-equivalence of the mean squared error in both cases, leading to an exponential decay as a function of the number of stages. This suggests that an exponentially decaying upper bound may be possible for these algorithms, although no such bound has been established in the literature.

The second set of experiments consists of 3 agents searching for an object in $\mathcal{X} = [0, 1]$. Each agent has observations taking 3 possible values. The sensing areas at stage n are denoted as A_n^1, A_n^2 and A_n^3 respectively. We assume that agent 3 has two choices of precision modes. The agent error models are shown in Table 6.2. For agent 3, the cost of mode 1 is $W^3(1) = 0$, whereas the cost of mode 2 is $W^3(2) = 0.07$, with the relative weight parameter γ set to 1.

Table 6.2: The error models for three Boolean agents

	$y = 0$	$y = 1$	$y = 2$
$f_0^1(y)$	0.3	0.5	0.2
$f_1^1(y)$	0.2	0.5	0.3
$f_0^2(y)$	0.7	0.2	0.1
$f_1^2(y)$	0.2	0.7	0.1
$f_0^{3,1}(y)$	0.3	0.1	0.6
$f_1^{3,1}(y)$	0.3	0.6	0.1
$f_0^{3,2}(y)$	0.2	0.1	0.7
$f_1^{3,2}(y)$	0.2	0.7	0.1

Note that the error model of each agent for different precision modes satisfies the symmetry conditions; thus the individual optimal operating points for each agent are $u^{1*} = u^{2*} = u^{3*} = 0.5$, for each of the two precision modes of agent 3. By the results of Theorem 6.3.3, the optimal precision mode for agent 3 must satisfy $l^{3*} = \arg\max_{l=1,2} G^3(0.5, l)$. Evaluating for precision mode 1, we have the net information gain (6.15) is 0.286 and cost is zero, whereas the net information gain under precision mode 2 is 0.365 with a cost of 0.07. Thus $l^{3*} = 2$ is the optimal precision mode.

To find regions A_n^1, A_n^2 and A_n^3, we first select $\{B_n^{i_3 i_2 i_1} : i_1, i_2, i_3 \in \{0,1\}\}$ as in Figure 6·6 such that

$$\int_{B_n^{i_3 i_2 i_1}} p_{n-1}(x)dx = \frac{1}{8}, \quad i_1, i_2, i_3 \in \{0,1\}$$

We do so by partitioning the unit interval into regions with equal probability mass, as illustrated in Figure 6·6. The resulting sensing regions are $A_n^1 = B_n^{001} \cup B_n^{011} \cup B_n^{101} \cup B_n^{111}$; $A_n^2 = B_n^{010} \cup B_n^{011} \cup B_n^{110} \cup B_n^{111}$; $A_n^3 = B_n^{100} \cup B_n^{101} \cup B_n^{110} \cup B_n^{111}$.

We conduct 5000 Monte Carlo experiments under the optimal joint sensing policy for three Boolean agents, randomly selecting the initial position $X \in \mathcal{X} = [0,1]$ uniformly. At each stage $n > 0$, we select the sensing areas A_n^1, A_n^2 and A_n^3 based on the current information state $p_{n-1}(x)$ and the optimal joint operating point $\boldsymbol{u}^*$ as above. The noisy measurements (y_n^1, y_n^2, y_n^3) are randomly generated according to the agent error models, and will be used to update the conditional density from $p_{n-1}(x)$ to $p_n(x)$. The process continues until $n = 20$ sensing stages are complete. The net reduction in differential entropy per time using agent 1, agent 2 and the second mode of agent 3 is 0.592 bits, so the differential entropy decays linearly at that slope. The empirical mean squared error and the lower bound at each stage are shown in Figure 6·7. The mean squared error shows the exponential decay we expect.

6.5.2 Experiments for Multiple Objects

In this section, we illustrate the performance of our search strategies using two agents to search for two objects in $\mathcal{X} = [0, 1]$. Table 6.3 contains the details of the agent measurement error models. The measurements Y are binary. The second agent has less noisy measurements.

Table 6.3: The specifications of two agents. For such agent measurement error models, $u^*_{11} = u^*_{10} = u^*_{01} = u^*_{00} = \frac{1}{4}$.

