Foraging, Prospecting, and Falsification - Improving Three Aspects of Autonomous Science

P. Michael Furlong

CMU-RI-TR-18-30

May 27, 2018

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Dr. David Wettergreen, Chair
Dr. Jeff Schneider
Dr. Nathan Michael
Dr. David R. Thompson (JPL)

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

This work was supported by NSERC, and the following NASA-funded projects: Robotic Investigation of Subsurface Life in the Atacama Desert (NNX11AJ87G), Reliable Autonomous Surface Mobility (NNX10CF52P), The Mojave Volatiles Prospector, Resource Prospector, ARADS, ICICLES

Opinions expressed in this document are solely those of the author.

Keywords: Science Autonomy, Exploration-Exploitation, Active Learning, Foraging, Prospecting, Falsification

To my parents, Alice Prim and Michael Furlong

Abstract

Robots exploring the subsurface ocean of Europa, for example, will not have reliable communications with scientists on Earth. Robots exploring with unreliable communications must conduct scientific exploration autonomously. Automation of scientific exploration requires both opportunistic and deliberative decision-making algorithms.

Opportunistic decision making chooses to investigate events or phenomena which could not have been anticipated from prior knowledge but which may yield valuable scientific data. Deliberative decision making uses prior knowledge to plan observations that increase information. Approaches to deliberative and opportunistic science autonomy that work in the laboratory may not work in the field.

This thesis presents three algorithms designed to improve the performance of robots conducting autonomous science investigations. The first algorithm, foraging, improves opportunistic responses by deciding between sampling immediately available objects or searching for better options. Foraging moves science autonomy beyond simply responding to matched templates or anomalous data. The work recognizes that robots may not get to choose which objects they can to sample, but must deal with what they encounter. Our approach has increased performance in selecting objects to sample when sampling costs are high, without neglecting opportunities when sampling costs are low.

The second algorithm addresses how to effectively conduct prospecting without relying on either arbitrary thresholds or responding to anomalies. Threshold-based algorithms cannot distinguish between anomalies and true changes in the distribution driving sensor readings. We present an algorithm that directly poses the question of whether or not a change has occurred. The change detection algorithm developed in this thesis encodes a level of confidence that a change has occurred, based on data collected by the robot. This can improve the efficiency of the investigation.

The third algorithm represents a new approach to information gathering based on falsification. Recognizing scientists come to missions with hypotheses formulated, the algorithm uses those hypotheses to choose sampling actions that help determine which hypothesis is most credible. Prior information gathering approaches consider one or fewer hypotheses, and focus mainly on sampling the hypotheses' domain. We show that our approach biases the belief in hypotheses towards the most credible one.

The thesis proven in this work is that **accounting for operational and environmental context improves science autonomy algorithms**. Each of these algorithms improve components of autonomous science, and thereby the process as a whole.

Acknowledgments

This work was funded by NSERC, as well as a number of NASA-funded projects. At CMU I was funded by the grants Robotic Investigation of Subsurface Life in the Atacama Desert (NNX11AJ87G) and Reliable Autonomous Surface Mobility (NNX10CF52P). At NASA my work was funded by the Mojave Volatiles Prospector, Resource Prospector, ARADS, and ICICLES projects. Completing the work would not have been possible without the support and assistance of many people.

First, my exceedingly patient advisor, David Wettergreen. He has put up with my non-traditional path through the program, and provided sage guidance. Likewise, my thesis committee members, Jeff Schneider, Nathan Michael, and David R. Thompson, have graciously provided valuable insights and shaped this work into a much better version of itself. Red Whittaker and Reid Simmons also provided valuable advice and criticism.

Thanks to Suzanne Lyons-Muth, Jean Harpley, Sanae Minick, and Lynnetta Miller, who help make the Robotics Institute work. Thanks also to Chuck Whittaker, Jim Teza, David Kohanbash, Srinivasan Vijayarangan, and Dom Jonak. They made the robots work.

Second, I must thank everyone at the Intelligent Robotics Group at NASA Ames, in particular Terry Fong, for his help and counsel. The IRG provided a home away from CMU, excellent work experiences, and support. Thanks to Linda Kobayashi, Lorenzo Flückiger, and Vinh To for facilitating the experiment with KRex2 in the Atacama desert. Thanks also to Brian Glass and the ARADS team for making room in the schedule for that experiment.

Rick Elphic, Tony Colaprete, Mary Beth Wilhelm, Alfonso Davila, Kathryn Bywaters, and Richard Quinn have very generously explained their instruments, scientific operations, and provided data. Mark Shirley illuminated the constraints of mission operations.

Michael Dille, Uland Wong, Brian Coltin, Trey Smith, DW, Oleg Alexandrov, and the rest of the IRG Reading Group provided interesting conversations and conspired on side projects. I had the good fortune to mentor visiting student Akash Arora and NSRTF Fellows Steve McGuire and Eugene Fang. Our collaborations have improved my own work.

Third, thanks to my friends and colleagues from CMU. Dave Silver and Melissa Haas, Dan Munoz, Stephane Ross and Sandra Champagne, Nate and Alyssa Wood, Mark Desnoyer and Catherine Izard, Scott Moreland and Ashley Kidd, Krzysztof and Monika Skonieczny, Ed Hsiao, Breelyn Kane-Styler and Alex Styler, Felix Duvallet, and Scott Satkin furnished good company and advice. Thanks to my officemate Greydon Foil for discussing ideas and providing majestic photos of the robot, Zoë. Wennie Tabib provided insightful advice on presenting the work. Special thanks to the dinner train, including but not limited to: Brian Becker, Heather Jones, Pyry Matikainen, Prasanna Velagapudi, Joydeep Biswas, Ellen Cappo, Eleanor Avrunin, Brina Goyette, Umashankar Nagarajan, Garratt Gallallagher and Nora Flum, Heather Justice, Nate Brooks, Rika Antonova, Peter Anderson-Sprecher, Alvaro Collet Romea, and Siddharth Sannan. They made life in the graduate program fun. Extra special thanks are due to Rachel Vistein, for unflagging encouragement and support.

Finally, I'd like to thank my family who have been unconditionally supportive. My aunt, Lt.Cmdr. (retd) Mary Furlong, was a diligent editor. Alain O'Dea helped turn the document something humans can read. Any remaining language errors are purely my own invention. My brother, Matthew, provided very direct advice on more than one occasion. And last, but not least, my parents. I wouldn't be here without them.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

A natural end-goal of the project of science autonomy is a fully autonomous robot that conducts scientific inquiry without human supervision. However, science autonomy is not a single problem to be solved. In this thesis we demonstrate improvements in different aspects of a robot scientist by being aware of the operational conditions encountered in field work. We examine three different aspects of the scientific process in the context of planetary science missions. Before we discuss the particular subjects of this research, we present the notion of science as a process, and explain which aspects of that process we are improving.

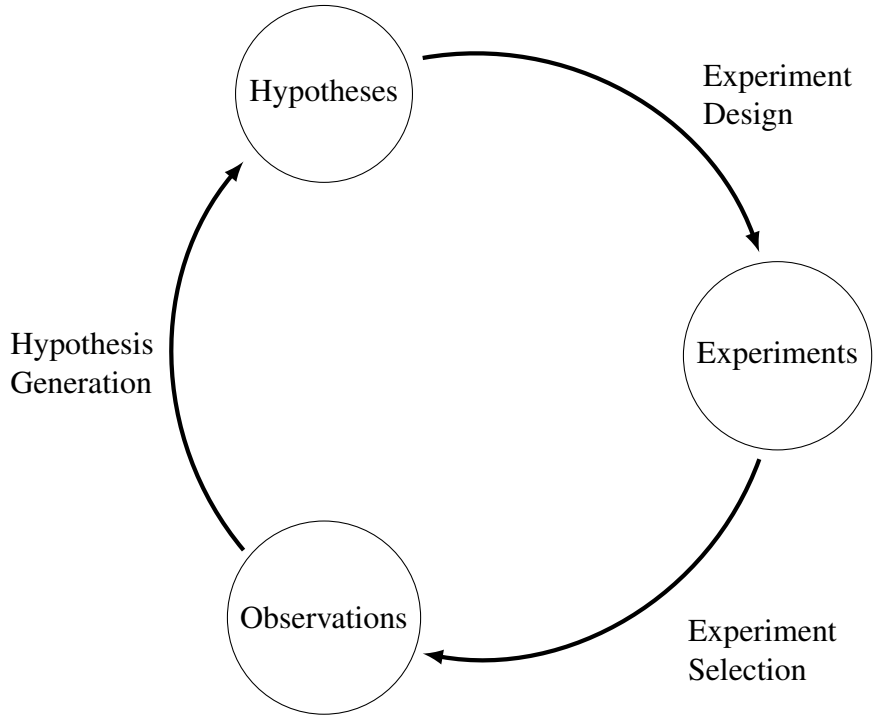


Figure 1.1: One view of the scientific process as a cycle. Hypotheses imply experiments to be conducted. Experiments produce observations. Observations update or produce new hypotheses.

The automation of the scientific process includes many different activities, but at the min-

imum it must include generating hypotheses from data, designing experiments to falsify those hypotheses, and selecting (and conducting) informative experiments - for whichever measure of information the experimenter prefers. One can connect these activities in a cycle, as depicted in Figure 1.1. The activities in the cycle of scientific inquiry act upon three artefacts: hypotheses, experiments, and observations. We define these terms as follows:

Hypotheses are models of some phenomena. Hypotheses must be falsifiable – the must make predictions about the world, that are either implicitly or explicitly observable. For example, a model of the spatial distribution of geologic material predicts that at specific locations there will be specific mixtures of materials. In this document the words "hypothesis" and "model" are used interchangeably. To represent it formally, we assume that a hypothesis, h, maps an input domain, X, to a probability distribution over potential observations, $P(Z)$. In symbols: $h : X \to (Z \to [0, 1])$.

Experiments are actions that the scientist can take that result in observations. It is important to ensure that these experiments are informative with respect to the hypotheses being tested. It is also important to ensure that experiments do not reproduce each other unnecessarily. In this document the word "experiment" and "action" are used interchangeably. In symbols we write $a : X \to Z$. In this thesis we consider only one kind of experiment, those that collect an observation, Z, at a given point in the domain of the hypothesis, X.

Observations are the results of experiments. Observations are the data that are used to construct or update hypotheses. "Observations" and "data" and "samples" are all phrases that will be used interchangeably in this document. In symbols we would write an observation as an input-output pair, (x, z), where $x \in X, z \in Z$. The i^{th} observation would be (x_i, z_i). x and z may be vector-valued, they also may also be random variables.

There is disagreement on which is these artefacts have primacy in the scientific process. Karl Popper would have us believe that hypotheses are the currency of scientists (Popper, 2005). Ian Hacking suggests that experiments are the most important object (Hacking, 1983). We suggest that hypotheses, experiments, and data are all equally necessary components of scientific theory, without one taking primacy over the others.

Science as a process is an iterative loop that one may join in at any point and that all three are perfectly valid starting points. Robot scientists will need to manipulate these artefacts when operating autonomously. Hence, roboticists will need to identify the processes that the robot scientist use to move between these entities.

The process going from hypotheses to experiments is called **Experiment Design**. For example, mutual information sampling is a type of experiment design. It identifies an action that can be taken at a given point in the input space of a hypothesis. Mutual information sampling (Lindley, 1956) also has the advantage of scoring the different candidate actions so they can be prioritized. Another type of experiment design is random selection (Lind, 1757; Peirce and Jastrow, 1884). Random sampling reduces the effect of biases on the selection process and has the advantage of forcing the experimenter to more thoroughly explore the domain of the hypothesis.

The process going from experiment to observations is called **Experiment Selection**. It is not necessarily typical to make a distinction between experiment design and experiment selection. The distinction we make between experiment design and experiment selection is this: Experiment design asks "of all the possible actions, which ones are informative with respect to the hy-

pothesis being tested?". Experiment Selection asks of "Of all the informative actions, and given some budget, which is the best collection of actions to take?". We can sub-classify experiment selection into groups of *reactive* and *deliberative* experiment selection. Both deliberative and reactive experiment selections can be made with respect to the constraints of mission resources.

We consider opportunistic science to be reactive experiment selection. This is because the algorithms can determine sampling actions only once opportunities to conduct experiments have been identified. Reactive sampling actions can only be decided *in situ* as one does not know what opportunities to conduct experiments will be encountered before one explores the environment.

Deliberative experiment selection, determines a set of experiments to be conducted, and then seeks out opportunities to conduct those experiments in the world. Deliberative sampling actions are planned ahead of time in an attempt to maximize the information gained.

The process going from observations to hypotheses is called **Hypothesis Generation**. Simple structured machine learning – i.e. linear regression – is a form of hypothesis generation. It chooses the hypothesis (weight settings) most likely given the constraints of the data to be explained and the family of hypothesis (linear functions) specified by the experimenter. Choosing the structure of the hypothesis is part of the hypothesis generation and so model selection must also be considered part of the hypothesis generation.

By iteratively engaging in hypothesis generation, experiment design, and experiment selection, robots can produce and test hypotheses, thereby improving their understanding of the phenomena they are investigating. Science autonomy, then, must include research into systems that do one or more of experiment design, experiment selection, and hypothesis generation.

In laboratory settings one has to consider the resources that are available to the robot scientist. A limited volume of reagent, for example, may change how one would prioritize different experiments. If a robot scientist is conducting field work there can be even more constraints. Limitations on time or power resources affect how far a vehicle can travel in a day, or whether or not it can backtrack to conduct a follow-up investigation. Noisy sensors complicate the interpretation of data. The robot scientists may not be able to choose which experiments it is able to conduct. These are some of the conditions which exist outside of the laboratory which add additional complications to autonomously conducting science.

In this thesis we address two cases of reactive experiment selection/opportunistic science, which we call "foraging" and "prospecting", in the context of ground vehicle operations. We define foraging to be the execution of discrete sampling actions in response to observing discrete objects in the environment in order to learn those objects' underlying distributions. We define prospecting to be the execution of discrete actions in response to observations of a field which the robot is moving through. We also address an experiment design approach which is based on the theory of falsification, which we then use to direct global exploration for a lander capable of re-ignition.

Of course, this is not an exhaustive listening of all the processes that go into science, and it does not address how multiple robot scientists would coordinate their behaviour. There are problems of perception, sample collection and manipulation, hypothesis generation, and communication of findings which are not addressed in this thesis.

1.1 Why Automate Science?

Having formulated a description of science as a process, we should also address why one would want to automate science. There are at least three reasons to automate the scientific process. First, because future flight missions will require it. Second, because human agents demand it. Third, because it poses a truly unique computational problem. The first two reasons are pragmatic, while the third is more interesting from a philosophical standpoint.

Future Missions Require Science Autonomy

NASA, and other space faring science agencies, will require autonomous science simply because of the strain on productivity that communications latency places on missions. The further afield a robot explorer travels, the lower bandwidth it will have available to communicate back to Earth, and the greater the latency will become.

Including humans in the decision making loop will require either designing robots to survive the decision making cycle or forcing scientists to make less nuanced decisions, potentially leading to less effective missions. Further, with profound delays in communications the likelihood of serendipitous discovery will drop precipitously.

A reliable autonomous scientist reduces science missions' reliance on communications links. If the robot can be treated as a trusted proxy for human scientists then we can send them to locations that would previously be considered unfeasible or wasteful. We engage in a more detailed discussion of when autonomy can aid mission operations in Appendix A.

The algorithms developed for space missions of course have applications here on Earth. Prospecting for resources is an important task whether it is in support of colonies on the Moon or in support of resource extraction for economic actors on Earth.

Humans Require Science Autonomy

The scientific method is a systematic process of generating quantifiable knowledge. Humans, who are riddled with cognitive biases, are a problematic component of this process. Confirmation bias, affirming the consequent, availability bias, and clustering illusions to name a few, interfere with human ability to conduct scientific inquiry.

The literature on the design of experiments, starting at least as early as (Smith, 1918), is an attempt build a mathematically rigorous approach to asking questions that ameliorates human biases. Science autonomy is another step along this path. An illuminating anecdote comes from Daston and Galison's book *Objectivity* (Daston and Galison, 2007, pp.11-13) about Arthur Worthington's research into the physics of how water droplets splatter when the hit a surface.

Worthington, through careful observation with his eye and a flashing light, documented how water droplets would split up upon hitting a hard surface. Circa 1897, he extensively documented symmetry and regularity in the patterns. The symmetry and beauty of his observations spoke to their necessary truth. What happened after he introduced photographic apparatus into his experiments warrants a protracted quotation from (Daston and Galison, 2007):

> For years, Worthington had relied on the images left on his retina by the flash. Then, in spring 1894, he finally succeeded in stopping the droplet's splash with a

> photograph. Symmetry shattered. Worthington said, "The first comment that any one would make is that the photographs, while they bear out the drawings in many details, show greater irregularity than the drawings would have led one to expect." But if the symmetrical drawings and the irregular shadow photographs clashed, one had to go. As Worthington told his London audience, brighter lights and faster plates offered "an objective view" of the splash, which he then had drawn and etched [...]. There was a shock in this new, imperfect nature, a sudden confrontation with the broken particularity of the phenomenon he had studied since 1875. Plunged into doubt, Worthington asked how it could have been that, for so many years, he had been depicting nothing but idealized mirages, however beautifully symmetrical.

Like Worthing we must ensure that observations are derived from meaningful, representative data. We must make sure that they are representing the true underlying statistics of the phenomenon we are concerned about and not imposing some preconceived notion of truth. If we cannot trust human perception to be unbiased, can we trust their decision making? Science autonomy can produce assistive tools for when our rationality fails us.

Science Autonomy is an Interesting Problem

Science autonomy is the mathematisation of the philosophy of science. Scientific inquiry presents an interesting set of behaviours that are designed to be rational and logical, but that need to be produced in a reasonable period of time given finite computing resources.

Science autonomy necessarily fuses reason and deduction with symbols grounded in the real world and in observable quantities. However, unlike other branches of robotics, it is fundamentally important that science autonomy produce information that is human interpretable.

If we contrast science autonomy with other machine learning approaches, there may not be much utility in producing, for example, a hyperparametric function approximation. Arguably a collection of measurements could be more valuable to a remote team of scientists. In turn, this leads to interesting questions about what it means to understand the universe and what are the limits of human understanding. While this thesis does not address these questions, they do motivate the field in general.

1.2 Thesis Statement

The work presented in this thesis looks at designing algorithms for science autonomy, with particular consideration for operations in the field. They are all underpinned by the notion that **accounting for operational and environmental context will improve the performance of science autonomy algorithms**.