	$y = 0$	$y = 1$
$f^1_0(y)$	0.7	0.3
$f^1_1(y)$	0.3	0.7
$f^2_0(y)$	0.8	0.2
$f^2_1(y)$	0.2	0.8

The optimal joint operating point is $\boldsymbol{u}^* = (1/4, 1/4, 1/4, 1/4)$. At each time step n, given $p_{n-1}(x_1, x_2)$, we select intervals $A^1_n = [a_1, b_1]$ and $A^2_n = [a_2, b_2]$. Since we can always pick $a_1 = 0$, we only have to determine three values at each stage, namely b_2, b_1, and a_2. They must satisfy three equalities

$$\begin{aligned}
&\int_{[b_2,1]\times[b_2,1]} p_{n-1}(x_1, x_2)dx_1dx_2 = \frac{1}{4} \\
&\int_{[b_1,1]\times[b_1,1]} p_{n-1}(x_1, x_2)dx_1dx_2 = \frac{1}{4} \\
&\int_{[0,a_2]\times[0,a_2]} p_{n-1}(x_1, x_2)dx_1dx_2 \\
+ &\int_{[b_2,1]\times[0,a_2]} p_{n-1}(x_1, x_2)dx_1dx_2 \\
+ &\int_{[0,a_2]\times[b_2,1]} p_{n-1}(x_1, x_2)dx_1dx_2 = \frac{1}{4}
\end{aligned}$$

The construction is illustrated in Figure 6·3.

We compare the optimal policy with a heuristic policy based on spatial partitions.

In the heuristic policy, the first agent is restricted to searching intervals $\tilde{A}_n^1 = [0, \tilde{b}_1]$ that start at the beginning of $\mathcal{X} = [0, 1]$, and the second agent is restricted to intervals $\tilde{A}_n^2 = [\tilde{a}_2, 1]$ that end at the high end of $\mathcal{X} = [0, 1]$. The values of $\tilde{b}_1$ and $\tilde{a}_2$ need to satisfy

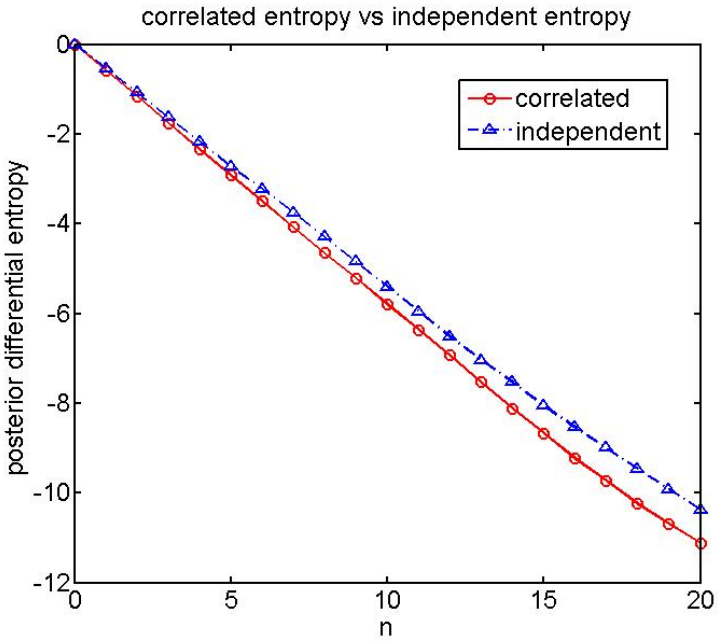
$$\int_{\tilde{b}_1}^{1}\int_{\tilde{b}_1}^{1} p_{n-1}(x_1, x_2)dx_1dx_2 = \frac{1}{2}$$
$$\int_{0}^{\tilde{a}_2}\int_{0}^{\tilde{a}_2} p_{n-1}(x_1, x_2)dx_1dx_2 = \frac{1}{2}$$

which are the optimal decisions for single-agent search. Thus, there is no coordination of interval selections among the two agents; instead, the coordination is that each agent will focus on different spatial regimes.

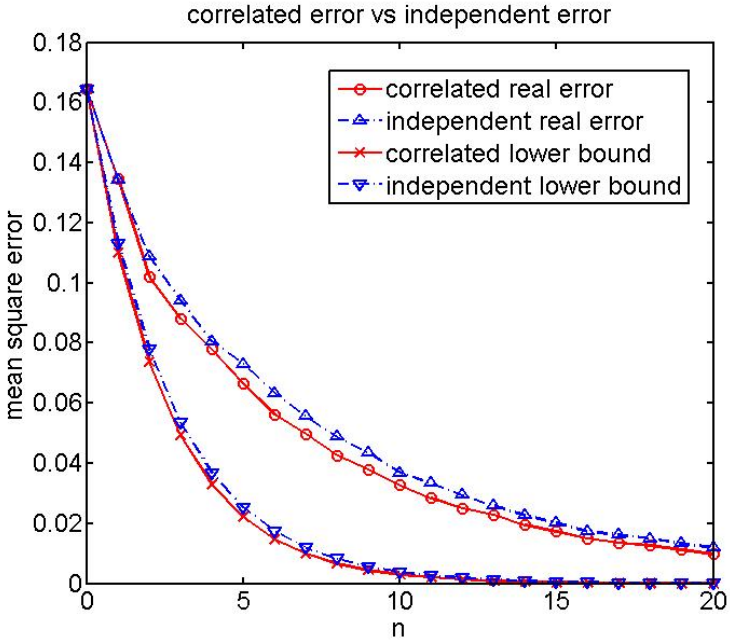
For both the optimal policy and the heuristic policy, we conduct 100 Monte Carlo experiments. In each experiment, we independently generate two object positions $X_1,\ X_2 \in \mathcal{X} = [0, 1]$ from a uniform distribution. We initialize the prior density of (X_1, X_2), i.e. $p_0(x_1, x_2)$, to be uniform over $\Omega = [0, 1]^2$; therefore, the initial differential entropy is $H(p_0) = -\int_0^1 \int_0^1 \log_2(1)dx_1dx_2 = 0$. At each time step n, given $p_{n-1}(x_1, x_2)$, we select $\boldsymbol{A}_n = (A_n^1, A_n^2)$ based on the optimal policy and $\tilde{\boldsymbol{A}}_n = (\tilde{A}_n^1, \tilde{A}_n^2)$ based on the heuristic policy, and generate the corresponding measurements (y_n^1, y_n^1) and $(\tilde{y}_n^1, \tilde{y}_n^2)$ respectively. Then we update the conditional density from p_{n-1} to p_n and compute the posterior differential entropy $H(p_n)$. The process continues until $n = 20$ sensing stages are complete.

We compute the differential entropy of the conditional probability density of the object locations at each stage of the 100 sample experiments, for using both the optimal policy and the heuristic policy. The average entropy paths are plotted in Figure 6·8. The trajectory for the optimal policy is shown in black solid line with circle points, while the trajectory for the heuristic is shown in blue dash-dotted line

with diamond points. We also include the theoretical benchmark, computed from Theorem 6.4.2, shown in the black straight dashed line. It is clear that the average differential entropy path under the optimal policy decays linearly as predicted, close to the theoretical benchmark, and faster than that under the heuristic policy.



(a) Differential entropy of optimal sensing strategies.



(b) Mean squared error of optimal sensing strategies.

Figure 6·5: Results with correlated and independent error models

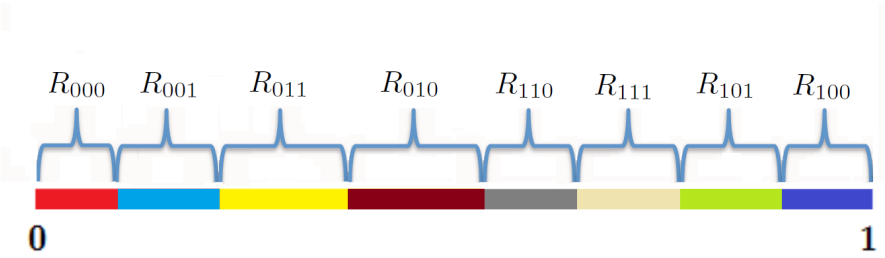


Figure 6·6: Partition of a line segment into $2^3 = 8$ disjoint subsets at each stage.