The work in this thesis improves upon the state of the art by improving the level of adaptability in algorithms for planetary robotics. We will consider this hypothesis in three settings: Opportunistic sampling of discrete objects (foraging), Opportunistic sampling in a field (prospecting), and Experiment design for hypothesis falsification (informative action planning).

We improve foraging algorithms by recognizing that agents do not have random access to all objects they would like to sample. We improve prospecting algorithms by recognizing that

decision thresholds decided *a priori* do not necessarily work for unknown environments. We used a decision making process that detects changes in distribution underlying observations with confidence, in order to make prospecting decisions.

We improve informative action planning by recognizing that scientists come to missions with pre-existing hypotheses, and we can use them to direct information gathering actions. Motivated by the concept of falsification, we incorporate the hypotheses into the planning process.

The experiments in this thesis mirror settings from analog and planned science missions. We extend the state of the art in science autonomy algorithms by considering the missions constraints from field work.

1.2.1 Opportunistic Sampling of Discrete Objects (Foraging) - Chapter 3

Opportunistic sampling of discrete objects is perhaps best exemplified by robots like NOMAD and the OASIS/AEGIS system deployed on the Mars rovers. Robots identify something in the environment – a rock, a strong match with a template – and then decide to sample it. This is very much a reflexive, stimulus-response sort of behaviour, if an object of interest is identified, then sample it. The purpose of sampling is to determine the distributions underlying these classes of discrete objects.

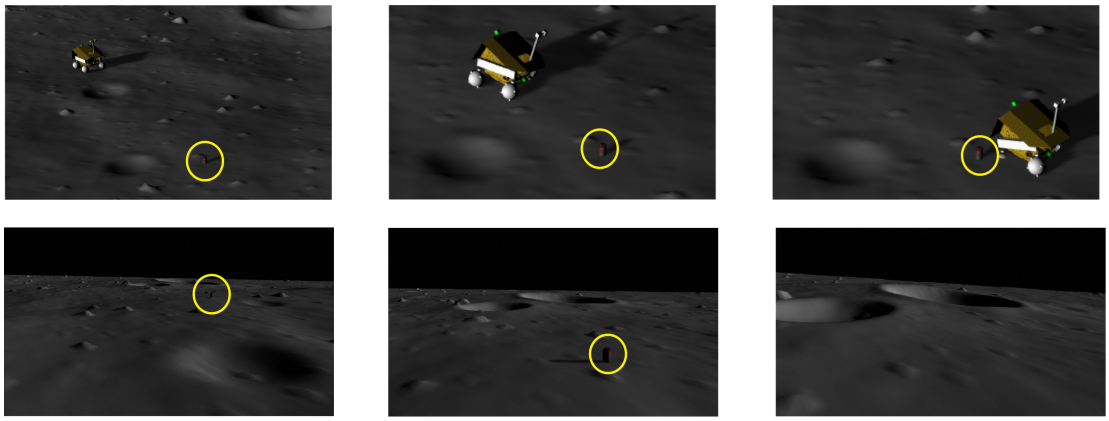


Figure 1.2: An example of a robot which encounters discrete objects in the environment. When the object, marked by the yellow circle, is far away it may not be identified by the perception system of the robot. As the robot gets closer to the object it can then be resolved by the perception system. Finally, as the robot passes the discrete object it no longer can affect the perception system. Images produced using the Resource Prospector simulator courtesy NASA Ames Intelligent Robotics Group.

The state of the art algorithms consider the value of these discrete objects to be binary, and they do not consider the result of sampling these objects. Additionally the state of the art algorithms in both science autonomy and statistical experiment design do not consider the distribution with which these sampling opportunities are encountered. Approaches from stream active learning which do model environmental conditions do not consider costs in a manner appropriate for field operations.

In this chapter we demonstrate that our algorithm, called Foraging, presents an improvement over the practice of always sampling available objects, for large sampling costs. We also

demonstrate that the sampling algorithm which uniform distributes samples among object types performs better than our proposed algorithm for some cost settings, but it also performs worse than both our algorithm and the algorithm which samples everything it encounters.

The proposed algorithm represents a compromise algorithm which has some of the improvement of the Uniform sampling strategy without losing the advantages of the strategy of always sampling. We have identified different regimes of sampling and exploration costs which favour the different algorithms, knowledge which could be used to plan mission sampling strategies.

1.2.2 Opportunistic Sampling In a Scalar Field (Prospecting) - Chapter 4

Opportunistic sampling in a scalar field, or prospecting, is the navigation a space in order to determine locations considered valuable with respect to an objective function, illustrated in Figure 1.3. In this chapter we consider robots that deploy discrete actions in response to the readings from its proxy sensor. A neutron spectrometer, for example, is a proxy sensor for water deposits, and is slated to be part of the resource prospector mission (Andrews et al., 2014). The neutron spectrometer will report a reading at any point in the environment, and even subtle changes in the sensor position can change the readings of the sensor.

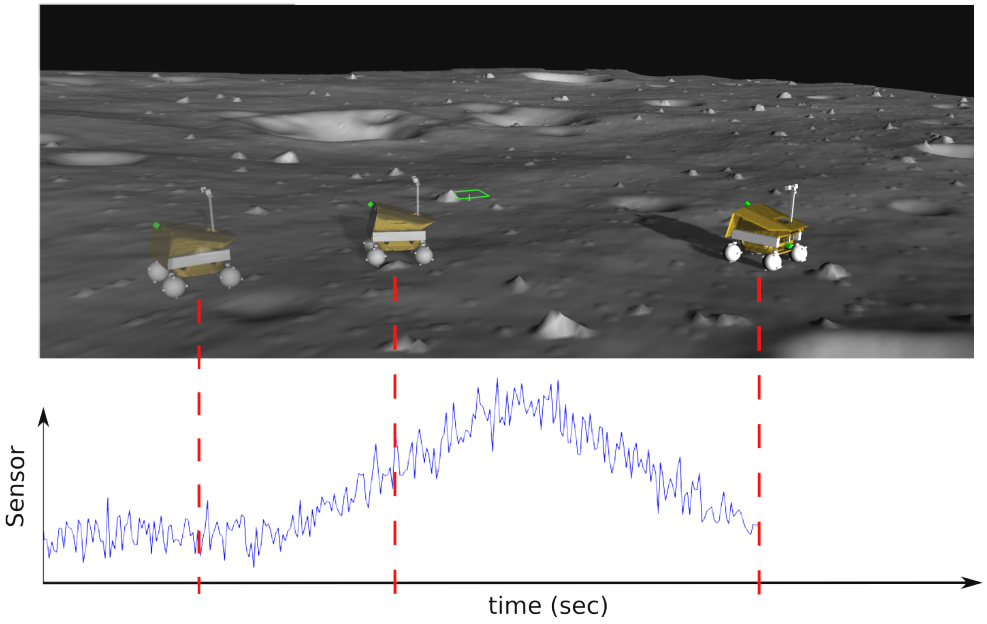


Figure 1.3: This image illustrates the notion of prospecting. The transparent versions of the robot represent previous points in time. For every location the robot passes through the primary sensor gives a valid reading. The robot has to choose where to deploy a sampling action in the field.

In practice, autonomous robots conduct prospecting by monitoring the readings from the prospecting sensor and if the reading crosses a threshold then deploy the sampling action. The threshold may be on the raw data being reported from the prospecting sensor or it may present in the form of a threshold in the surprise of a single reading, given a distribution. In either case, the

response is to an immediate sensor reading, and not to the belief that a change in the underlying distribution has occurred.

Our approach is able to detect sub-threshold changes in sensor readings, and does not need to be tailored to every individual setting. We demonstrate on real-world data from the Mojave Volatiles Prospector (MVP) project that our algorithm is able to detect sub-threshold distribution changes that are not detected using the threshold approach. The threshold used in our experiment was determined for this test site and the data were all collected in the same region. This illustrates the strength of our algorithm compared to the those used in practice.

The algorithm gives a measure of confidence in the belief that the underlying distribution has changed, which can be tailored to the risk posture of a mission. It could also let the autonomous scientist judge accordingly how risky an action would be, given mission resources and constraints. In principle the technique used in this chapter should translate to vector fields as well, but that application remains for future work.

1.2.3 Global Planning for Hypothesis Falsification - Chapter 5

State of the art in informative path planning focusses on collecting data that can be used to estimate some unknown function, for example the spatial distribution of subsurface halite in an environment. The paths are formulated to maximize information gained, for some measure of information gain. However, these approaches assume that the robots are engaged in exploration with no or one hypothesis being investigated as part of the mission.

In this chapter we examine the problem of designing experiments, and hence selecting sampling points in 2D space, for a robot collecting observations that are designed to distinguish which of a set of hypotheses are most likely to be correct. The robot is considering multiple competing hypotheses about what it would observe at different locations in space, and it needs to determine sampling locations which have the greatest likelihood of distinguishing between hypotheses. This is an information gain problem, but instead of trying to maximize information gained in the distribution over the parameters of one hypothesis, we are attempting to maximize information gained in the belief over the hypotheses themselves.

Our approach is a sampling based approach that is agnostic of the form of the hypotheses under investigation. As with other sampling-based planning, the approach lends itself to designing experiments in 2d environments, but also in more abstract mathematical spaces.

We demonstrate that by being aware of the hypotheses under investigation we are able to determine paths that more readily identify which of a set of hypotheses best explain the data. We also demonstrated that when none of the hypotheses are "correct" (i.e. a probability of predicting the observations > 0.5) our proposed algorithm is better at predicting that none of the hypotheses should be considered.

1.3 Scope of Work

Science autonomy touches on many aspects of robotics and artificial intelligence. Consequently we feel it is important to clearly demarcate the scope of this work. The problem settings tackled

in this thesis are narrowly focused on the author's work at NASA Ames Intelligence Robotics Group, although the algorithms developed may have applications outside of the studied domains.

We do not consider perception problems in this thesis. Perception is, in itself, a wide field of work. We assume that some mechanism capable of identifying discrete objects in the world is available to our robot scientist. Recent advances in machine learning for perception promise that even better systems may be available in the future. It would be beneficial, but sample-expensive, to use samples collected in a science mission to help train a perception system to distinguish meaningful features in sensor data.

We do not address efficient path planning. For the first two components of the work we assume that a trajectory has already been given to the robot and our algorithms are able to interrupt that trajectory. Opportunistic science algorithms which are path-agnostic may make less efficient decisions, but they can be more easily integrated into an existing robot. This situation aligns with anticipated operations in future NASA missions, where mission planners may wish to determine long-range trajectories themselves.

This is not to suggest that there is no role for planning algorithms in autonomous space missions. The work presented in Chapter 3 and Chapter 4 can work in conjunction with existing trajectory planners. Separating the concerns of the planner from those of the proposed algorithms reduces the search space a planner must explore, making execution time faster, but possibly at the cost of performance of the overall system.

In the third component of this thesis we consider a greedy planner. Here our problem setting is with a landing vehicle that is capable of reignition and visiting multiple sites, and as such does not pass through intermediate points in the map between goals. While more efficient path planning would be beneficial, the main purpose of the chapter is to consider the difference between mutual-information sampling in the domain of the hypothesis and mutual information sampling in the belief space of a set of hypotheses. Additionally, as this algorithm also has applications to exploring domains where it is possible to have random access to elements in the domain of the hypotheses, path planning is not necessarily a meaningful concept.

The work covered in Chapter 5 revolves around the falsification objective, which can be used by a planner as a reward function. This is a form of mutual information sampling, which helps resolve which of a set hypotheses is most accurate. This sort of a planner can act as an assistive tool, helping negotiate mission priorities between competing hypotheses.

From a broader perspective, planners can play an important part of autonomous missions. The MER rovers employed a variation on the D^* planner (Maimone et al., 2006). While that is an example of a planner being deployed tactically on-board an autonomous robot, planners can also be useful from an strategic mission planning perspective.

Mission planners need to design trajectories for robots to follow, this requires weighing mission objectives and constraints. This places the burden on operators to think at both a strategic and a tactical level. We can use automated planners to alleviate this pressure, by letting human planners design the mission objective function, and having algorithms design the trajectories, then have the humans accept or reject the proposed trajectory.

Using planning algorithms in this way gives remote science teams a different language for interaction with robotic systems. It would let them focus on the strategic objectives – maximizing the science return – and simply evaluate the produced trajectories, modifying the planner's objective function until a satisfactory result is returned. This also produces an opportunity for

the algorithmic planners to learn the human mission planners' utility function, in order to better determine what are and are not acceptable trajectories.

We do not consider multiple communicating robot scientists, nor do we consider hypothesis generation, summarizing hypotheses and data for human consumption, or prioritizing data for downlink. We do highlight some approaches to these problems in Chapter 2, but the topics themselves are outside of the scope of this thesis.

1.4 Summary

We view the scientific process as a collection of tools for designing experiments, conducting experiments, and using collected data to update or generate new hypotheses. We believe by accounting for operational and environmental context we can improve the performance of some of these tools. In this thesis we consider two reactive, opportunistic science algorithms, and a third, deliberative science algorithm which plans actions to determine which of a set of hypotheses are most likely to be correct.

Chapter 2

Related Work

Robots have been involved in collecting scientific information since at least the Voyager probes (Krimigis et al., 1983). Since then vehicles have used increasing levels of autonomy, as exemplified by the robot Zoë from the Life in the Atacama Desert project (Thompson, 2008). In this chapter we consider robots that have been engaged in field work, specifically planetary science and underwater exploration.

It should be stressed that while some of the approaches discussed in this chapter are considered more or less autonomous, this is not meant to be a value judgement of the work. Even vehicles which ferry instruments in order to collect data, but make no decisions about data collection, can be vital components of an autonomous science mission.

Further, the algorithms covered in this chapter are discussed in a way that strictly considers the algorithms themselves, and not the concept of operations in which they are embedded. Algorithms may appear to be brittle when considered out of context, but with a remote science team in the loop the system as whole can be highly capable, and adaptive to the environment.

First we briefly discuss robots which engage in relatively passive data collection. Next we review work relevant to this thesis, namely opportunistic sampling and informative action selection. Then we review hypothesis generation research, which motivates the falsification-based informative action selection developed in Chapter 5.

2.1 Passive Sampling

The first robot scientists were simple probes that carried scientific instruments, recorded data and relayed those data to remote humans. Any decisions about how and where to collect samples were made off-board the robot. Using robots eliminates risks to human life that would be incurred by sending humans to explore, not to mention the costs of bringing humans back.

Early examples of these robot scientists include NASA's Voyager (Krimigis et al., 1977, 1983) and Surveyor (Ezell, 1988, pp 325-331) probes, as well as the Venera (Vakhnin, 1968) and Vega (Sagdeev and Moroz, 1986) programs. The Vega landers, despite Vega 1 failing during landing and Vega 2 lasting only 56 minutes on the surface, returned information about the Venusian surface and atmosphere without any intelligence or autonomy (Bertaux et al., 1996; Surkov et al., 1986). Similarly the Viking landers revolutionized the understanding of the Martian sur-

face and hydro-geological processes there (Raeburn, 1998).

Passive sampling methods are by no means obsolete. The Phoenix (Arvidson et al., 2009) and Philae (Hand, 2014) landers collected data from their respective landing sites, without ever leaving them. The proposed Titan Mare Explorer was a vehicle which landed on a liquid body and passively follow its currents (Stofan et al., 2013).

These space exploration missions may not necessarily fit the image one might have of a robot explorer. A vehicle that makes decisions about where it travels, where it collects samples, and what it does with those samples once collected seems more appropriate. However, when little is initially know, collecting any data is valuable. On an alien body one lander can reveal a wealth of information without needing to be fully autonomous.

More recognizable as a robot, the Dante II robot (Bares and Wettergreen, 1999) demonstrated the utility of controlled mobility when collecting observations. Dante II was designed to descend into volcanoes to collect valuable scientific information. Vulcanologists have died collecting samples from volcanoes. The success of Dante II underscores how robotic scientists can reduce risks to human life while enabling scientific information gathering.

Kunz et al. (2008) describes operations with two robots, Puma and Jaguar, which conducted undersea exploration. Puma and Jaguar operated under the Arctic ice sheet as an analogue for operations on Europa or Enceladus. There the vehicles collected conductivity, temperature, and depth readings while following a zig-zag pattern covering the surface area of a region under the ice, as well following an oscillating depth profile while exploring. Trajectory following allowed the robots to collect the necessary data without human intervention.

Bingham et al. (2010) used robots to conduct deep water archaeology. Their robot follows a lawnmower pattern, using precise navigation information to co-register a suite of sensor readings and build accurate maps of underwater archaeological sites. Similarly, the robot developed by Furgale et al. (2010) uses good motion estimates to register ground penetrating radar with terrain geometry while following human-defined trajectories. Their algorithm also corrects for the terrain's topography, improving the sensor's behaviour. While these robots do not make decisions about their actions they nevertheless collect valuable and interesting scientific information.

The lawnmower, or boustrophedon, path (LaValle, 2006, §7.6, due to Acar and Choset), which robots like those in (Bingham et al., 2010; Furgale et al., 2010), follow is a special case of a space filling curve. Space filling curves are paths that can pass through every point in an two dimensional region, and have applications in planning for coverage. These curves can also generalize to higher dimensional spaces.

Spires and Goldsmith (1998) have conducted work into using space-filling curves for geographic exploration. They note some benefits of efficiency for multi-robot exploration. However, there are at least two problems with using space filling curves.

The first problem with space filling curves is that they assume an obstacle-free volume. This is unlikely to be the case in field work, requiring algorithms for decomposing the space into unoccupied volumes, or more sophisticated planning algorithms.

The second problem with space filling curves, although not necessarily true for the lawnmower pattern, is that the robots sometimes move away from the goal while following the curve. Depending on the probability of robot failures and the mission constraints, this may not be acceptable.

There are also logistical concerns when using space-filling curves. Choosing the resolution

of the space-filling curve *a priori* does not guarantee a good fit to the spatial resolution of the underlying phenomenon of interest. Space-filling curves may produce trajectories much longer than an informative path planner might, wasting time and resources. Informative path planners gain efficiency over the lawnmower pattern by adapting their sampling to minimize costs.

Despite these shortcomings, it is possible that space-filling curves can satisfy mission requirements, and so they should be considered. In combination with opportunistic science a space filling curve may be more than adequate. Indeed, the lawnmower pattern is used to great effect in research such as (Jakuba and Yoerger, 2008) and (Yoerger et al., 2000). The lawnmower path is still actively used in research, Wilson and Williams (2017) have produced a new means for building coverage within regions of interest defined by the contours of a Gaussian process regression model the vehicle is learning. Their approach to covering the regions of interest is to use the lawnmower pattern.

This is a very small sampling of robotic platforms that have been used to collect scientific data. The important lesson to take here is that these vehicles have provided useful data, and with a minimum of autonomy. Given the validation and verification overhead costs of using autonomous systems in flight missions, simple systems that follow space filling curves can be beneficial and inexpensive. While "simply" collecting data is the first instantiation of robot explorers it is by no means an outdated methodology. When entering a new environment any data collected is extremely valuable. Arriving in one location and collecting observations can be profoundly informative. However, if one has limited sampling resources, then one may need to consider more complex robots.

2.2 Opportunistic Sampling

A more complex autonomous scientist would be one that can react opportunistically to the phenomena it observes[1]. Plans generated before exploring a space necessarily cannot account for unexpected sampling opportunities. Because sampling opportunities encountered can be both unexpected and scientifically valuable, robot scientists must be able to choose to sample opportunistically.

In the previous section, it was assumed that sensors were always recording or effectively free to sample. Since there is no sampling decision to be made, opportunistic sampling is not required, as all observations will be recorded. However, if there are costs for collecting samples, or an instrument must be activated, then the robot needs to make decisions about how to spend its sampling resources. Sampling costs make intelligent opportunistic sampling algorithms a necessity for robots exploring an unknown environment, and increase the likelihood of making serendipitous discoveries.

Opportunistic sampling can be decoupled from the trajectory planning process. Viewed through the window of design of experiment literature opportunistic sampling has parallels in the Multi-Armed Bandit (MAB) (Robbins, 1952) and Secretary Problem(Ferguson, 1989) literature.

[1]While opportunistic sampling algorithms necessarily react to stimuli as they encounter them, the algorithms themselves are not necessarily purely reactive algorithms in the sense of behaviour-based robotics (Brooks, 1991).

Multi-armed bandits are a formulation for addressing the exploration/exploitation problem when deciding which of a set of actions is most rewarding. However, the MAB formulation assumes that it is possible to take any action at any time, which is not necessarily the case when the actions are sampling objects distributed arbitrarily through an environment.

In the secretary problem an agent must pick the best out of a sequence of candidates, where the agent can rank every candidate it encounters. The task also has the constraint that once a candidate has been passed up, the player cannot return to it. This formulation makes the assumption that the decision maker has perfect knowledge about the relative values of different sampling opportunities, which is not necessarily the case when exploring an environment with noisy observations.

2.2.1 Opportunistic Sampling of Discrete Objects (Foraging)

We first consider robots that opportunistically sample discrete objects. Discrete objects are items that show up in the sensor data that can be segmented out from sensor data stream(s), i.e. a rock in an image. Many of these approaches use template matching to identify targets, or variations on these themes such as SVMs or Bayes' nets. The fundamental idea behind them is the same, however, is that an object is identified in the data stream, distinct from its surroundings, and then sampled.

The first robot to consider in opportunistic sampling is the NOMAD robot (Wagner et al., 2001). NOMAD explored the Canadian arctic looking for meteorites. While it was following its path, NOMAD would search camera images for data that matched a visual template. If a template match crossed a threshold then it was determined that a target had been discovered, and NOMAD would collect a sample. However, the results of those samples are not used to determine the relevance of the templates used by the robot. NOMAD may be the first example of an autonomous robot scientist that makes decision in the field. Many of the robots we will discuss in this section use some form of template matching, the implications of which we will discuss below.

In a slightly earlier paper, Gilmore et al. (2000) identifies objects that warrant attention from scientists or that can be put into a downlink for remote human scientists. They use classifiers to determine what part of the sensor data represents an interesting object. The robot directs cameras for higher resolution images, but does not change the overall trajectory of the vehicle. The classifiers that are used are informed by mission-relevant constraints, but they don't consider the quality of the chosen classifiers. The system requires orbital imagery to be collected prior to mission deployment, which limits its ability to operate locations without precursor data. While precursor data collection is happening on the Moon and Mars, there are locations where precursor data can't be collected, such as in underwater caves or in lava tubes.

The Advanced Sciencecraft Experiment (ASE) software that ran on EO-1 identifies events of interest by using the multi-spectrum Hyperion instrument to identify targets of interest (Chien et al., 2003). This work was extended in (Chien et al., 2005), which presents software that ran on the EO-1 that was responsible for "science data analysis, mission planning, and run-time robust execution".

Targets of interest are identified as one of: Thermal anomaly detection, cloud detection, flood scenes, change detection, or other classifications based on on-board classifiers. Each of the

classifiers have an associated score which is used to score potential targets. The classifiers and the relative scoring of outputs is pre-determined by scientists. This can be updated during the mission, but they are not modified by the results of observations by the ASE software itself.

Should identified targets be deemed interesting, and in alignment with scientist-specified mission objectives, then the vehicle trajectory will be modified to collect further observations. The observation plans are then modified to collect more observations of high-ranking targets and to remove planned observations of low-valued targets. Because the EO-1 was constantly orbiting the planet it could take advantage of multiple views of the same location to identify time-varying events in addition to any static features or characteristics that may be of interest to scientists.

The EO-1 employed a planner called CASPER which is capable of building and repairing plans (Chien et al., 2000). CASPER schedules actions in order satisfy mission objectives. CASPER performs plan repair in the case of conflicts between different objectives entered into the plan. CASPER is also used with the AEGIS software, which we discuss below, for the same purpose.

Both ASE and AEGIS employ additional checks to ensure commanded actions are safe and viable. Complimenting an opportunistic sampling algorithm with a planner like CASPER is important, as robots can then reason about the consequences of the selected sampling actions.

In the ASE software we see the framework of: Identify target, describe and rank targets, then collect follow up data with high-cost instrument. ASE can be viewed as a progenitor of the Onboard Autonomous Science Investigation System (OASIS) and the Autonomous Exploration for Gathering Increase Science (AEGIS) system, which can be viewed as a particular instantiation of OASIS.

The OASIS/AEGIS system is currently in operation on Mars and is an excellent exemplar of an opportunistic science system. The successful deployment of AEGIS is vitally important to recognize, as it underscores that autonomous systems can be trusted to make scientific decisions. Even though AEGIS was not given a longer leash until well into the life span of the MER rovers, it still represents a substantial investment and an impressive body of work.

OASIS, as described by Castano et al. (2007), uses three methods to identify science targets. The first is to match templates (as either weights on features or exemplar images) identified by scientists. The second is to use novelty detection in a feature space, which is done using three different modelling methods. The third is to use representative sampling, which ensures it sends examples of stimuli from all the clusters it identifies in a scientist-defined feature space. The representative sampling ensures that rare objects (e.g. uncommon rock types) don't get starved for attention due to an over abundance of other types of objects.

OASIS is a framework which has formed the basis of a number of different science autonomy systems. For example, Gaines et al. (2010) use scientist-defined signatures (read: templates) to identify objects of scientific value. Additionally, they use novelty detection to prioritize data for downlink to Earth. Their data prioritisation scheme comes from the OASIS system, it combines novelty detection, signature analysis(templates), and extracting representative samples. This does let the robot redirect itself to collect more science data. OASIS allows the robot to change its path and to override scheduled tasks, permitting greater freedom for the vehicle than what was deployed on Mars in the form of AEGIS.

Estlin et al. (2012) document AEGIS, which was uploaded to the Mars Exploration Rovers (MER) in 2009. The purpose of AEGIS is to select potential science targets from sensor data

that may warrant follow-up activities. AEGIS uses wide field-of-view (FOV) sensors, like the navigation cameras on MER, to direct the use of narrow FOV sensors, such as a spectrometer or high magnification imager, and thereby collect valuable scientific information.

AEGIS executes at the end of a traverse and uses navigation imagery to find sites of interest. Should any be found the targets are imaged using the narrow FOV multi-spectral panoramic imager onboard MER. Should no targets be identified the system takes no actions. By pre-filtering the data in the navigation imagery and selecting relevant or interesting targets AEGIS saves on bandwidth by only returning potentially interesting data to Earth. Additionally, not transmitting the original navigation imagery for scientists to identify science targets reduces collection time by the time to transmit the original data, have scientists identify targets in the scene, determine best actions to take, and return commands to the rover in the next command cycle.

AEGIS processes imagery in the following pipeline: It identifies targets, extracts features from the image, prioritizes the targets, determines where it needs to point the narrow FOV sensors, the collects data with the narrow FOV sensors. This framework is inherited through OASIS from EO-1. The generality of this algorithm means it is not strictly tied to visual data or to the multi-spectral imager, as noted in (Estlin et al., 2012).

AEGIS identifies targets by using an algorithm called ROCKSTER which identifies rocks in a visible light intensity image. Once rocks are identified they are described by the features of size, reflectance, shape, and location. Scientists have pre-loaded templates that identify feature values of interest, given the local terrain. It remains unclear if the contextual information of "in the local terrain" is encoded in some way other than the fact that the scientist have uploaded templates to be used in the current context. With the rocks prioritized AEGIS identifies the location of the rock(s) of interest and begins collecting information with the narrow FOV multi-spectral imager.

There are considerable computational constraints placed on AEGIS by the computing hardware of the MER rovers. Consequently AEGIS uses edge detection to find rocks. There has been continued work at JPL using the TextureCam (Thompson et al., 2012) algorithm, which employs a computationally efficient random forest classifier to effect a better classifier for a wider range of targets.

AEGIS computes the score, v, of targets using only two features extracted from the data, f_1, f_2, at a time. The target scoring algorithm, given in Eq. (2.1), is defined by three variables, $\alpha_1, \alpha_2, \beta$.

$$v = \alpha_1 f_1 + \beta \alpha_2 f_2 \tag{2.1}$$

α_1 and α_2 determine whether high or low feature values are preferred, and take values of either 1 or -1. β is in the range $[0, 1]$ and determines the relative weighting of the two features. There is the additional ability to exclude targets based on thresholded feature values, i.e. $f_1 \geq C_1, f_2 \geq C_2$, for some thresholds C_1, C_2, in order to remove potentially spurious targets.

AEGIS' scoring function is fairly limited in what it can express. But this should be taken not as a limitation of the AEGIS algorithm, which in principle can use much more complex scoring functions, but of the computing environment on which it is deployed.

This objective function lacks, however, is some grounding in the results of the follow-up sampling action. In defense of AEGIS, radiation tolerant computers may not be able to execute

adaptive algorithms quickly, and using an adaptive algorithm places additional burden through the validation and verification process. Because the communications link between Earth and Mars is a reliable, and the mission cadence does not demand autonomous operations, humans can provide adaptivity on behalf of the rover.

AEGIS represents a substantial step forward in autonomous science because the system not only identifies and classifies sensor data, it makes decisions about what data to collect and has been deployed on another planetary body. The authors of (Estlin et al., 2012) identify several directions that they feel are valuable for improving AEGIS. Chief among them are planning with respect to resource allocation and planning for collecting observations that require direct contact or vehicle motion.

The ProViScout team led by Mark Woods developed an autonomous scientist in the papers (Shaw et al., 2007; Woods et al., 2008, 2009, 2011; Paar et al., 2013). Like the OASIS family of systems, ProViScout is a framework for autonomous science. ProViScout identifies targets to investigate and when it is safe, in the context of mission objectives, deviates from the current plan of action to investigate those targets. It is also capable of conducting plan repair to finish the mission, should opportunistically selected activities cause unforeseen delays.

Paar et al. (2013) review field experiments with the ProViScout where they combine human designed feature detection and anomaly detection. They address, in simulation, the problem of testing with limited sampling budgets and the problem of field testing their science decision making algorithm (SARA), described in detail in (Woods et al., 2008, 2009). In the field, SARA was supplied with a large sampling budget, so the decision making process would not be confounded by cost constraints. In addition to any decisions made by the software, they also had standard science investigation procedures that were conducted at the end of every transect.

The ProViScout system uses pre-defined science goals (Woods et al., 2011). It relies on aerial data, which might not always be available in exploratory settings, i.e. lavatubes or underwater scenes. This system uses a contextual model, which is fairly unique in opportunistic sampling systems. However, the contextual model used in (Paar et al., 2013; Woods et al., 2011) is a pre-specified set of parameters. The contextual model can be updated, but must be done by an outside agent (Woods et al., 2009).

Like OASIS/AEGIS, ProViScout also uses classifiers for science targets that scientists have pre-trained (Woods et al., 2011). The targets identified by SARA are assigned scores by scientists in order to direct the behaviour of the robot (Woods et al., 2009).

Relying on human-assigned weightings assumes that the remote scientists know the relevance of the different classes of objects. On the one hand, this is a simple way for humans to communicate priorities to the robot. On the other hand, it limits the ability of robots to modify the prioritization of different classes of objects, with respect to mission objects, based on observations. Building this adaptivity into the robot could be constructed easily in an information theoretic framework, something acknowledged by Woods *et al.*.

The plan repair component of ProViScout, analogous to OASIS' CASPER, is documented in (Woods et al., 2008). The decision to engage in the opportunistic sampling actions is given to the Timeline Validation Control and Repair (TVCR) planner(Shaw et al., 2007). The sampling opportunity is prioritized by its science value score, and then is given to the planner to insert into the plan. Once the plan has been made consistent with mission goals, the robot executes the new plan, which may contain the newly requested action. However, this decision doesn't account

for the environmental abundance of any other such opportunities, only those which have already been scheduled.

Both the OASIS/AEGIS and ProViScout systems are very similar frameworks for conducting science autonomously. They both have reflexive responses to making the sampling decision: If the object is in the scene, passes their respective prioritization schemes, then sample it. The questions that the systems do not ask is: what is the value of this sampling opportunity, relative to what is available in the environment? This particular question we address in chapter 3.

Having operators identify direct templates or feature weightings to identify objects risks imparting confirmation bias to the robot. Using trained classifiers based on a corpus of data is less likely to impart confirmation bias to the robot. Specifying what objects the scientists want to be identified and not how they want the algorithm to identify them reduces the influence of preconceived notions about the value of different features. However, any trained classifier is limited by the expressiveness of its structure and the data it was trained on, so biases may still reside in the system.

While trained classifiers do let scientists determine which data are interesting, they exclude all stimuli that are not identifiable by those classifiers. This is an acceptable approach if the classifiers are reliable, but given that robots on planetary missions are exploring strange new worlds, that assumption may not hold. An algorithm should to be able to identify objects or clusters in sensor data relevant to the hypotheses the scientists are investigating, but which may not have been in its original training set.

Wagstaff et al. (2013) presents an algorithm, called DEMUD, for identifying potentially interesting targets *in situ* and reporting them to scientists. Wagstaff *et al.* used an SVD-based approach to identify stimuli that are novel, given a corpus of prior observations. The observations are organized into a design matrix – a matrix where every row represents one observation. Such an approach avoids the risk of bringing preconceived biases into object recognition and selection.

More importantly DEMUD identifies the features that make the stimulus interesting. When something is considered uninteresting it is removed from the design matrix, and the process is repeated. In this way DEMUD builds a library of objects that reflect the statistics of the environment. The core of DEMUD is that it evaluates the singular values of the design matrix generated from all previously collected stimuli. DEMUD outperforms competing algorithms designed for novelty detection, managing to find all six of the candidate classes in the UCI glass data set (Frank and Asuncion, 2010).

DEMUD is obviously not the only unsupervised learning algorithm, but it is important because it was designed to operate efficiently, making it amenable to execution on space rated hardware. DEMUD is important for learning things that are new or anomalous in the scene, but we also want to connect these features to a variable of interest, like the work in (Das et al., 2015) which is discussed in Section 2.2.2. Combining these approaches would help improve the ability of robots to recognize and understand anomalous phenomena.

The approaches described above to opportunistic sampling of discrete objects share a common shortcoming. They make decisions to sample an object once it has been identified via some means of classification, and scored as passing a pre-determined threshold on value. The systems do not consider the relative value of sampling these classes of objects, nor do they consider the distribution of these objects in the environment, they simply respond with a sampling action

once conditions have been met. We address these shortcoming in chapter 3 and show that an improvement in behaviour is possible with an algorithm that is adaptive to the environmental condition.

2.2.2 Opportunistic Science in Fields - Prospecting

Distinct from opportunistic sampling of discrete objects is opportunistic sampling in fields, mathematical fields that map the robot's location to a vector or scalar value. With discrete objects there are only certain locations where the robot can collect samples. When sampling in a field, any location the robot occupies is a viable sampling point, although the value of that location may vary.

Thompson et al. (2013) model time series data collected by a robot following a transect. The robot has a sensor that is collecting data at a fixed rate and the only control the robot has is to slow the vehicle to collect more observations. The robot is further constrained by a limited sampling budget. The algorithm considers the degree to which the data has deviated from an assumption of stationarity. The data collected are fit using Gaussian process regression, with a non-stationary kernel. How far the data has deviated from being stationary is determined by how much one of the kernel parameters is increased during the data fitting process. The speed of the vehicle is controlled proportionally to that parameter.

Like the work discussed in (Thompson et al., 2013), Girdhar et al. (2012) modulate the speed of an underwater robot collecting data. The robot collects images with a camera with a fixed frame rate, analyzes the scene, and if the scene is anomalous the vehicle slows down, collecting more images. Their algorithm builds a topic model for the images using a bag of words of ORB features (Rublee et al., 2011), and uses that topic model to score how surprising the current scene is.

Girdhar *et al.* use the surprise of the scene to control the speed of the vehicle. Images are modelled as distributions over the topics in the topic model, and surprise is measured with a symmetric KL divergence between the new image and the closest image previously observed. As surprise increases the speed of the vehicle is reduced proportionally, collecting more data, and vice versa. With each image collected the topic model is updated, so with more observations, the robot builds a better understanding of the world.

Girdhar continues development of the autonomous underwater explorer in (Girdhar et al., 2013a) and (Girdhar and Dudek, 2016). In these two papers the topic modelling is not applied to the whole scene in the navigation cameras of the robot, but to sub-regions of the image. This way different parts of the image can be identified as novel and hence attract the attention of the robot. Instead of simply throttling the vehicle speed the robot follows whatever novel stimuli it encounters.

The robots of Girdhar *et al.* seek novel phenomena, which is valuable for learning about the world. But it would be beneficial to ground the learned topics to other phenomena of interest. Das et al. (2013) build feature descriptions of underwater environments and relates them to the density of life in water samples. In contrast to Girdhar *et al.*, the work of Das *et al.* begins to relate concepts across sensing modalities.

The work in (Das et al., 2013) was later extended in (Das et al., 2015), where they focused on improving repeated surveys through integration of previously collected data. The algorithm in

(Das et al., 2015) is simultaneously attempting to learn the relationship between environmental features and the abundance of plankton discovered in water samples while maximizing the number of samples of plankton collected. Their approach is based around the submodular secretary algorithm.