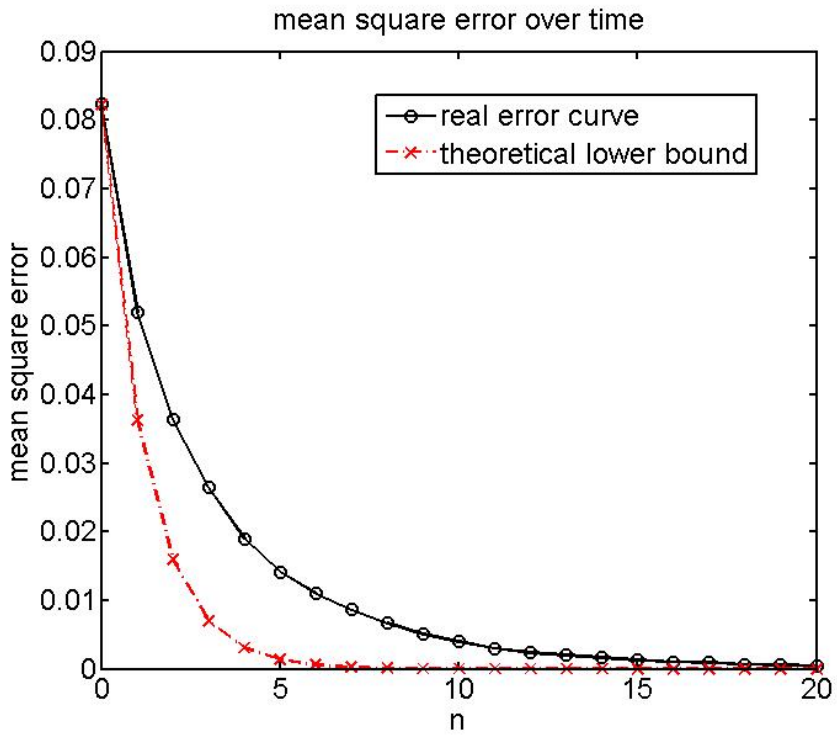


Figure 6·7: The average mean squared error for three-agent joint search under the optimal joint sensing policy, as a function of time.

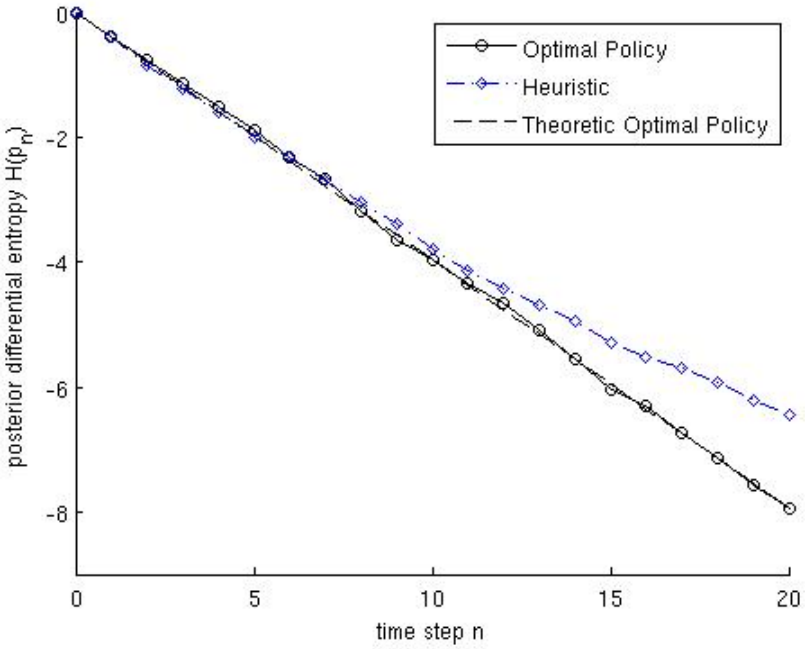


Figure 6·8: The posterior entropy reduction paths for the optimal policy, the heuristic and the theoretic optimal benchmark.

Chapter 7

Conclusions

7.1 Summary of the Thesis

In Chapter 3, we investigated the problem of sparse multi-agent discrete search without false alarms. The agents have limited visibility, i.e., each agent can only search a subset of the locations. For the special case where the agents are homogeneous and thus the detection performance only depends on the location, we provided a novel perspective of viewing the problem as a min-cost flow problem, and gave the optimality conditions for the agent allocation to be optimal. We developed a fast specialized algorithm for solving the problem. The algorithm uses a heap to create one unit of demand at a sink node at each iteration, then finds an augmenting path to assign the demand. When it fails to do so, we identify and remove an isolated group of nodes from further consideration. We proved that the algorithm can always find an optimal solution in finite time, and analyzed the time complexity of the algorithm. We performed experiments to compare our specialized algorithm with a general min-cost flow algorithm, the capacity-scaling algorithm.

We also addressed the general case where the agents are heterogeneous and thus the detection performance depends on both location and agent, which is known to be NP-hard. We showed that the problem can be reduced to a submodular maximization problem over a matroid. We developed an approximate algorithm with guaranteed performance and a block coordinate ascent algorithm. We also provided an upper bound of the optimal value.