The submodular secretary problem is a variant on the secretary problem that permits collecting k samples instead of just one (Bateni et al., 2010). It achieves this by dividing up the transect into k segments, and executing a secretary problem algorithm in each one. They demonstrate a reduction in regret in this paper, using both pre-recorded data and from field operations. The basic secretary algorithm is to ignore the first N/e sampling opportunities, and collect the first object with a value greater than or equal to the highest value among the first N/e objects. The secretary problem is very useful when the explorer has non-reusable resources, like the water samplers in (Das et al., 2015).

Secretary algorithms rely on being able to score candidate actions. In (Das et al., 2015) they use the GP-UCB algorithm to score the encountered sampling opportunities. As a baseline they tested an algorithm which scored sampling opportunities with the variance of the predicted reward. While they found that baseline had a higher regret in terms of quantity of plankton observed, it seems to have learned what appears to be a statistically indistinguishable model for predicting plankton abundance from environmental conditions.

Yoerger et al. (2007) uses the Automatic Benthic Explorer (ABE) to localize underwater hydrothermal plumes. They command the robot to follow a lawnmower path through the water, collecting measurements with sensors that measure the density of chemicals ejected from hydrothermal vents. Once the vehicle has reached the end of the transect it plans new site visitations based on whether or not sites meets a revisitation criterion. The scalar-valued revisitation value is not specified in this work, but it is a static threshold (Camilli et al., 2004).

This approach can result in a lot of back tracking on the part of the robot, but that may be acceptable when mission risk posture permits it. However, static thresholds do not account for sensor noise or variability in the underlying process. Further, if the quantity being examined is transient, waiting until the end of the transect to decide to sample it could mean missing valuable observations.

Ferri et al. (2010) address the shortcomings of the work in (Yoerger et al., 2007). Like Yoerger et al. (2007) they complete a lawnmower pattern, identify chemical densities that relate to hydrothermal vents, but if the densities exceed a threshold they then engage in a searching activity to localise the hydrothermal vent.

Ferri et al. (2010) have an adaptive threshold based on how often it has crossed it's original translating path, and how many spiral investigations it has conducted compared to a recommended number of spirals. This approach is an improvement over (Yoerger et al., 2007), but it seems that (Ferri et al., 2010) could benefit further from techniques from secretary problem literature.

The approaches presented in (Girdhar et al., 2012) and (Thompson et al., 2013) are about continuous control of vehicles in response to changes in the observations without and with budget constraints, respectively. Das et al. (2015), Yoerger et al. (2007), and Ferri et al. (2010) deploy discrete actions in response to a vector- (Das et al., 2015) or scalar-valued (Yoerger et al., 2007; Ferri et al., 2010) sensor readings. Das et al. (2015) relate the local observations to a secondary value through the GP-UCB algorithm in a way which is not done in (Girdhar et al., 2012) and that

isn't strictly necessary in (Thompson et al., 2013),(Yoerger et al., 2007), or (Ferri et al., 2010).

The above algorithms lack a notion of confidence in their observations. The work of Girdhar *et al.* chases novelty, Thompson et al. (2013) do not consider the relative likelihood that the underlying distribution has changed from stationary to non-stationary, only that it has been fit by the model. The robot in (Yoerger et al., 2007) finishes the entire transect before identifying the opportunities for sampling, incurring expensive travel times, while potentially missing opportunities to sample time-varying phenomena. While an improvement, the strategy in (Ferri et al., 2010) operates on a static threshold based on instantaneous readings, which leaves the decision making process vulnerable to erroneous readings. Additionally, by using a threshold on readings the vehicle can't react to sub-threshold observations. An approach that acts on belief that the underlying observations have changed could benefit these systems.

The approach in (Das et al., 2015) uses a secretary problem algorithm to collect water samples to simultaneously maximize the volume of plankton observed and learn the relationship between environmental characteristics and plankton abundance. Since their objective is to maximize the observation of plankton in samples their use of the GP-UCB encodes a measure of uncertainty in the prediction, helping it to make safer observations, but it seems to inherently assume that the independent observations – the environmental characteristics – are stable readings. Filtering those values, or building confidence that the underlying distribution has changed could be a beneficial filtering stage for the algorithm.

2.3 Informative Path Planning

Next we consider robots that actively generate paths and actions in the world instead of simply reacting to observations. The robots we review next are determining paths or trajectories that maximize the information gained about their environment. Primarily these algorithms build maps that relate the physical location of the robot to some measurable quantity.

Robots operating without infrastructure often need to conduct simultaneous localization and mapping (SLAM) in order to build accurate georegistered models of data. Stipulating that SLAM is an important aspect of robotics, and an active research topic, this thesis will not go into detailed analysis of SLAM, and interested readers are directed to (Aulinas et al., 2008). What is important to understand is that mapping enables robot scientists 1) to determine spatio-temporal relationships between observations, revealing patterns not otherwise visible, and 2) to determine what regions of the world have not been explored so they may plan their future actions.

Frontier exploration (Yamauchi, 1997) depends heavily on the existence of maps, maps that record not only data where the observer has been, but that are capable of representing where the robot hasn't explored. A robot engaged in frontier exploration drives towards regions of a map that it has not previously seen. The robot may prioritize or triage frontiers for exploration based on different rubrics – traverse cost, distance to a goal, map coverage/certainty, etc. – and then visit the frontiers to collect information (Tao et al., 2007).

Frontier exploration can be viewed as a special case of information gain planning. However, as the space being explored becomes increasingly abstract then identifying frontiers becomes increasingly difficult. That formal information gain planning supersedes frontier exploration is noted in (Visser et al., 2007). Freda et al. (2009) use information gain to map a configuration

space. They use a frontier based exploration method to make sure the space is effectively explored. Thompson et al. (2015b) seek out unexplained spectral signatures in satellite imagery, effectively exploring frontiers of the observed spectral space.

Frontier exploration is driven by an important point: One should sample where one is most uncertain about the state of the world. This is a concept which was outlined in Kristin Smith's work (Smith, 1918), which pioneered the field of design of experiments, and it continues to remain relevant. In many ways frontier exploration is what all science exploration robots are doing. However, instead of simply seeking frontiers in physical space, modern robot scientists explore frontiers in information.

(Thompson, 2008) and (Thompson et al., 2011) present an algorithm that plans paths through overhead multispectral imagery that collects a set of observations that maximize their informativeness measure. Thompson and Wettergreen measure the informativeness of candidate sites by the information that would be gained given all previous observations. The change in entropy is measured with respect to the covariance function of a Gaussian process regression over the satellite imagery. If the entropy function $H(X)$ is defined over a set of observations X, and the rover has to choose a subset of all possible of observations Q, and collect actual observations A, then the rover in (Thompson, 2008) finds the path that reaches a goal location while minimizing the quantity $H(Q) - H(A)$. While data the rover actually encounters when it enters the location corresponding to $a \in A$ is not integrated into the future plans of the rover, as it continues its exploration it can adapt its path to acquire a representative set of data.

This work is expanded in (Thompson et al., 2015b), where the robot seeks out unexplained spectral signatures in satellite imagery. The robot here moves to locations in satellite maps of spectral data to collect surface observations to determine what local mixture of materials explains the spectra observed from orbit. As the robot travels it builds up a database of spectral signatures which it then uses to try and "unmix" the satellite observations.

Pixels are unmixed by taking the database of spectral observations collected thus far, and identifying endmembers. Endmembers act as a linear basis to represent the other as yet unsampled satellite pixels. Those with the highest residual error are considered difficult to explain given the current database of observations. The sites that are difficult to explain are the most interesting ones, and so after every ground observation the robot re-plans and identifies new locations to visit and sample.

Similar to (Thompson et al., 2015b) is the work of Girdhar et al. (2014). They take satellite images, break each pixel down into a bag-of-words of feature descriptors, and learn a topic model for each of the pixels. Then the pixels are scored for topic perplexity, which is a measure of how well an individual pixel is explained by the topic model. The sites that represent the greatest perplexity are then visited and samples are collected.

The perplexity score of Girdhar and Dudek parallels the unmixing procedure of Thompson et al. (2015b). The major difference between the two is that Thompson *et al.* plans a shortest path that maximizes the information gained, while Girdhar and Dudek let the robot follow an unconstrained path. Consequently their trajectories appear rather more idiosyncratic, but there is no reason their approach could not be used with a more conservative planner.

Building further on the work presented in (Thompson et al., 2015b), Candela et al. (2017) changes the problem somewhat by explicitly accounting for a hypothesis about the geological units present in remote sensor data. The global map is divided up into regions, each of which

represent a geological unit. The robot then designs trajectories that maximize the information gained in the probability distribution over the geological unit assigned to a region, given the observations made within those regions. This work is important for carrying with it a hypothesis about the environment it is exploring, and planning in order to shore up support in that hypothesis. However, it does only consider one hypothesis (the region divisions) at a time.

Hollinger et al. (2013) plan informative paths using a Gaussian Process with a non-stationary kernel function. This approach permits modelling variable uncertainty in the object it is tracking, which is important for representing non-stationary models. In a similar vein, (Hollinger and Sukhatme, 2014) uses a sampling based approach to determine trajectories for an unmanned water vehicle that maximizes information quality - a generic concept standing in for variance reduction or information gain.

Work by the Williams' group at the University of Sydney, documented in (Bender et al., 2010), uses the results of *in situ* measurements to direct the actions of an AUV to adaptively map the spatial distribution of an underwater coral reef. Bender et al. (2010) represent the spatial distribution of the coral reef with a Gaussian process classifier. The robot's sensory data was classified into one of two classes (reef or sand), and the robot investigated ambiguous regions of the map, the most interesting locations being where $p(reef) = p(sand) = 0.5$. By investigating those sites they reduce the ambiguity in the map. They compare the behaviour of their algorithm to a robot following a lawnmower pattern, and find their proposed algorithm improves the quality of the map.

One downside of the approach detailed in (Bender et al., 2010) is that the Gaussian process classification does not admit entropy calculations without a computationally intensive Monte Carlo estimation. The computational demand makes using approaches like mutual information or maximum entropy sampling challenging, especially for limited computing hardware. The reward function they do use, finding ambiguous points, when scaled up to arbitrary numbers of classes could be considered similar to the perplexity measure of interestingness used in (Girdhar et al., 2014).

The work by Charrow et al. (2015) also plans an information gathering path. They manage to extract significant speed up in planning time by virtue of using the Cauchy-Schwartz Quadratic Mutual Information metric (Principe, 2010) and by discretizing the world being explored into voxels. Through the use of these methods their informative path planner is found to be much faster than with the standard Shannon Information Gain, as used in most other works in this chapter, but still achieving the same objectives. Building upon Charrow *et al.*, the work of Tabib et al. (2016) hinges on the notion that given an observed object, the information gained from sensors observing that object should be independent, and as such, additive. This means that the robot can efficiently plan for multiple passive sensors simultaneously. This is, to the best of the author's knowledge, the first planning algorithm that address the multi-modal informative path planning in a principled way.

Also eschewing the more standard Shannon information gain, Miller et al. (2016) use the expected value of the Fisher information to determine points of interest. Like mutual information (Lindley, 1956), Fisher information is used as a score to select the most informative experiments. Their path planner produces smooth paths that maximize the number of high information value observations. Fisher information and the mutual information are intimately related through the curvature of mutual information, but they are not identical quantities. However, a rigorous com-

parison of the behaviour of robots maximizing mutual information and those maximizing Fisher information does not exist in the literature, to the best of the author's knowledge. One would conjecture that they would produce similar behaviour, but without such evidence it is difficult to select one reward function over the other and this should be studied further.

Similarly, (Schwager et al., 2017) uses the *gradient* of mutual information as a reward function. However, this was done not because of a connection between mutual information and Fisher information. Schwager et al. (2017) examine teams of exploring robots, and use the failures of team members as a way of encoding hazards into the map of interesting phenomena. Where robots fail, their sensors report no information, effectively zeroing out the gradient of information gain, causing other team members, which are following the gradient of information gain towards maxima, to avoid hazards encountered by less fortunate robots.

Das et al. (2013) take an interesting approach to science navigation by considering a composition of functions. They learn one function which maps locations in space to characteristics the environment at those locations, and a second function which maps environmental characteristics to the quantity of interest, namely abundance of plankton in water samples. Like the work of Thompson *et al.*, they use a Gaussian process to learn spatial model of environmental characteristics, but they also use a second Gaussian process regression that maps environmental features to plankton abundance.

The environmental characteristics are learned from features that describe data from cameras and other sensors, such as temperature and chemical density sensors. The robot collects water samples that are then processed between surveys to train the second Gaussian process.

After each sample is processed the robot plans new trajectories that maximize the chance of discovering life. They use the unscented transform to propagate uncertainty in feature predictions through to uncertainty in life abundance predictions. This can then be used to estimate the distribution over the abundance of life over the entire map, which is then fed into a maximum information gain planner. While this particular implementation of their project does have the disadvantage of not being able to update the map of life abundance *in situ*, the approach could be used unmodified if on-board processing of water samples is possible.

The interesting part of Das *et al.*'s design is that, assuming the features are useful for predicting other scientific quantities, this approach could readily admit multi-modal sensing for scientific inquiry, simply by maintaining other Gaussian processes mapping the feature space to the observations of other instruments.

These algorithms were mainly focused on planning efficient trajectories that also increased informativeness, by some measure of informativeness. Almost all of them rely on the information gain being submodular, meaning that greedy algorithms can approximate the best algorithms within some acceptable margin of error. Hollinger and Sukhatme (2014) is the notable exception this approach. There are four candidate functions for information gain used in the work discussed above: Variance reduction, Shannon Mutual Information, Cauchy-Schwartz Quadratic Mutual Information, and Fisher Information. Shannon and Cauchy-Schwartz mutual information are intimately linked concepts, but it seems that Cauchy-Schwartz is faster to compute, according to Charrow et al. (2015), and therefore favourable.

Variance reduction is an approach that has been used since at least the beginning of Design of Experiments literature (Smith, 1918), and is the concept underlying the class of Upper Confidence Bound algorithms (Lai and Robbins, 1985) used in the Multi-Armed Bandit literature.

Further, variance has a reciprocal relationship with Fisher information, so reducing variance is equivalent to maximizing Fisher information. Connecting these concepts even further, Fisher information is proportional to the second derivative (or curvature) of the mutual information between two distributions (Gourieroux and Monfort, 1995). However, Fisher information does have a strong dependence on parameterization, which could make implementation of more general algorithms challenging.

There are a number of other variance minimizing criteria used in experiment design, above and beyond those discussed here. The D-Optimality criterion is a means for selecting informative experiments (Croarkin et al., 2002). If every experiment is represented in the row of a matrix, X, then the best set of experiments, given a fixed budget, is the collection of experiments which maximizes the determinant, $|X^T X|$. This approach has the advantage of only needing to know the parameters of the experiments being conducted.

Also worth discussing is the notion of maximum entropy sampling. With each experiment represented as a random variable X_i, then the best collection of experiments to conduct A is the set which maximizes the entropy of the joint distribution, $f_A(X_1, \ldots, X_N)$, where $X_1, \ldots, X_N \in A$. This is also equivalent to maximizing the determinant of the covariance matrix of the selected experiments, A. All of the above criteria can all be considered as objectives when designing informative actions.

Smith (2007) considered navigating a grid world in order to deploy an ultraviolet fluorometer in order to detect microbial colonization of materials. This work demonstrates that doing onboard data analysis can improve the decision making processes of robot explorers. The problem was solved as a POMDP, which let the robot reason about the effects of its actions on its state of knowledge while it was exploring. This approach is rare in that it reasons about the actual observations it may encounter while navigating, instead of just making sure it has observed a reasonable sampling of the surface. POMDPs are, unfortunately, computationally intensive, and do not scale well as the action space increases.

Choudhury et al. (2017) attempt to overcome the computational complexity involved in using a POMDP model by using a reinforcement learning strategy. The objective is to produce a function which can predict the relative value of candidate sampling in order to pick the best one.

In (Choudhury et al., 2017), different exploration tasks are simulated where a robot is tasked with maximizing the information gained about a hidden world map. Here the algorithm can use an oracle with access to true world knowledge to select actions during training. The current state of the robot's world knowledge and the oracle's decision is used to train a heuristic that approximates the optimal decisions without the computational overhead.

There are limitations in that this algorithm is only as good as the data upon which it was trained. Nevertheless, a reinforcement learning strategy could be readily adapted to online performance, and it is a reasonable approach to approximating POMDP based solutions.

Arora et al. (2018) detail an algorithm for a robot that is seeking to locate a subsurface distribution of water. The robot has to plan not only trajectories that inform the distribution of subsurface water, but it also has to determine which of a set of instruments it will employ to collect observations. These instruments have different costs, and the observations may change the trajectory that may be the most informative. Their planning problem is complicated by not obeying sublinearity. They use approximate planning measures to deal with the complexity of the planning problem, and the use of an on-board robot-learned hypothesis about the relation-

ship between the navigation sensors and the instruments used to determine the abundance of subsurface water.

The work of Choudhury et al. (2017) and Arora et al. (2018) both attempt to mitigate computational complexity. The first through approximating the reward function, and the second through approximating the solution. The computational complexity of these problems can be crippling, but mitigation strategies like those discussed can to mitigate computational demands.

All the information gathering algorithms presented above are attempting to plan trajectories that are by some measure the most informative about the environment they are exploring. With the exceptions of Candela et al. (2017) and Das et al. (2013), these algorithms do not consider hypotheses about the phenomena they are exploring, only efficient ways to explore the environment. Even (Candela et al., 2017) and (Das et al., 2013) consider only one hypothesis at a time. In the next section we discuss different approaches to generating hypotheses, an important component of the scientific process, but if we are to combine informative path planning and hypothesis generation we will need a class of planners which are aware of and capable of reasoning about multiple hypotheses.

2.4 Hypothesis Generation

A great deal of excellent work in autonomous science research goes into ensuring that useful information is collected in a principled way, as per the work reviewed above, what is often lacking is the question of what is done with the data after the collection. In flight missions, scientists would use these data to generate hypotheses.

Hypotheses are an important part of conducting science, but generating hypotheses is not an actively studied part of autonomous field science. While this thesis does not investigate the automatic generation of hypotheses, this is an interesting area of research that is relevant to the work conducted in chapter 5. Further, since chapter 5 is about designing experiments to test hypotheses, it is worthwhile being aware of mechanisms for generating hypotheses.

Levin introduced the notion of searching for Turing machines, which are functions that map input bit strings to output bit strings (Levin, 1973, 1984). Given a fixed language for describing Turing machines and a problem to be solved, the simplest machine – the one described by the shortest string – could be discovered by first searching through all the shorter machines. Each one of the candidate Turing machines can be viewed as hypotheses that need to be tested to see if they satisfy the relationship under investigation.