In Chapter 4, we studied the problem of multi-agent sparse search of a moving object. The motion of the object follows a time-homogeneous Markov chain with known transition probability matrix. Similar as in Chapter 3, the agents have limited visibility. The agents only have discrete search efforts, resulting in an integer problem.

We provided necessary conditions for an optimal search plan. We developed a forward-and-backward (FAB) algorithm which is a block coordinate descent algorithm, and illustrated with counterexample that it does not always find an optimal solution. We then developed a two-stage FAB algorithm. The first stage of the algorithm focuses on a relaxed problem that allows continuous search effort, in preparation for initializing the second stage which focuses on the integer problem. We proved that the first stage of the algorithm converges to an optimal solution of the relaxed problem.

In experiments, we used the RS algorithm in [Royset and Sato, 2010] to compute the relative optimality gaps achieved by our two-stage FAB algorithm. Our algorithm found solutions of less than 5% relative optimality gap for models of size 3 agents, 225 locations and 10 time periods in a few seconds. Experiments also showed that the two-stage FAB algorithm finds better objective values than its three variants. It scales well as the problem size grows too.

We provided fast specialized algorithms to solve the subproblems in the two-stage FAB algorithm. Our algorithm for solving the integer subproblems is a specialized primal-dual algorithm. It has a time complexity much smaller than general min-cost flow problems. Our algorithm for solving the relaxed subproblems is based on the block coordinate descent method. Experiments showed that it is orders of magnitude faster than general constrained nonlinear optimization algorithms.

We also generalized the problem to the case of multiple heterogeneous agents where the probabilities of detection do not only depend on the locations and the

time periods, but also on the agents. We formulated the problem as maximizing a monotone submodular set function over matroid constraints. We proposed two greedy algorithms and prove that a greedy-style algorithm has an approximation ratio of $\frac{1}{2}$. We also provided a lower bound of the optimal objective value.

In Chapter 5, we presented a new class of fast algorithms for solving budget and motion constrained search posed as orienteering problems. The algorithms are based on using approximate algorithms for task assignment based on knapsack problems, while using estimates of required resources based on spanning tree information as a surrogate for required tour distances. We subsequently refined the tours and task assignments using traveling salesperson approximations such as the Lin-Kernighan-Helsgaun algorithm. We first developed algorithms for single agent problems, then developed extensions for teams of agents.

We conducted extensive numerical experiments comparing our algorithms with other proposed fast algorithms for orienteering problems on benchmark problems as well as on a physical scenario derived from LANDSAT data. Our results indicate that our new algorithms have stable computation times across multiple scenarios, and achieve collected rewards that are significantly higher than the competing algorithms across most scenarios tested.

In Chapter 6, we studied the problem of optimal adaptive search by a team of agents collecting noisy observations concerning the presence of an object in observed areas. Our results extend the prior work of [Jedynak et al., 2011] by considering teams of agents with correlated measurement errors. We established an equivalence between the correlated error multi-agent problem and a single-agent, multi-region problem, and exploited this equivalence to derive an explicit solution for the optimal value function and the optimal strategies. The optimal strategies are characterized by a unique, optimal joint operating point, which can be obtained as the solution of

a strictly concave maximization problem.

When agent errors are independent across agents, the concave maximization problem for the optimal joint operating point can be decoupled into individual scalar concave maximization problems, leading to a simple computational procedure for its solution. We established sufficient conditions for characterizing symmetry properties of general error models that allow for the analytic solution for the optimal joint operating point.

We extended our multi-agent formulation with independent errors to the case where agents can choose between different precision modes as well as observation regions, at a cost that depends on the precision mode selected, extending the single agent results in [Sznitman et al., 2013]. The optimal strategies are again characterized by an optimal joint operating point and set of optimal joint precision modes. We showed that computing the optimal joint operating point mode and precision modes again decouples into single-agent scalar concave maximization problems.

We further studied the case where there are multiple objects to search. We derived an explicit solution to Bellmans equation for the cost-to-go, as well as a construction for the optimal strategies that involved solution of a strictly convex minimization problem. The resulting optimal control policies were evaluated in simulations involving two objects and two agents, which illustrated that the expected empirical cost-to-go agreed closely with the theoretical prediction from the solution of Bellmans equation. We compared this empirical cost with the cost of a heuristic based on geographic partitioning, and found that the optimal strategies had significant improvement in performance.