The search space can be quite large, but the algorithm that successfully completes the task will eventually be found. Levin's universal search procedure only considers functions that map from exact inputs to exact outputs. To accept approximately correct solutions, one could simply place tolerable level of error on the output and stop searching once that tolerance has been met.

Langley et al. (1987) presented a series of algorithms which discover empirical laws relating different datasets, as well as semantic rules for relating a dataset that has been collected by some other process. Their work covers four algorithms, BACON, GLAUBER, STAHL, and DALTON. These algorithms deal with using data for finding quantitative laws, qualitative laws, inferring components of substances, and formulating structural models. In many respects algorithms presented in that book are different types of searches through a hypothesis space, and is

an important precursor work to science autonomy algorithms presented below.

Levin's search was further developed by Schmidhuber in (Schmidhuber, 1995) and (Wiering and Schmidhuber, 1996) for (single layer) neural networks and POMDPs, respectively. Again, the process is to start with simple attempts at solutions to the problem and progressively make them more complex until a satisfactory solution is found. Here the algorithm depends on a vocabulary of symbols that can be used to produce the hypothesized solutions. Each of the solutions can then be tested for fitness – how accurately they solve the problem – and then either discarded or retained accordingly.

The robots Adam and Eve were developed at Aberystwyth University to automatically conduct laboratory experiments (King et al., 2004, 2009a; Qi et al., 2010). These robots automate much of the tedious work in laboratories, and help reduce variability in production. Adam generated functional genomics hypotheses about yeast, and tested the hypotheses in a laboratory (King et al., 2009b). Adam conducted this work by exhaustively testing different hypotheses it generated about the yeast.

Eve, on the other hand, methodically screens candidate drugs, but is able to stop the exhaustive search of the space of hypothesized drugs in order to generate quantitative structure-activity relationship (QSAR) models (Qi et al., 2010). QSAR modelling is a mechanism for determining the predictive power of the drug components to the potency of the drug. With a completed QSAR the robot can then predict drugs with similar structures which may be effective. This permits the robot to design new hypotheses about drugs which may be effective in treating different diseases. This gives the robot the freedom to follow up on promising drugs instead of having to wait to finish testing all possible drugs in the space of drugs it is testing (Sparkes et al., 2010).

Schmidt and Lipson (2009) developed an algorithm for deriving natural laws based around symbolic regression (Cramer, 1985) which has since been developed into the product Eurequa[2]. This approach relies on being able to represent equations and relationships as parse trees in a grammar over possible equations. They then apply a genetic algorithm to search the space of parse trees in order to determine a hypothesis which best fits the collected data.

Because the genetic algorithm doesn't progressively through the space of possible hypotheses in the same way that a Levin search does there is a need to control for over fitting. The approach used by Eureqa is to score each hypothesis based on its fitness to the data set under inquiry combined with a penalty term for the computational complexity of that hypothesis. In (Ly and Lipson, 2012) they use the Akaike information criterion (AIC) which is defined in Equation 2.2.

$$\text{AIC}(h, D) = 2|h| - 2\ln(\mathbb{P}(D|h)) \tag{2.2}$$

Where $|h|$ is the algorithmic complexity of the hypothesis being scored and $\ln(\mathbb{P}(D|h))$ is the log likelihood of the data, D, given the hypothesis, h. Smaller values of AIC are preferred. The search algorithm that describes this system could be modelled as in Algorithm 2.1, where the generate_hypotheses method is the symbolic regression algorithm described in (Ly and Lipson, 2012).

[2] www.nutonian.com

Algorithm 2.1 A skeleton of the hypothesis search algorithm used in Eurequa. hypothesis_generation is their symbolic regression function, and the score function is the Akaike information criterion.

function SEARCH_HYPOTHESES(grammar,D)
  $H \leftarrow \emptyset$
  $scores \leftarrow \emptyset$
  while not terminated **do**
    $H' \leftarrow$ generate_hypotheses$(H, scores, grammar)$
    $scores \leftarrow$ score$(h, D) \; \forall \; h \in H'$
    $H \leftarrow H'$
  end while
  return $\underset{h \in H}{\operatorname{argmax}}$ score(h, D)
end function

The system that supports Eureqa assumes that the data to be fit has already been collected. It then uses the genetic algorithm to search through the space of possible hypotheses which are expressed in a grammar given by a vocabulary and set of operations specified by the user. The system has managed to re-discover physical laws from data that are in a human-interpretable format, as opposed to, e.g. a neural network.

Genetic algorithms have the advantage of biasing the search space towards members of the space that have already been successful, and hence eliminate a number of hypotheses that are unlikely to be productive. However, there is always the risk that the "best" hypothesis will never be evaluated, although this risk reduces the more generations are bred in the learning process.

A similar approach to searching the hypothesis space is embodied in the Automatic Statistician[3]. This is a work that has been developed in three papers, (Grosse et al., 2012; Duvenaud et al., 2013; Lloyd et al., 2014). The program is designed to study a dataset and develop descriptions of the data in human-interpretable format. With the automatic statistician the hypotheses are the symbolic representations of the kernel functions, and it attempts to find one that best explains the data.

The algorithm works by generating covariance functions for Gaussian process regression and classification models and then translating those into natural language statements about the data. The method for generating the kernel functions is to start with a simple function and progressively make it more complex, as in Levin search, by iteratively applying the rules of the grammar to generate new and more complex functions.

The automatic statistician assumes that all data are available initially and exhaustively searches through the space of possible hypotheses, to a fixed depth of the search tree, to find the best one. They score their hypotheses using the Bayesian Information Criterion (BIC) (Schwarz et al., 1978), given in Equation 2.3, which is very similar to the AIC used in Eureqa, however it more sharply penalizes the complexity of the model. Where h is the hypothesis, D the dataset, $|h|$ is the complexity of the hypothesis, and $|D|$ the number of points in the dataset.

$$BIC(h, D) = |h| \ln(|D|) - 2 \ln(\mathbb{P}(D|h)) \tag{2.3}$$

[3] `https://www.automaticstatistician.com`

As with Eureqa there isn't an explicit stopping criteria, although the authors of the automatic statistician do put a limit on how deep in the tree of possible kernel functions it is permitted to search. With arbitrary complexity a model can be made to fit data arbitrarily well – over fitting – so there needs to be a way to trade off the fitness of the hypothesis with the complexity of the hypothesis. Where the Eureqa uses the AIC, the Automatic Statistician uses the BIC. We sketch the automatic statistician's algorithm in Algorithm 2.2.

Algorithm 2.2 A skeleton of the hypothesis search algorithm used in the automatic statistician. The generate_hypothesis function is a Levin search through the space of kernel functions, and the score function is the Bayesian Information Criterion

function SEARCH_HYPOTHESES(grammar,vocabulary,verbs,max depth,D)
  $H \leftarrow \emptyset$
  while depth not reached **do**
    $H \leftarrow$ grammar_expand$(H, D, vocabulary, verbs, depth,)$
  end while
  return $\underset{h \in H}{\operatorname{argmax}}\, \text{score}(h, D)$
end function

Where Eureqa uses a guided search, the Automatic Statistician uses a depth-bound Levin search through a grammar of kernel functions, but ultimately they both attempt to find a hypothesis in the space of hypotheses which best fits a dataset, according to a fitness function that trades off performance with algorithmic complexity. Neither the automatic statistician nor Eureqa have a mechanism for recognizing that a new hypothesis is required, they simply search until they have found the best one. Nor do they determine where to collect samples to further improve their understanding of the hypotheses under consideration. In Chapter 5 we address both these shortcomings.

In the closing remarks of (Schmidhuber, 1995) the authors make the observation that true generalization is impossible. That any one model fit to data only really speaks about the data collected and can't necessarily be trusted to predict on data that has not been previously observed. In many respects this reflects Hume's problem of induction (Vickers, 2016) - any hypothesis cannot be truly trusted to predict outputs from inputs it has not been developed from. The logical problem of induction is what drives Popper's falsification criterion and what also drives scientists to collect more data in order to test out their hypotheses, and when they find them wanting, generate new ones.

An ad hoc hypothesis is a modification to a favourite hypothesis to save it from being falsified in the face of new data. While the phrase "ad hoc hypotheses" is often considered a disparaging remark, it is important to remember that all hypotheses are ad hoc responses to new data that could not previously be explained, what separates "good" hypotheses from "bad" hypotheses is that the modifications do not reduce the predictive ability of the hypotheses and that those hypotheses are not later themselves falsified.

It is the need to continually falsify hypotheses that drives the notion of active learning. Agents need to collect more data in order to determine which of a set of possible hypotheses is the "best" hypothesis. Here best will be taken to mean maximizing or minimizing some objective function

over the hypotheses and the data to be explained by them. Unless a very comprehensive set of hypotheses are provided from the outset, an agent may well have to generate new hypotheses to explain the data they collect in pursuit of their scheme of falsification.

What all these approaches have in common is that they are using a grammar to generate a (possibly infinite) space of potential hypotheses, and they operate on a fixed dataset. They search the hypothesis space, either in a breadth first search, like (Levin, 1973), or in the case of (Schmidhuber, 1995), using a genetic algorithm to perform the search like the Eureqa project, or exhaustively with an imposed limit like the automatic statistician. These hypothesis generation algorithms do not consider, once a set of credible hypotheses have been identified, how to expand the dataset in order to determine which of those hypotheses are most credible.

2.5 Summary

In opportunistic sampling of discrete objects we have seen that the state of the art algorithms ignore the relative availability of different classes of objects. While they do evaluate taking opportunistic sampling actions with respect to the effect on the overall mission, they do not consider the likelihood of encountering more valuable sampling opportunities. We address that shortcoming in Chapter 3.

With prospecting algorithms we see that they don't consider with what confidence a triggering event has been observed. We address that shortcoming in Chapter 4, in order to demonstrate an improved ability to deploy secondary sampling actions.

Finally, when it comes to planning for information gathering we see two things. The first is that the vast majority of informative planning algorithms consider at most one hypothesis when planning paths and sampling actions for autonomous robots. The second thing we see is that algorithms which generate hypotheses only consider fixed datasets at the moment of generation.

What is not answered by these bodies of work is how to plan actions that are informative, in order to determine the most accurate hypothesis. In Chapter 5 we present an algorithm that plans informative actions in order to determine which of a set of hypotheses most accurately describes collected data. This kind of planning will be a fundamental component of robots which are simultaneously attempting to generate and choose between multiple competing hypotheses.

Chapter 3

Opportunistic Sampling of Discrete Objects (Foraging)

While exploring unknown environments robots will necessarily encounter unanticipated phenomena. To handle this situation robots require some mechanism that allows them to react opportunistically to those phenomena. In this chapter we consider robots that are encountering discrete objects and that are interested in learning the underlying distributions of observations from sampling these discrete objects.

The object classes and the underlying distributions could represent many different things. For example, it could be types of drugs and expected outcomes, or, continuing a medical theme, it could be patients representing clusters of symptoms and recommended treatment. It could be classifiers acting on a data stream, and the underlying distribution could be its accuracy, as verified by some oracle.

This work was developed in the context of the Life in the Atacama Desert project (Wettergreen et al., 2005). The algorithm could act as a tool to bring some human adaptivity on-board the vehicle. In the AEGIS system (Estlin et al., 2012), humans decide whether or not to activate the autonomous science system.

Our proposed foraging algorithm can provide some of that human decision making on-board a robot for when the communications link to a remote science team is limited. However, the tool could also be used to suggest sampling actions to human operators during a high tempo missions that has high communications throughput.

To keep the project grounded in planetary exploration, we consider different classes of geologic materials, rocks in particular, as the discrete objects, and the underlying distribution would be over the colonization by microbes of that class of rocks. We assume that the robot can identify the different objects with some inexpensive proxy sensor, but it can only collect observations informing microbial colonization by using a more expensive sensor. The overall goal of the robot is to estimate the distributions underlying the different classes of objects with minimal error.

We further assume that the robot has no global information. That is, the robot does not know how many objects it will encounter, nor what their classes will be. We assume that the robot is following some pre-determined trajectory, such as a lawn mower pattern as in Figure 3.1.

The classes of objects that the robot encounters are drawn independently from a distribution representing the prevalence of different classes of objects in that environment. In each encounter

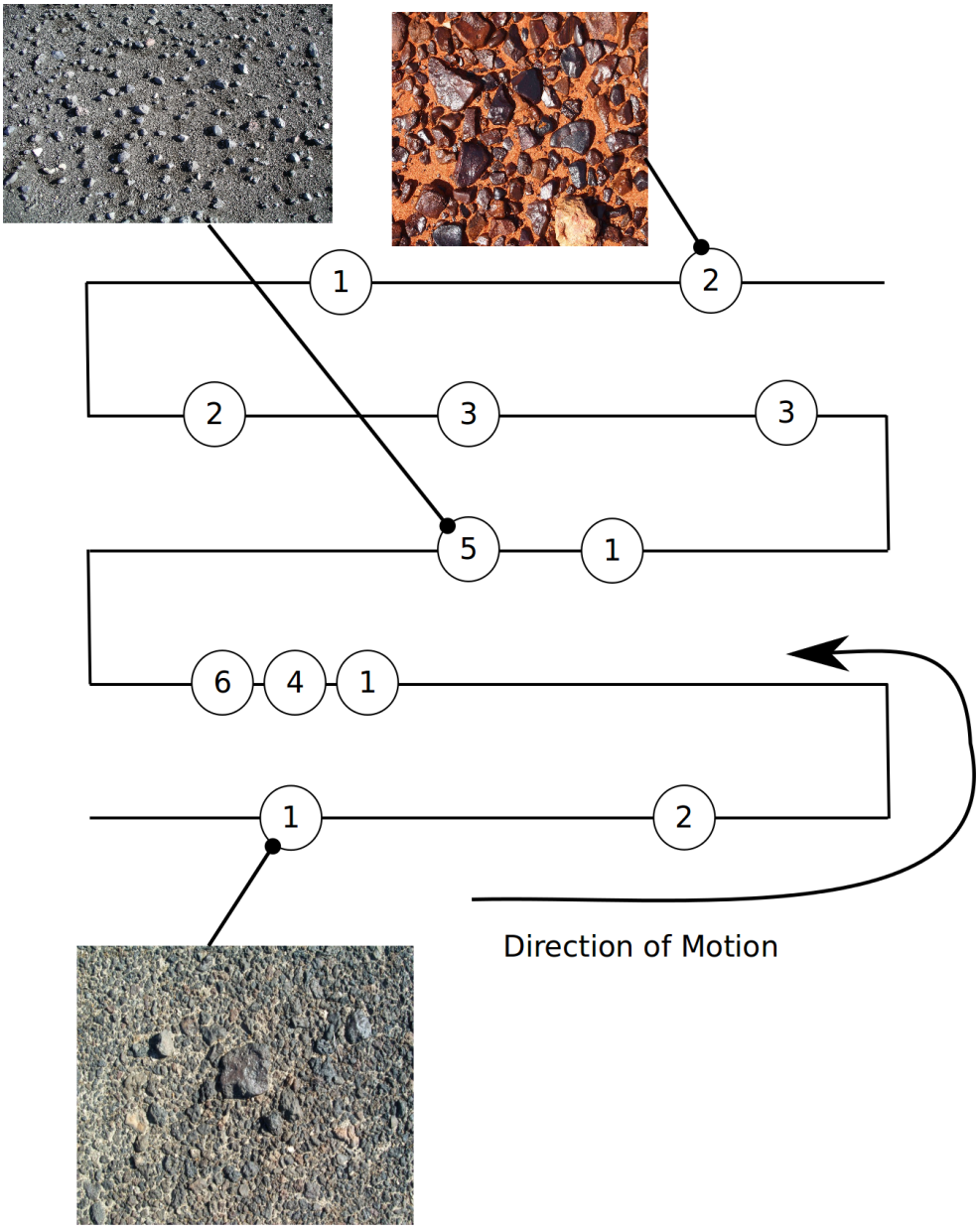


Figure 3.1: A cartoon of a path explored by a rover. The images represent different classes of desert pavements that may encountered by a rover as it follows a pre-determined path.

with an object the robot can either choose to sample or continue exploring along the trajectory. Thus, the problem can be thought of as a stream of sensing opportunities, providing varying reward, and each requiring a decision to either sample or move on.

The state of the art algorithms in robotics literature for opportunistic sampling represent a reflex-like approach to opportunistic sampling. They generally either match a template or they respond to the novelty of the class of object. Seeking novelty is beneficial from both an information theoretic and a logistical approach. If class of object is rare, either it will provide greater information gain, because the explorer is unlikely to have collected many samples from it, or logistically because there may not be many remaining opportunities to sample that class of object, so passing up the object doesn't make sense.

A template matching approach, however, is more problematic. Note that we include pre-trained classifiers in the phrase "template matching". On the one hand classification schemes are an efficient way for scientists to communicate their preferences for sampling to robots. On the other hand, if these templates are not properly vetted then template matching is a means by which confirmation bias can be encoded into the robot. That, in turn, can reduce the quality of the science conducted.

Both novelty seeking and template matching ignore the results of their sampling actions. That is to say, how much information was gained by the most recent sample of an object of a given class. There are standard tools in the design of experiments literature which do address this shortcoming. However, those algorithms make assumptions which do not hold in field operations. This will be discussed in section 3.1.

Another consideration is that the number of samples that a planetary robot might carry. The Phoenix lander carried four copies of the single-use Wet Chemistry Laboratory (WCL) instruments (Hecht et al., 2009). Zoë, in the second Life in the Atacama Desert project had 20 sample receptacles it could use to collect soil samples (Paulsen et al., 2013). The upcoming Resource Prospector mission will likely be only collecting on the order of tens of samples with its drill (Andrews et al., 2014). In planetary exploration missions sampling resources can be very limited, and as such one wants to be effective with small total sampling budgets.

In this chapter we introduce an algorithm to address the shortcomings of the novelty-seeking and template matching algorithms. The method we propose makes two major modifications to the state of the art. First, the objects are valued by the expected information gained by sampling the object. Second, we compare the anticipated information gain from the current object to the expected information gain for the next encountered object. With these improvements our algorithm is able to make the decision to either sample the object that is immediately available to the robot, or to continue along the trajectory in the hopes of finding a more informative opportunity for sampling.