7.2 Directions for Future Work

There are several directions in which the results in Chapter 3 can be extended. One direction is to study multi-agent whereabouts search without false alarms where the objective is to find an optimal agent allocation to maximize the probability of correctly stating where the object is. A second direction is to consider when one search action of different agents cost different amounts of effort. Optimal strategies may not exist for this problem; techniques such as branch and bound may be needed. A third direction is to extend to cases where false alarm measurements are possible.

There are several directions to extend the problem studied in Chapter 4. One direction is to consider searching for multiple objects. Possible objectives for this problem include maximizing the number of objects detected over a fixed horizon. A second direction is to consider both agent motion constraints where each agent is constrained to moving to a subset of locations depending on its present location and agent visibilities. In this case, one needs to carefully construct feasible search plans that satisfy the motion constraints. A third direction is to extend to cases where false alarm measurements are possible.

There are several directions to enhance the algorithms developed in Chapter 5. In particular, for the team orienteering problem, one may come up with different ways to initialize trees for the agents. One may also consider using other approaches to select nodes to visit in the tours other than greedily growing trees.

There are several directions in which the results in Chapter 6 can be extended. One direction is to develop upper bounds on the mean squared error for selected optimal algorithms. To the best of our knowledge, the only upper bound on the expected error was developed in [Waeber et al., 2011] for the case of single-agent search of an object in a unit interval with binary symmetric errors using Horstein's probabilistic bisection algorithm [Horstein, 1963]. One may consider generalizing the

result in [Waeber et al., 2011] to single or multiple agents with general measurement error models. A second extension is to develop approaches for approximating the optimal search strategies when agents consist of physical platforms with sensing area constraints; similar issues arise in the results in classical search theory [Stone, 1975]. Another possible extension is to consider single or multiple moving objects.

References

Ahlswede, R. and Wegener, I. (1987). *Search Problems*. Wiley.

Ahuja, R. K., Kumar, A., Jha, K. C., and Orlin, J. B. (2007). Exact and heuristic algorithms for the weapon-target assignment problem. *Operations Research*, 55(6):1136–1146.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall.

Alajaji, F. and Chen, P.-N. (2015). *Information Theory for Single User Systems*. Springer.

Arkin, V. I. (1964a). A problem of optimum distribution of search effort. *Theory of Probability & Its Applications*, 9:159–160.

Arkin, V. I. (1964b). Uniformly optimal strategies in search problems. *Theory of Probability & Its Applications*, 9:674–680.

Atia, G. and Saligrama, V. (2012). Boolean compressed sensing and noisy group testing. *IEEE Transactions on Information Theory*, 58(3):1880–1901.

Bangla, A. K. and Castañón, D. A. (2013). Forward and reverse auction algorithms for nonlinear resource allocation. In *Proceedings of the IEEE Conference on Decision and Control*, Florence, Italy.

Benkoski, S. J., Monticino, M. G., and Weisinger, J. R. (1991). A survey of the search theory literature. *Naval Research Logistics*, 38(4):469–494.

Bertsekas, D. P. (2005). *Dynamic Programming and Optimal Control*, volume 1-2. Athena Scientific, third edition.

Bertsekas, D. P. and Shreve, S. E. (1978). *Stochastic Optimal Control: the Discrete Time Case*. Academic Press, Orlando.

Black, W. L. (1965). Discrete sequential search. *Information and Control*, 8(2):159–162.

Blum, A., Chawla, S., Karger, D. R., Lane, T., Meyerson, A., and Minkoff, M. (2007). Approximation algorithms for orienteering and discounted-reward TSP. *SIAM Journal on Computing*, 37(2):653–670.

Brown, S. S. (1980). Optimal search for a moving target in discrete time and space. *Operations Research*, 28(6).

Burnashev, M. V. and Zigangirov, K. (1974). An interval estimation problem for controlled observations. *Problemy Peredachi Informatsii*, 10(3):51–61.

Calinescu, G., Chekuri, C., Pál, M., and Vondrák, J. (2011). Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766.

Castañón, D. A. (1987). Advanced weapon-target assignment algorithm. Technical report, ALPHATECH, Inc., Burlington, MA.

Castañón, D. A. (1995). Optimal search strategies in dynamic hypothesis testing. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(7):1130–1138.

Castro, R. and Nowak, R. (2008). Active learning and sampling. In Hero, A., Castañón, D. A., Cochran, D., and Kastella, K., editors, *Foundations and Applications of Sensor Management*. Springer.

Chao, I.-M., Golden, B. L., and Wasil, E. A. (1996a). A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475–489.

Chao, I.-M., Golden, B. L., and Wasil, E. A. (1996b). The team orienteering problem. *European Journal of Operational Research*, 88(3):464–474.