The proposed algorithm draws on techniques from optimal foraging theory and sequential experiment selection. Its use is motivated by observations of human and animal behavior, exemplified by geologists making decisions about investigating local phenomena without prior access to detailed maps. They are able to effectively choose between either sampling materials in front of them or exploring for more profitable sampling locations. These decisions may not be globally optimal, but they offer robots a mechanism for making foraging decisions (either to engage with the environment or to continue exploring) in a computationally tractable way.

The algorithm is then extended to address another common situation in scientific exploration,

namely environmental variability. As the robot traverses the environment, it may easily pass between regions where environmental conditions differ, which affects the distribution underlying the classes of objects. In the Atacama desert 100% of photosynthesis-promoting translucent rocks are colonized by microbes in semi-arid regions, but less than 50% of such rocks in arid regions, and less than 1% in the hyper arid core (Warren-Rhodes et al., 2007, 2006). If one can detect changes in the observed quantity, then one might infer changes in unobserved quantities in the environment, and trigger other investigative actions. Reacting to such changes is also relevant for information gathering purposes, so sampling decisions will not be based on historically observed but now inaccurate class information. The proposed extension incorporates an additional statistical test to detect a change in class distribution to notify operators, separate data segments, and reset the observation history that might otherwise misinform upcoming sampling decisions.

To demonstrate the value of the algorithm we conducted a number of simulation experiments. First we gave the robot a problem with three classes of objects to investigate. Next we explored the effect of the arrival distribution on the behaviour of the foraging algorithm relative to baseline algorithms, and the effect of the underlying distribution on the algorithms.

In all experiments we varied the exploration and sampling costs for the robot and non-uniform arrival probabilities. In the first experiment we used a uniform arrival distribution with a small number of class of objects. In experiments 2,3, and 4 we considered two arrival distributions which are heavily skewed in favour of one class of object over the others. We also varied the underlying distribution of the most common algorithm (experiment 3) in order to determine the effect of the algorithms' prior belief in colonization on their performance.

In experiment 5 we tested a change detection extension to the algorithm, by changing the distribution underlying the classes of objects during operations. Here we selected one sampling and exploration cost, which were small relative to the overall sampling budget, in order to see how detecting changes in the underlying distribution can improve estimates of class distribution parameters.

The remainder of this chapter begins with a brief survey of the relevant literature. Next, a detailed comparison of the proposed foraging algorithm and one based upon existing principles from the design of experiments literature. Finally, discussion of experimental results from a simulated exploration scenario indicates that under limitations on sample collection and overall mission time, the foraging algorithm presents a statistically significant improvement for a realistic range of sampling costs.

3.1 Prior Work

Automating experiment design and selection is not without precedent. Kristine Smith started the field of optimal experiment design in 1918 (Smith, 1918). Recently robots have been employed to conduct scientific inquiry autonomously (Wagner et al., 2001; Castano et al., 2007; King et al., 2004). Current robot scientists' reliance on global information makes operating in truly unknown environments challenging. Additionally, previous approaches in sequential decision making from statistics do not necessarily reflect the settings that autonomous robots encounter in the real world. The particular approaches in question are formulations of sequential experiment selection problems, which are the Secretary Problem and the Multi-armed Bandit (MAB).

3.1.1 The Secretary Problem

The secretary problem asks a decision maker to select the best candidate from sequentially presented candidates where it is not possible to return to rejected candidates. In the original setting, there is only one position for the candidate to fill, (Ferguson, 1989) and the optimal strategy is to reject the first $\frac{N}{e}$ candidates and then accept the first candidate who is ranked better than any of the previously seen candidates. Further, the decision maker was able to objectively score the candidates without cost. In our setting, candidate value is unknown before sampling, and sampling a class incurs a sampling cost.

There have been many variations on this problem including selecting multiple candidates (Vanderbei, 1980), or when the total number of candidates is random (Presman and Sonin, 1972), for more variations (Ferguson, 1989) is an excellent source. The Odds algorithm has been designed as an optimal solution to the secretary problem, but it requires knowledge of how many opportunities there are to collect samples (Bruss et al., 2000). More recently, the submodular secretary problem has been developed to handle selection of k candidates, as opposed to just one (Bateni et al., 2010). It simply divides the transect into k contiguous regions and runs the standard secretary algorithm on each segment.

What distinguishes the secretary problem from the particular science autonomy problem we propose is that we do not know the value of a candidate, or class, when we encounter it. Objects must be sampled to learn their true value. Additionally repeatedly sampling the same class decreases the value to the decision maker, whereas in the secretary problem the encountered value of the object is the value of the object. Since most of the secretary problems rely on previous observations of candidate values, our expected decrease in candidate value would not be compatible.

3.1.2 Multi-armed bandits

Sequential experiment selection, a type of active learning, is addressed in the multi-armed bandit (MAB) literature. This was introduced by Robbins (1952) as a means of sequentially selecting which experiments to conduct with a limited budget. In Robbins' work, selecting experiments is modelled on determining the payouts of one-armed bandit machines – each machine representing a different experiment. The player has a fixed sampling budget and has to sequentially choose which machine to play, trading off exploiting expected rewards from well-studied arms against exploring different arms, learning more accurately the payouts of those arms.

Lai and Robbins (1985) use a value function in which uncertainty in arm rewards makes an arm more interesting. Recently decision rules like Thompson sampling (Thompson, 1933) and Bayesian Optimal Control (Ortega and Braun, 2010) have gained popularity. Other techniques addressing the exploration/exploitation problem use uncertainty as a reward metric (Burnetas and Katehakis, 1997; Auer, 2003; Balcan et al., 2006). In our setting, because the agent only needs to learn the distribution and not use it for anything, uncertainty is the only necessary reward.

Balcan et al. (2006) presents a method for learning classifiers by requesting samples from the input space with the greatest classification error. Classification error and uncertainty in function value are fungible quantities in this case. An analogy can be drawn between the classifiers used in (Balcan et al., 2006) and the bandit arms used by Auer and Ortner (2010).

Several factors distinguish the MAB setting from the problem explored in this chapter. In MAB, the agent has access to any arm (analogous to a class in our setting) it chooses at any given time. The agent in our setting does not get to choose which of the classes it can investigate. Any previously seen classes are no longer available, and new classes arrive per a random model. Additionally, the standard MAB setting does not have switching costs, although there are some formulations which do include such costs (Jun, 2004). In our setting, there is a cost incurred with every choice to continue exploring.

3.1.3 Optimal Foraging

Foraging is the problem encountered by animals seeking to maximize energy intake when operating in unknown environments. The central question of the problem is whether it is more valuable to continue extracting resources from the current location than it is to seek out resources in new locations. Charnov (1976) introduced a technique for dealing with "patchy" environments, in which there are distinct regions that contain different classes of resources. The forager can extract value from these patches, with diminishing returns (modeling resources consumed), or choose to continue to wander randomly in the hopes of encountering more valuable locations.

The optimal time to leave a patch, according to Charnov's Marginal Value Theorem, is when the expected return from continuing to sample a patch is less than the expected return from searching the environment. In this formulation, the expected return from both the current patch and the environment are offset by the cost of extracting resources in this patch and the energy spent seeking a new patch.

Pirolli and Card (1999) studied researchers attempting to acquire information. They modelled the rate of information gain and had their agent decide to leave a patch when the rate of information gain was lower than that of the environment. What differentiates their setting from ours is that their decision maker can choose which patch to sample, yet our exploring agent cannot.

Kolling et al. (2012) studied humans engaged in a gambling task in which players have to consider the option they have before them and the opportunities the environment provides. Subjects were repeatedly presented with a choice of playing a gambling game or being randomly presented with a different game. Each game was a Bernoulli trial with some unknown probability of success. Kolling *et al.* identify possible neural substrates for foraging decisions in humans. The behaviour was near optimal, with some skewing of probabilities near 0 or 1.

3.1.4 Opportunistic Science

In chapter 2 we discussed the NOMAD(Wagner et al., 2001), OASIS/AEGIS (Castano et al., 2007; Estlin et al., 2012), and ProViScout (Paar et al., 2013) systems for deploying samples opportunistically. These algorithms use classifiers to identify objects in the scene, which are then prioritized via value functions specified by remote scientists. Any identified objects which meet a criterion for being sampled are then passed on to their scheduler to be integrated into the mission plan. In this chapter we demonstrate that recognizing the availability of the different classes of objects in the environment, can improve estimates of the underlying distributions.

Das et al. (2015) present a method for deploying samples as robots explore along a transect, as does Girdhar et al. (2013b) and Thompson et al. (2013). However, the problem settings in these papers have robots operating in either scalar or vector fields, as opposed to sampling discrete objects. These works will be discussed in more detail in Chapter 4.

Other autonomous science work demonstrate that information gain, either in terms of Shannon(Thompson, 2008), Cauchy-Schwartz Quadratic(Tabib et al., 2016), or Fisher(Miller et al., 2016), is a useful quantity to evaluate the relative worth of different sampling actions. In this chapter we used the Shannon definition of information gain, however, in future work we would recommend using the Cauchy-Schwartz Quadratic mutual information, if for no other reason than the decrease computation time noted by Charrow et al. (2015).

Sun et al. (2011) proposed a method for maximizing the expected information gain of a sequence of actions in a Markovian world. This work can be viewed as a more general formulation of the solution presented in this work. It forecasts out the actions of the robot to either a fixed or infinite horizon possible sampling opportunities. Our work can be viewed as employing the model developed by Sun et al. (2011) with a single time-step horizon, something more amenable to operation in restricted computing environments.

Thompson et al. (2015a) present a method for deploying observations using the Planetary Instrument for X-Ray Lithochemistry (PIXL). This instrument is a type of spectrometer which collects samples while being moved at the end of a robot arm. This algorithm is interesting because instead of dealing with discrete classes of objects, it deals with spectra, which exist in a $\mathbb{R}^n$, instead of $\mathbb{N}^+$.

The objective of the algorithm is to determine when to deploy observations with the spectrometer with long integration times. Because of the possible imprecision of returning to previous locations, the instrument back-tracking to previous observations is not permitted.

Thompson et al. (2015a) consider two algorithms for deciding when to deploy sampling actions. The first looks to see if the currently observed spectrum, u, differs from any spectrum, v, in the library of spectra collected during operations. If the difference, by a weighted distance function, crosses a threshold τ, then the sampling action is triggered. The second algorithm tested in the work will deploy a sampling action if any of the channels in the currently observed spectrum is greater than n standard deviations of the mean, estimated from the previously collected spectral data.

Both these algorithms can be viewed as types of novelty detection. In the first case, the library of spectra and the distance function forming a density estimator over the space over observations. In the second case, the distributions over the channels form their own density estimates over the space of possible observations. Both approaches produce improvements in performance compared to the random and periodic sampling algorithms tested as baselines.

Foil presents an alternative approach to deploying the sampling actions in (Foil, 2016, Chapter 6). They consider classifications of the spectra that are being observed by the instrument. Using a probabilistic classifier, they attempt to determine if the probability of the class being misclassified is above a certain threshold ($P(\text{misclassification}) > 0.997$). They claim results that are at least as good as or better than the methods presented in (Thompson et al., 2015a).

Previous work by the author employing optimal foraging techniques for science autonomy has considered robots with sampling budgets limited by a number of containers and assumed knowledge of the number of sampling opportunities that would occur (Furlong and Wettergreen,

2014b,a). While the limited sampling budget is realistic, foreknowledge of the transect is not necessarily so. This chapter improves upon the prior work by using productivity, the ratio of information gained to resources expended, to reason about sampling choices and gives a constraint of time instead of an unknowable number of sampling opportunities. In those works we found that algorithms which simply engage with every available opportunity never outperform uniform and foraging sampling algorithms, for that reason we do not consider them in this chapter.

3.2 Method

We consider a scenario where a rover is following a path set for it by remote scientists. The robot has a budget of 100 units of time which it can expend while following the trajectory. While following this path the rover will repeatedly encounter objects that belong to one of K possible classes. Initially the robot does not know how many different types of materials it may encounter.

At every encounter the robot has a choice of sampling that object, represented by taking action $x \in X$ and making an observation Z, or continuing along the path in the hopes of finding a more interesting sampling opportunity. The role of the agent is to determine $P\left(Z|x\right) = \theta_x^z \left(1-\theta_x\right)^{1-z} \forall x \in X$.

The experimental setup is a variation on Charnov's patchy foraging (see Section 3.1.3). In this case we assume a patch is exhausted by taking one sample. If the agent chooses to continue searching it will be presented with a new object, drawn with probability $P\left(X = x\right)$. The reward for taking a sampling action is information gained about the underlying distribution, which is the reduction in the entropy of $P(Z|X = x)$ as a result of the latest observation. This reward function decreases in expectation, but in some instances may increase. However, if the distribution underlying the classes of objects changes, then we might expect to see the rewards increase for a period.

In this chapter we distributions underlying the classes of objects as Bernoulli random variables, representing whether a phenomenon of interest is present or not. In the grounding example of searching for life in a planetary science setting, it would model whether or not a material is colonized by microbes.

We place a Beta prior on the parameter $\theta_x \sim Beta\left(\alpha_x, \beta_x\right)$ that determines the probability that a class of material is colonized. Post observation we estimate that the probability of a class of object being colonized is $\mathbb{E}\left[\theta_x\right] = \alpha_x / \left(\alpha_x + \beta_x\right)$. Here α_x is the number of times material x was observed being colonized ("success"), and β_x is the number of times material x was observed as not being colonized ("failure").

We anticipate that the agent will encounter a number $K = |X| \leq \infty$ classes of random variables while exploring. Different experiments have different values of K. However the agent is never informed of how many classes of objects exist in the environment, and as such the algorithms must adapt as they encounter new classes of objects.

3.2.1 Algorithms

Three algorithms for sampling decision-making are evaluated in these experiments. Two of these algorithms estimate the value of action $x \in X$ by using Lindley's value of an experiment

(Lindley, 1956), given in Equation 3.1. This reward represents the expected information gain over all possible observations that may result from choosing to take sampling action x.

$$R(x) = H(\theta_x | z_{x,1:t-1}, x) - \mathbb{E}_Z [H(\theta_x | z_{x,1:t}, x)], \tag{3.1}$$

Where $z_{x,1:t}$ refers to the t observations that were collected for object class x. The expectation over Z is computed from the robot's current belief in $P(Z|X = x)$, which is where the prior $Beta(\alpha, \beta)$ has influence. In our case this value can be computed exactly, because the space of possible values of Z is small. However, for different random variables this would require estimating the value if closed-form solutions are not available.

In the final experiment we consider a fourth algorithm, which is a modified form of the Foraging algorithm. This algorithm attempts to detect whether the distribution underlying one of the classes of objects has changed. Should it do so, the algorithm caches the previous observations and re-initializes the algorithm.

Greedy (Baseline Algorithm) The first baseline algorithm, **greedy** sampling (Algorithm 3.1), will only choose to sample the encountered object, x_t, if it is has the highest reward compared to any other $x \in X$. This algorithm does not take into account the cost of moving to finding the next x_t, nor the rate at which they arrive. This algorithm corresponds to the simple greedy strategy of maximizing immediate reward.

Algorithm 3.1 Greedy Sampling Strategy

```
function INIT_GREEDY_SAMPLING
    X ← ∅
    R(·) ← ∅
end function
function GREEDY_SAMPLE(x_t)
    if x_t ∉ X then
        X ← X ∪ x_t
        return ACTION_SAMPLE
    end if
    if R(x_t) > R(x) ∀ x ∈ X \ {x_t} then
        return ACTION_SAMPLE
    else
        return ACTION_EXPLORE
    end if
end function
```

Uniform (Baseline Algorithm) The second baseline algorithm, **Uniform** sampling, will choose to sample x_t if any other class of object, $x \in X$, has been sampled more than x_t. Like Greedy, this algorithm does not take into account the cost of traverse nor the cost of taking a sampling action. This algorithm attempts to distribute samples uniformly across all classes. Uniform sampling provides valuable comparison as it has been previously shown to be a robustly successful strategy (Furlong and Wettergreen, 2014b).

Algorithm 3.2 Uniform Sampling Strategy

function INIT_UNIFORM_SAMPLING
  X $\leftarrow \emptyset$
  $N_{(\cdot)} \leftarrow \emptyset$
end function
function UNIFORM_SAMPLE(x_t)
  if $x_t \notin X$ **then**
    $X \leftarrow X \cup x_t$
    $N_{x_t} \leftarrow 1$
    return ACTION_SAMPLE
  end if
  if $\exists\, x \in X \setminus \{x_t\}\ s.t.\ N_{x_t} < N_x$ **then**
    return ACTION_SAMPLE
  else
    return ACTION_EXPLORE
  end if
end function

Foraging The proposed algorithm, **foraging** (Algorithm 3.3), chooses to sample if the expected rate of reward of x_t is greater than or equal to the expected reward from continuing to explore the environment and sample the next encountered object. We call the ratio of expected reward to costs the *productivity* of the algorithm. The foraging algorithm captures exploration and sampling costs, J_e and J_s in Algorithm 3.3, respectively, when making its decision.

We place a Dirichlet prior on the occurrence of the classes of objects, estimating the probability of encountering class x_t as $\hat{P}(X = x_t) = n_{x_t} / \left(\sum_{x \in X} n_j\right)$, where n_{x_t} is the number of times x_t has been encountered. Initially, all classes of objects $n_x = 0$, so the algorithm only believes in the existence of a class of object after it has been observed. The distribution $\hat{P}(x_t)$ is used to compute the estimated value in the environment, in Algorithm 3.3.

Algorithm 3.3 Foraging Sampling Strategy

function INIT_FORAGE_SAMPLING
  $X \leftarrow \emptyset$
  $R(\cdot) \leftarrow \emptyset$
  $N_{\cdot} \leftarrow \emptyset$
end function
function FORAGE_SAMPLE(x_t)
  if $x_t \notin X$ **then**
    $X \leftarrow X \cup x_t$
    $N_{x_t} \leftarrow 0$
    return ACTION_SAMPLE
  end if
  $N_{x_t} \leftarrow N_{x_t} + 1$
  sample $\leftarrow R(x_t) / J_s$
  explore $\leftarrow \mathbb{E}_X [R(x_t)] / (J_s + J_e)$
  if sample $\geq$ explore **then**
    return ACTION_SAMPLE
  else
    return ACTION_EXPLORE
  end if
end function

Foraging with Change Detection The foraging with change detection algorithm uses the same decision rule as Algorithm 3.3, but after it makes an observation it checks to see if the distribution underlying those observations has changed, as in the function "DETECT_CHANGE" in Algorithm 3.4. It detects the change with a likelihood ratio test. It maintains two windows of observations for each $x \in X$, one which is initially populated with **window_size** many observations, the other populated with the **window_size** most recent observations.