Chao, I.-M., Golden, B. L., and Wasil, E. A. (1996c). *Test instances in [Chao et al., 1996b]*. `http://www.mech.kuleuven.be/en/cib/op`.

Chew, M. C. (1967). A sequential search procedure. *The Annals of Mathematical Statistics*, pages 494–502.

Cover, T. M. and Thomas, J. A. (2012). *Elements of Information Theory*. John Wiley & Sons.

Davey, S., Gordon, N., Holland, I., Rutten, M., and Williams, J. (2016). *Bayesian Methods in the Search for MH370*. Springer.

Dezső, B., Jüttner, A., and Kovács, P. (2011). LEMON – an open source C++ graph template library. *Electronic Notes in Theoretical Computer Science*, 264(5):23–45.

Ding, H., Cristofalo, E., Wang, J., Castañón, D. A., Montijano, E., Saligrama, V., and Schwager, M. (2016). A multi-resolution approach for discovery and 3D modeling of archaeological sites using satellite imagery and a UAV-borne camera. In *Proceedings of the American Control Conference*, Boston, MA.

Dobbie, J. M. (1974). A two-cell model of search for a moving target. *Operations Research*, 22.

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306.

Du, D.-Z. and Hwang, F. K. (1993). *Combinatorial Group Testing and Its Applications*. Series on Applied Mathematics. World Scientific, Singapore.

Eagle, J. (1984). The optimal search for a moving target when the search path is constrained. *Operations Research*, 32(5).

Eagle, J. and Yee, J. (1990). An optimal branch-and-bound procedure for the constrained path, moving target search problem. *Operations Research*, 38(1).

Everett, H. (1963). Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11:399–417.

Fisher, M. L., Nemhauser, G. L., and Wolsey, L. A. (1978). An analysis of approximations for maximizing submodular set functions - II. *Mathematical Programming Study*, 8:73–87.

Frank, A. and Asuncion, A. (2010). *UCI Machine Learning Repository*. `http://archive.ics.uci.edu/ml`.

Gallager, R. G. (1968). *Information theory and reliable communication*, volume 2. Springer.

Gluss, B. (1959). An optimum policy for detecting a fault in a complex system. *Operations Research*, 7(4):468–477.

Goldberg, A. V. (1997). An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22(1):1–29.

Golden, B. L., Levy, L., and Vohra, R. (1987). The orienteering problem. *Naval Research Logistics*, 34(3):307–318.

Grippo, L. and Sciandrone, M. (2000). On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, 26.

Helsgaun, K. (2000). An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130.

Horstein, M. (1963). Sequential transmission using noiseless feedback. *IEEE Transactions on Information Theory*, 9(3):136–143.

Jedynak, B., Frazier, P. I., and Sznitman, R. (2011). Twenty questions with noise: Bayes optimal policies for entropy loss. *Journal of Applied Probability*, 49.

Kadane, J. B. (1968). Discrete search and the Neyman-Pearson lemma. *Journal of Mathematical Analysis and Applications*, 22:156–171.

Kadane, J. B. (1971). Optimal whereabouts search. *Operations Research*, 19(4):894–904.

Kadane, J. B. and Simon, H. A. (1977). Optimal strategies for a class of constrained sequential problems. *The Annals of Statistics*, pages 237–255.

Kelly, D. J. and O'Neill, G. M. (1991). The minimum cost flow problem and the network simplex solution method. Master's thesis, University College Dublin.

Koopman, B. O. (1946). Search and Screening. Operations Evaluation Group Report No. 56. Technical report, Center for Naval Analyses, Alexandria, VA.

Koopman, B. O. (1953). The optimum distribution of effort. *Journal of the Operations Research Society of America*, 1:52–63.

Koopman, B. O. (1956a). The theory of search, part I. Kinematic bases. *Operations Research*, 4:324–346.

Koopman, B. O. (1956b). The theory of search, part II. Target detection. *Operations Research*, 4:503–531.

Koopman, B. O. (1956c). The theory of search, part III. The optimum distribution of searching effort. *Operations Research*, 4:613–626.

Kraft, D. (1988). A software package for sequential quadratic programming. Technical report, DLR German Aerospace Center Institute for Flight Mechanics, Koln, Germany.

Kress, M., Royset, J. O., and Rozen, N. (2012). The eye and the fist: Optimizing search and interdiction. *European Journal of Operational Research*, 220(2):550–558.

Lloyd, S. P. and Witsenhausen, H. S. (1986). Weapons allocation is NP-complete. In *Proceedings of the Summer Computer Simulation Conference*.

Matula, D. (1964). A periodic optimal search. *The American Mathematical Monthly*, 71(1):15–21.

Pollock, S. (1970). A simple model of search for a moving target. *Operations Research*, 18.

Rajan, P., Han, W., Sznitman, R., Frazier, P., and Jedynak, B. (2015). Bayesian multiple target localization. *Journal of Machine Learning Research*, 37:1945–1953.

Richardson, H. R. and Stone, L. D. (1971). Operations analysis during the underwater search for Scorpions. *Naval Research Logistics Quarterly*, 18:141–157.

Royset, J. O. and Sato, H. (2010). Route optimization for multiple searchers. *Naval Research Logistics*, 57(8).

Silberholz, J. and Golden, B. (2010). The effective application of a new approach to the generalized orienteering problem. *Journal of Heuristics*, 16(3):393–415.

Song, N.-O. and Teneketzis, D. (2004). Discrete search with multiple sensors. *Mathematical Methods of Operations Research*, 60(1):1–13.

Staroverov, O. V. (1962). On a searching problem. *Theory of Probability & Its Applications*, 8(2):184–187.

Stewart, T. J. (1980). Experience with a branch-and-bound algorithm for constrained searcher motion. In Haley, K. B. and Stone, L. D., editors, *Search Theory and Applications*. Springer.

Stone, L. D. (1975). *Theory of Optimal Search*. Academic Press New York.

Stone, L. D. (1992). Search for the SS Central America: Mathematical treasure hunting. *Interfaces*, 22:32–54.

Stone, L. D., Keller, C. M., Kratzke, T. M., and Strumpfer, J. P. (2014). Search for the wreckage of Air France Flight AF 447. *Statistical Science*, pages 69–80.

Sun, T., Li, Y., Xu, L., and Kelly, K. F. (2014). Compressive imaging and spectroscopy–beyond the single pixel camera. In *Fringe 2013*, pages 109–115. Springer.

Sznitman, R., Lucchi, A., Frazier, P. I., Jedynak, B., and Fua, P. (2013). An optimal policy for target localization with application to electron microscopy. In *Proceedings of the International Conference on Machine Learning*.

Tognetti, K. P. (1968). An optimal strategy for a whereabouts search. *Operations Research*, 16(1):209–211.

Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, pages 797–809.

Tsiligkaridis, T., Sadler, B. M., and Hero, A. O. (2014). Collaborative 20 questions for target localization. *IEEE Transactions on Information Theory*, 60(4):2233–2252.

Van Trees, H. L. (2004). *Detection, Estimation, and Modulation Theory*. John Wiley & Sons.

Vansteenwegen, P., Souffriau, W., and Van Oudheusden, D. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10.

Waeber, R., Frazier, P. I., and Henderson, S. G. (2011). A Bayesian approach to stochastic root finding. In *Proceedings of the 2011 Winter Simulation Conference*, pages 4033–4045.

Wagner, D. H. and Stone, L. D. (1974). Necessity and existence results on constrained optimization of separable functionals by a multiplier rule. *SIAM Journal on Control*, 12:356–372.

Wang, J., Trapeznikov, K., and Saligrama, V. (2014). An LP for sequential learning under budgets. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 987–995.

Washburn, A., Royset, J. O., and Stone, L. D. (2016). *Optimal Search for Moving Targets*. Springer.

Washburn, A. R. (1975). Search for a moving diffusing target. *Proceedings of the ORSA/TIMS Conference, Las Vegas*.

Washburn, A. R. (1980). On search for a moving target. *Naval Research Logistics Quarterly*, 27:315–322.

Washburn, A. R. (1983). Search for a moving target: The FAB algorithm. *Operations Research*, 31(4).

Wegener, I. (1980). The discrete sequential search problem with nonrandom cost and overlook probabilities. *Mathematics of Operations Research*, 5(3):373–380.

Wegener, I. (1981). The construction of an optimal distribution of search effort. *Naval Research Logistics*, 28:533–543.

Wegener, I. (1982). The discrete search problem and the construction of optimal allocations. *Naval Research Logistics*, 29:203–212.

Welsh, D. J. A. (1976). *Matroid Theory*. Academic Press.

Zahl, S. (1963). An allocation problem with applications to operations research and statistics. *Operations Research*, 11:426–441.

CURRICULUM VITAE