The two observation windows represent hypotheses about the parameter θ_x. A third window of the **sample_size** most recent observations is used as the test population. An instance of Wald's sequential probability ratio test (Wald, 1945) determines if the observations in the second window represents a different distribution from the first. The threshold for detecting a change in the distribution, **change_threshold**, is selected as specified in (Wald, 1945). **window_size** is arbitrarily set to be 30, and **sample_size** to 5. If a distribution change is detected the current world model is cached, and the rover resets its sampling algorithm to an initial state.

Algorithm 3.4 Foraging with Change Detection

function INIT_FORAGE_SAMPLING
  X $\leftarrow \emptyset$
  $R(\cdot) \leftarrow \emptyset$
  $N_{(\cdot)} \leftarrow \emptyset$
  window$_{a,x} \leftarrow$ queue($\emptyset$)
  window$_{b,x} \leftarrow$ queue($\emptyset$)
  sample$_x \leftarrow$ queue($\emptyset$)
  sample_size $\leftarrow 5$
  window_size $\leftarrow 30$
end function
function DETECT_CHANGE($x, z_{x,t}$)
  if size(window$_{a,x}$) $<$ window_size **then**
    push(window$_{a,x}, z_{x,t}$)
  end if
  push(window$_{b,x}, z_{x,t}$)
  if size(window$_{b,x}$) $>$ window_size **then**
    pop(window$_{b,x}$)
  end if
  push(sample$_x, z_{x,t}$)
  if size(sample$_x$) $>$ sample_size **then**
    pop(sample$_x$)
    $\theta_{a,x} \leftarrow$ sum(window$_{a,x}$)/size(window$_{a,x}$)
    $\theta_{b,x} \leftarrow$ sum(window$_{b,x}$)/size(window$_{b,x}$)
    $\Lambda \leftarrow \sum_{j=0}^{j=\text{sample\_size}} \log\left(\frac{P(\text{sample}_x(j)|\theta_{a,x})}{P(\text{sample}_x(j)|\theta_{b,x})}\right)$
    if $\Lambda >$ change_threshold **then**
      cache(window$_{a,x}$) $\forall x \in X$
      window$_{a,x} \leftarrow$ window$_{b,x}$ $\forall x \in X$
      init_forage_sampling()
    end if
  end if
end function

3.3 Experiments

We conducted five experiments to demonstrate the effectiveness of our algorithm, varying the underlying distribution of each class, class arrival probability, and introducing a class distribution change during the experiment. The costs of sampling and searching were varied over $\{0.1, 0.2, 0.5, 0.75, 1.0, 1.5, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ for experiments 1 through 4. In all experiments, we ran 50 trials of each algorithm for each setting of experiment parameters and costs. In each experiment we consider different numbers of objects, K, and we assume that the robot

makes no errors in identifying these classes.

3.3.1 Experiment 1 - Uniform Arrival Distribution, Different Underlying Distributions

In the first experiment the arrival probability is fixed with a constant uniform distribution. That is to say the probability that the next random variable to be presented to the agent is $P(X = x) = 1/3$. In this experiment we vary the underlying distribution of the random variables, $P(Z|X = x)$, which is the probability that objects in class $x \in X$ is colonized. The used values are given in Table 3.1.

Table 3.1: Experiment 1 Parameter Settings

Experiment	$P(Z\|X = 1)$	$P(Z\|X = 2)$	$P(Z\|X = 3)$
1.1	0.01	0.50	0.99
1.2	0.01	0.30	0.01
1.3	0.01	0.50	0.01
1.4	0.01	0.75	0.01
1.5	0.01	0.99	0.01

3.3.2 Experiment 2 - Skewed Arrival Distribution with Identical Underlying Distributions

We consider arrival probabilities which are highly skewed such that there is one overwhelmingly available class of object. We use two kinds of distributions to examine this case. The first, with $K = 6$ objects, we set the arrival probabilities as per Table 3.2.

Table 3.2: The highly unbalanced arrival distributions used in this experiment. One class is given a majority of the probability mass, while the remaining probability mass is shared among the remaining classes of objects. In these experiments we let $K = 6$.

Experiment	$P(X = 1)$	$P(X = x)\,\forall\, x \in \{2, \ldots, K\}$
2.1	0.9	0.1/(K-1)
2.2	0.8	0.2/(K-1)
2.3	0.7	0.3/(K-1)
2.4	0.6	0.4/(K-1)
2.5	0.5	0.5/(K-1)

We also consider a skewed distribution which is not as extreme. For this we use a Zipfian distribution (Newman, 2005), given in Equation 3.2. We consider different numbers of objects with $K \in \{5, 6, 7, 8\}$. The list of experiments using this arrival distribution is given in Table 3.3.

$$P_{Zipf}(X = x; s, K) = \frac{1/x^s}{\sum_{n=1}^{K}(1/n^s)} \tag{3.2}$$

Table 3.3: Arrival distributions which follow Zipf's law. These distributions favour some classes of objects much more than others. In these experiments the number of classes of objects varied from $K = 5$ to $K = 8$.

Experiment	$P(X = x)$
2.6	$P_{Zipf}(X = x; s = 1, K = 8)$
2.7	$P_{Zipf}(X = x; s = 1, K = 7)$
2.8	$P_{Zipf}(X = x; s = 1, K = 6)$
2.9	$P_{Zipf}(X = x; s = 1, K = 5)$

In all experiments in this section we set $P(Z|X = x) = 0.3 \; \forall \; x \in X$. We compare the performance of the Foraging algorithm against the Greedy and Uniform sampling algorithms.

3.3.3 Experiment 3 - Skewed Arrival Distribution with Distractor Object

In this experiment we consider Zipfian and skewed arrival distributions and we vary the underlying distribution of the second most common class of object, which for both distributions is $X = 2$. We have $K = 8$ objects in this experiment. The arrival distribution for all classes and the underlying distribution for the most common class is given in Table 3.4. $P(Z|X = x) = 0.3 \; \forall \; x \in \{2, \ldots, K\}$. We want to determine how the effect of a object with a surprising underlying distribution will effect the behaviour of the Foraging algorithm.

Table 3.4: The Different settings for the arrival distributions of the different classes of objects and the distribution underlying the most commonly occurring class.

Experiment	**Arrival Distribution**	$P(Z\|X = 2)$
3.1	$P(X = 1) = 0.8, P(X = 2, \ldots, K) = 0.2/(K - 1)$	0.1
3.2		0.3
3.3		0.5
3.4		0.6
3.5	$P(X = x) = P_{Zipf}(X = x; s = 1, K = 8)$	0.1
3.6		0.3
3.7		0.5
3.8		0.6

3.3.4 Experiment 4 - Skewed Arrival Distribution with Random Underlying Distributions

In the previous experiment the majority of the distributions underlying the classes of object were held constant at $P(Z|X = x) = 0.3$. In this experiment we consider four different settings of $P(Z|X = x)$ which were drawn randomly from the interval $]0, 0.5]$. These parameters are given in Table 3.5.

Table 3.5: The underlying distributions used in experiment 4. For Experiments 5.1 to 5.4 we use the unbalanced distribution where $P(Z|X=1)=0.8$ and the remaining 7 objects share the remaining 0.2 of the probability mass. In experiments 4.5 to 4.8 the arrival distribution follows the Zipf distribution, $P_{Zipf}(X; s=1, K=8)$.

Condition	Experiments	$P(Z\|X=\{1,\ldots,K\})$
RAND1	4.1,4.5	$\{0.14, 0.28, 0.28, 0.04, 0.27, 0.42, 0.38, 0.43\}$
RAND2	4.2,4.6	$\{0.10, 0.32, 0.12, 0.32, 0.23, 0.39, 0.36, 0.06\}$
RAND3	4.3,4.7	$\{0.04, 0.44, 0.01, 0.16, 0.31, 0.49, 0.03, 0.50\}$
RAND4	4.4,4.8	$\{0.34, 0.33, 0.25, 0.07, 0.14, 0.39, 0.37, 0.31\}$

3.3.5 Experiment 5 - Distribution Change

In this experiment we compare the Foraging algorithm against the Foraging algorithm with change detection, alongside a uniform sampling algorithm that collects samples for every sampling opportunity the foraging algorithm encounters. We compare only to a version of the Uniform sampling algorithm which already knows how many objects exist in the environment. Consequently it has an unfair advantage over the Foraging algorithm.

We return to $K=3$ classes of objects, and fix the arrival probabilities $P(X)=\{0.4, 0.3, 0.3\}$, and fix the sampling and searching costs at 0.01 and 0.1, respectively. Halfway through this experiment the underlying distribution is changed from $P(Z|x)=\{0.001, 0.500, 0.001\}$ to $P(Z|x)=\{0.001, 0.001, 0.300\}$. In the planetary setting this would represent moving between environmental regions, causing a change in colonization behaviour. The change occurs when the agent gets halfway along its path.

3.4 Results

To determine the success of an algorithm we measure the KL divergence between the true and estimated distributions. The performance of each algorithm is measured by the error in the algorithms' estimate the underlying distributions by reporting the sum of the KL divergence of the underlying distributions for the different classes of objects. This performance metric is given in Equation 3.3.

$$\text{error}(alg) = \sum_{x \in X} D_{KL}\left(\theta_x || alg.\hat{\theta}_x\right) \tag{3.3}$$

For experiments 1 through 4 we present 3D plots showing how the effect on the error in estimating the parameters θ_x from using the Foraging algorithm over the baseline algorithms as cost of sampling, J_s, and exploring, J_e, are changed. In addition, we present 2D plots showing where either the foraging algorithm performs better than the control algorithm, or the control algorithm performs better than the foraging algorithm, or when their performance is indistinguishable.

We report the effect size with Cohen's d (Cohen, 2013). Cohen's d is the ratio of the mean to the standard deviation of the difference between the trials. Values greater than 1.3 are considered to be very large, above 0.8 to be large, and below 0.5 to be moderate, and below 0.2 to be

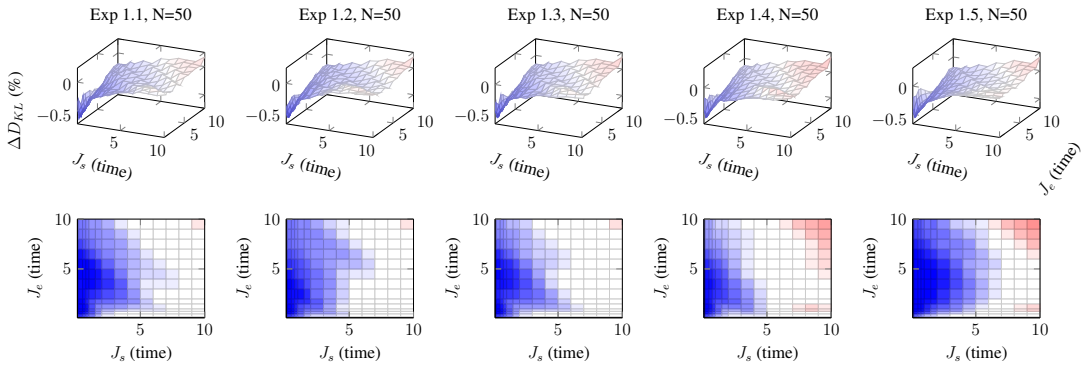


Figure 3.2: In this situation the Foraging algorithm performs as well as or better than the Greedy algorithm. With the exception of the very high sampling and exploration costs in Experiment 1.4 and 1.5.

insignificant. In the plots of Cohen's d where the behaviour of the algorithms is indistinguishable we set the value to zero.

In the figures plotted, dark blue means that the foraging algorithm had superior performance, and the red regions means that baseline algorithms had superior performance. In the plots of effect size where the graph is white and set to zero that means there was not a statistically significant difference in performance, to a 95% confidence level. Reports of the performance of the three algorithms relative to a strategy which always samples is available in Appendix C.

3.4.1 Experiment 1 Results - Uniform Arrival Distribution, Different Underlying Distributions

In this experiment we had a small number of objects, and a uniform arrival distribution we can see that the Foraging algorithm performs as well as or better than either the Greedy (Figure 3.2) or Uniform (Figure 3.3) sampling algorithms. The Greedy algorithm has a slight advantage when there are very high sampling and exploration costs for Experiment 1.4 and 1.5, there the Greedy algorithm has a slight, but statistically significant improvement over Foraging. Across the board, Foraging performs as well as or better than Uniform sampling, mainly where there are small sampling costs but large exploration costs.

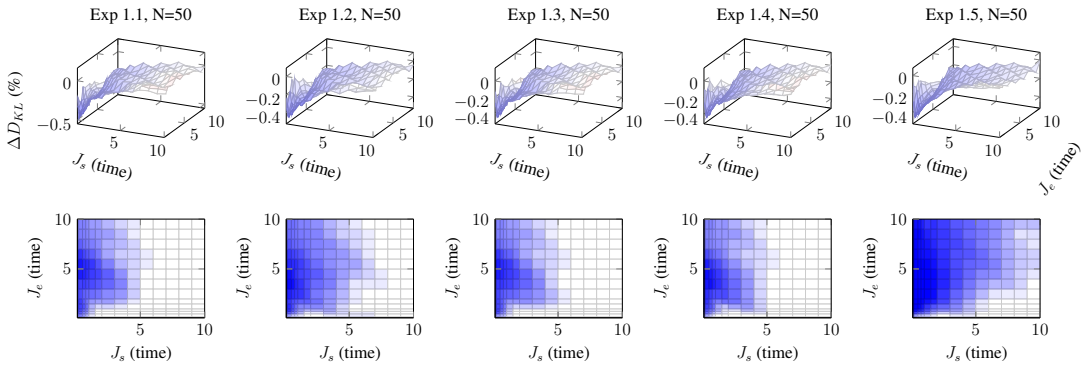


Figure 3.3: In this situation the Foraging algorithm is as good as or better than the Uniform algorithm, for all sampling and exploration costs.

We can see here that when the arrival distribution is uniform and the number of classes are small, the Foraging algorithm is an acceptable algorithm to use. However, as can be seen in Figs. C.1 to C.3, the approach of sampling every opportunity is a competitive approach, and outperforms all the algorithms for large sample and exploration costs.

3.4.2 Experiment 2 Results - Skewed Arrival Distribution, Identical Underlying Distributions

When the arrival distribution follows the unbalanced distribution, and the underlying distributions are uniform, the Foraging algorithm generally performs as well or better than the Greedy algorithm, as seen in Figure 3.4. There is a small region where the Greedy algorithm performs better than Foraging, and that seems to be at low exploration costs, and shifts rightward as the arrival distribution becomes less unbalanced.

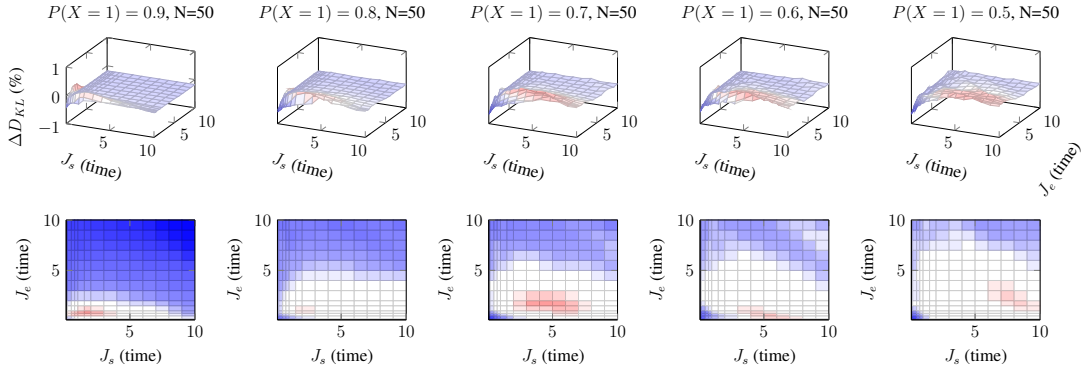


Figure 3.4: Foraging vs Greedy Algorithm, unbalanced arrival distribution. As the arrival distribution approaches uniform, the performance of the Foraging algorithm approaches that of Greedy. Top row is the percent change in performance, bottom row is the effect size of the change in performance. Dark blue means Foraging is better, dark read means the baseline algorithm is better, white means there is no statistically significant difference between the algorithms.

In comparison to the Uniform algorithm, we can see that the Uniform algorithm is statistically significantly better than the Foraging algorithm for many sampling costs and small to moderate exploration costs. However, for larger exploration costs the Foraging algorithm performs better than the Uniform algorithm, although that gap decreases as the arrival distribution becomes less unbalanced. As the distribution becomes less unbalanced the region where the Uniform algorithm is better than Foraging becomes more diffuse. Further, the minimum sampling cost where Uniform performs better than Foraging increases. For small sampling and exploration costs Foraging is significantly better than Uniform.

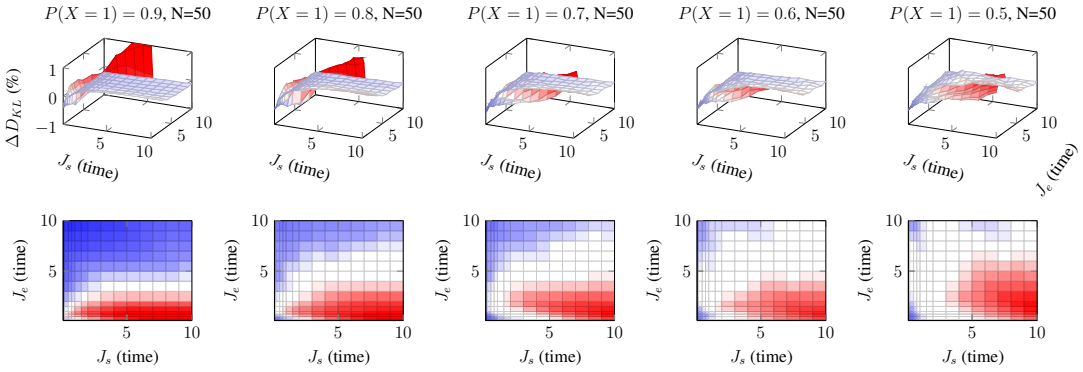


Figure 3.5: Foraging vs Uniform Algorithm, unbalanced arrival distribution. As the arrival distribution approaches uniform, the performance gap between Foraging and Uniform algorithm decreases. Uniform has an advantage for moderate to large sampling costs, whereas Foraging tends to have an advantage for moderate to large exploration costs. Top row is the percent change in performance, bottom row is the effect size of the change in performance. Dark blue means Foraging is better, dark red means the baseline algorithm is better, white means there is no statistically significant difference between the algorithms.

Comparing all the algorithms to an approach that samples every object it encounters, as illustrated in Figs. C.4 to C.6, we can see that while the Foraging algorithm performs strictly as good as or better. The Greedy and Uniform algorithms perform better than the Foraging algorithm for some cost settings, as discussed above, but occasionally they perform worse than always sampling.

When the arrival distribution follows Zipf's law, as in Fig. 3.6, we see that the Foraging algorithm is as good as or better than the Greedy algorithm, with the small exception of moderate sampling costs. Foraging has a statistically significant improvement for large sampling costs when the number of classes of objects are larger. However, as this number decreases, the gap between Foraging and Greedy decreases.

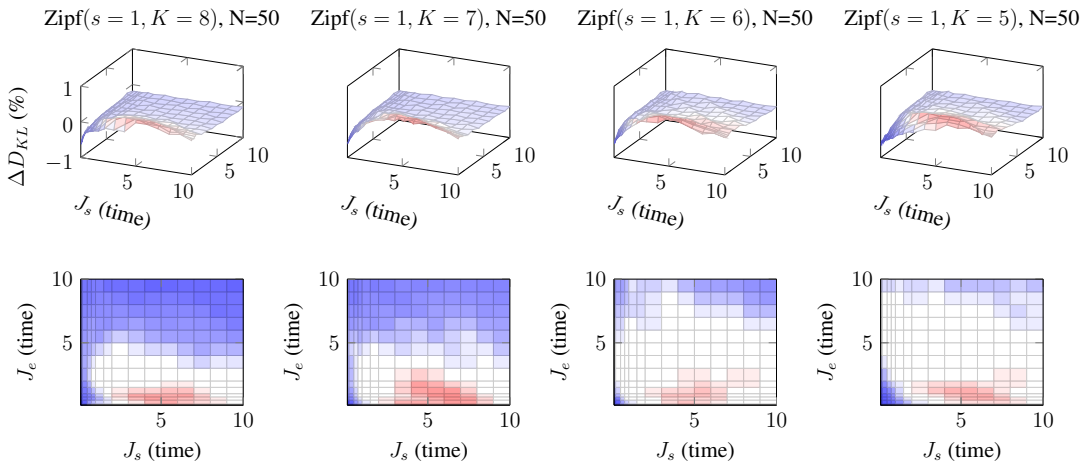


Figure 3.6: Foraging vs Greedy Algorithm, arrival distribution follows Zipf's law. As the number of objects are reduced the performance of the foraging algorithm decreases. Top row is the percent change in performance, bottom row is the effect size of the change in performance. Dark blue means Foraging is better, dark read means the baseline algorithm is better, white means there is no statistically significant difference between the algorithms.

When compared to the Uniform algorithm, in Fig. 3.7, we see that Foraging is as good as or better than Uniform for moderate to small sampling costs, and for moderate to large exploration costs. When samples are expensive the Uniform algorithm generally performs better with large effect size, and this area tends to be more diffuse as the number of objects are reduced.

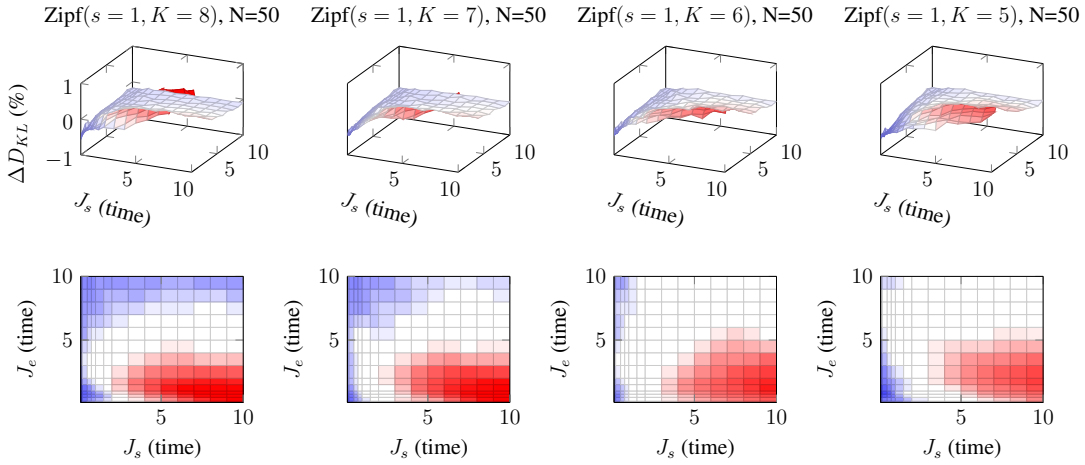


Figure 3.7: Foraging vs uniform Algorithm. As number of objects is decreased the Foraging algorithm loses its advantage for large exploration costs. However, the degree of advantage that the Uniform algorithm has decreases with fewer objects. Uniform sampling has superior performance with large effect size for small exploration costs and large sampling costs. Top row is the percent change in performance, bottom row is the effect size of the change in performance. Dark blue means Foraging is better, dark read means the baseline algorithm is better, white means there is no statistically significant difference between the algorithms.

Again, while the Foraging algorithm does not always perform better than the Greedy or

Uniform algorithms, we can see that it does not perform worse than an algorithm which always collects samples, as in Fig. C.9. Uniform (Fig. C.7) and Greedy (Fig. C.8) perform worse than always sampling for small sampling and exploration costs.

We can see that the relative performance of the algorithms is sensitive to the arrival distribution. The Foraging algorithm has greater performance with small sampling and exploration costs. Effectively, when there are many opportunities to sample, the Foraging algorithm takes advantage of them when Greedy and Uniform do not. Uniform, however, will only sample an object if it is not the most sampled object. Consequently when the sample costs are expensive and the exploration costs are low, the optimistic stance of Uniform provides a distinct advantage.

3.4.3 Experiment 3 Results - Skewed Arrival Distribution with Distractor Object

In this experiment we attempted to determine the effect of a distractor object on the relative performance of the Foraging algorithm. We varied the underlying distribution of the second most frequently encountered object to vary from the initial prior belief.

When the arrival distribution was the unbalanced distribution, there was no major change to the performance relative to the Greedy algorithm, as seen in Figure 3.8. Generally the Foraging algorithm is as good as or better than the Greedy algorithm for a wide variety of sampling and exploration costs. There is a small region where the Greedy algorithm has a statistically significant, with small effect size ($d < 0.5$), improvement over Foraging.

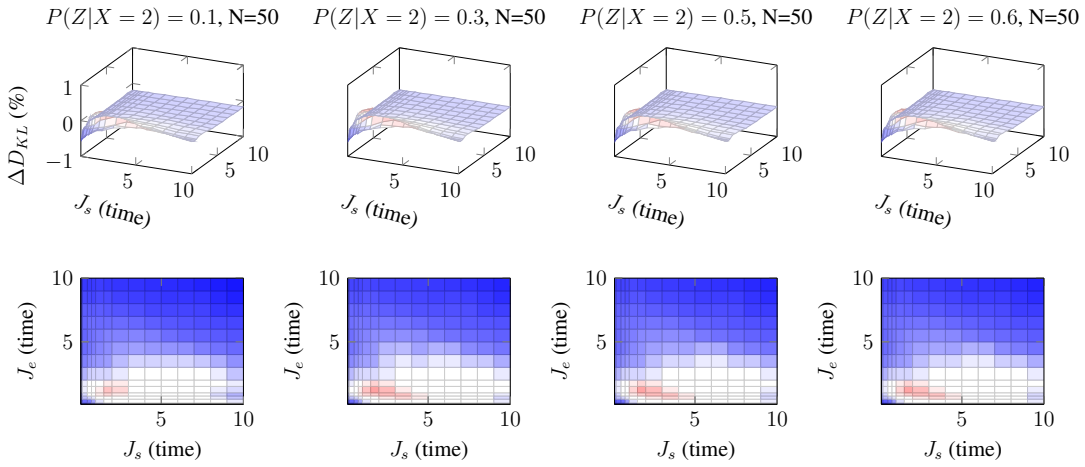


Figure 3.8: The Foraging algorithm vs Greedy, unbalanced arrival distribution. There is no substantial change in performance across the different settings of the underlying distribution. Greedy has a slight advantage for sampling costs of about 1.5 and exploration costs of about 1.

When compared to the Uniform algorithm, Foraging is as good as or better for moderate to large exploration costs, as in Figure 3.9. However, for small exploration costs, the Uniform algorithm is statistically significantly better and with large effect size. There is a small improvement in Foraging at the most extreme value of the underlying distribution of the distractor object.

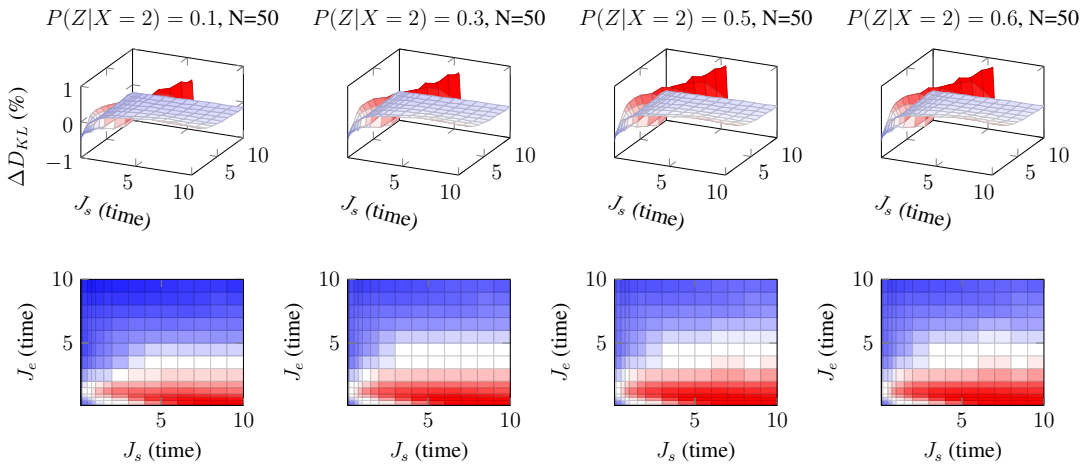


Figure 3.9: The Foraging algorithm vs Uniform, unbalanced arrival distribution. The Uniform sampling algorithm maintains the advantage for moderate to large sampling and small exploration costs. There gap in performance between Foraging and Uniform is slightly decreased for moderate exploration costs in the case $P(Z|X=2)=0.1$.

When the arrival distribution follows Zipf's law, we see that the gap in performance between Greedy and Foraging is decreased for large exploration costs. The advantage that Foraging enjoys for small exploration and sampling costs is expanded, as can be seen Fig. 3.10. Greedy gains a small, but statistically significant advantage for small exploration costs and moderate sampling costs.

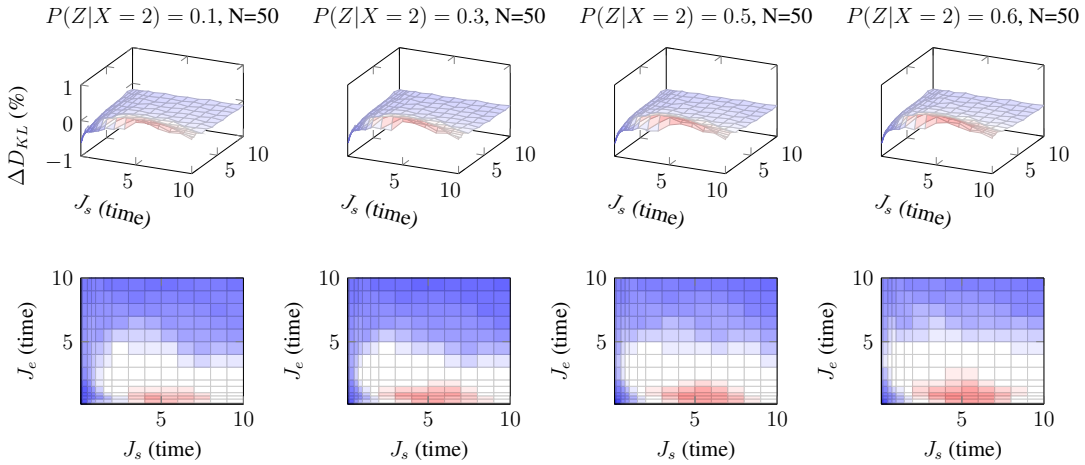


Figure 3.10: The Foraging algorithm vs Greedy, Zipfian arrival distribution. The Foraging algorithm either outperforms or is statistically indistinguishable from the Greedy algorithm, except for moderate sampling and small exploration costs. At the smallest settings of exploration and sampling costs the Foraging algorithm has a statistically significant improvement over Greedy.

Compared to the Uniform algorithm, following a Zipfian arrival distribution, the Foraging algorithm does not change significantly as a function of the distribution underlying the distractor object. The Foraging algorithm is as good as or better than Uniform for moderate to large exploration costs, and again for small sampling and exploration costs.

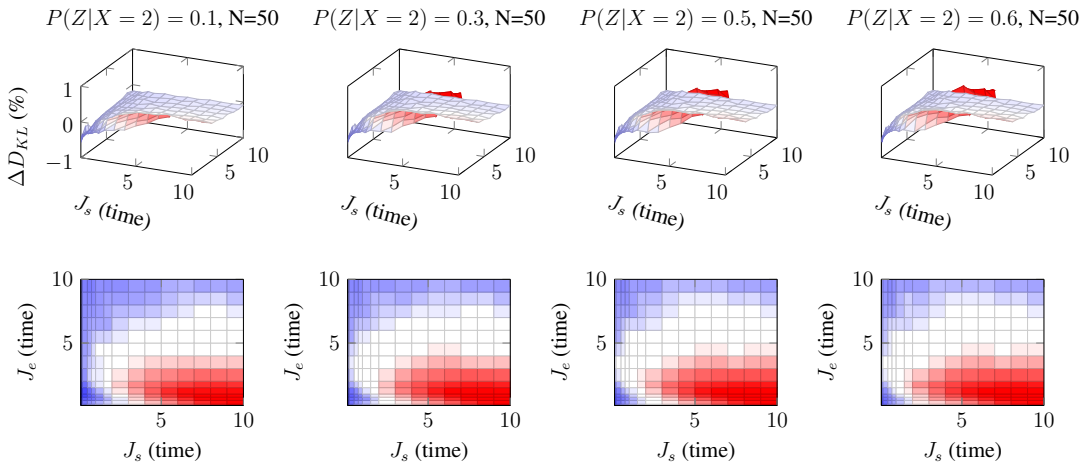


Figure 3.11: The Foraging algorithm vs Uniform, Zipfian arrival distribution. For moderate to large sampling costs and small exploration costs the Uniform algorithm is statistically significantly better than the Foraging algorithm. Otherwise the Foraging algorithm is as good as or better than the Uniform algorithm. There is no major change in performance as a function of the change in the underlying distribution of the second most common class of object, $P(Z|X=2)$. Again, Foraging is a large improvement over Uniform for small sampling and exploration costs.

For both arrival distributions the Foraging algorithm continues to be strictly as good as or better than an approach which samples everything it encounters, as seen in Figure C.12. While the magnitude of the increase may not be as great, we see that the range where the Foraging algorithm outperforms the always-sample approach is broader than in either Uniform (Figure C.10) or Greedy (Figure C.11).

The results from this experiment demonstrate that the relative performance of the Foraging algorithm isn't overtly sensitive to the underlying distribution of the second most commonly encountered object. What we can conclude from this is that the Foraging algorithm preferentially collects quantities of information instead of chasing after surprises. However, one should consider, if the encountered classes of objects do vary significantly from the prior, perhaps an explorer should spend more time investigating them. Since the Foraging and Greedy algorithms use the same reward function, and the Uniform algorithm doesn't consider the observations collected at all, none of the algorithms can be said to react to surprising observations.

3.4.4 Experiment 4 Results - Skewed Arrival Distribution with Random Underlying Distributions

In the previous two experiments we kept the underlying distributions of most of the objects constant at $P(Z|X = x) = 0.3 \; \forall \; x \in X$. In this experiment we compared the Foraging algorithm to the baselines where the underlying distributions were set as in Table 3.5. As we can see, the difference in performance of the Foraging algorithm compared to the baseline algorithms is highly sensitive to the underlying distributions of the classes of objects.

When compared to the Greedy algorithm, for either the unbalanced (Fig. 3.12) or the Zipfian (Fig. 3.14) arrival distributions, the Foraging algorithm reduces its advantages, for all but the smallest sampling and exploration costs. In the case of the RAND3 parameter setting, the Greedy

algorithm actually performs as well as or better than the Foraging algorithm for many sampling costs and for moderate exploration costs.

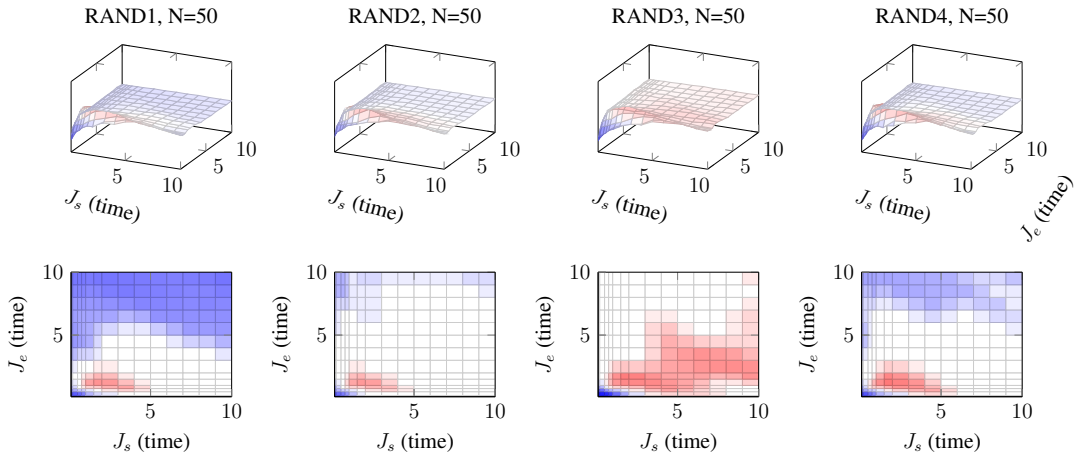


Figure 3.12: Foraging vs Greedy, unbalanced arrival distribution. The gap in performance between Foraging and Greedy closes, and in some settings Foraging loses its advantage for moderate exploration costs.

The Uniform algorithm demonstrates a substantial improvement in performance for the settings RAND3 and RAND4, with an increase in the region where it performs as well as or better than Foraging, as seen in Figure 3.13. We also see much larger regions where the performance between the two algorithms is not distinguishable at a 95% confidence interval.

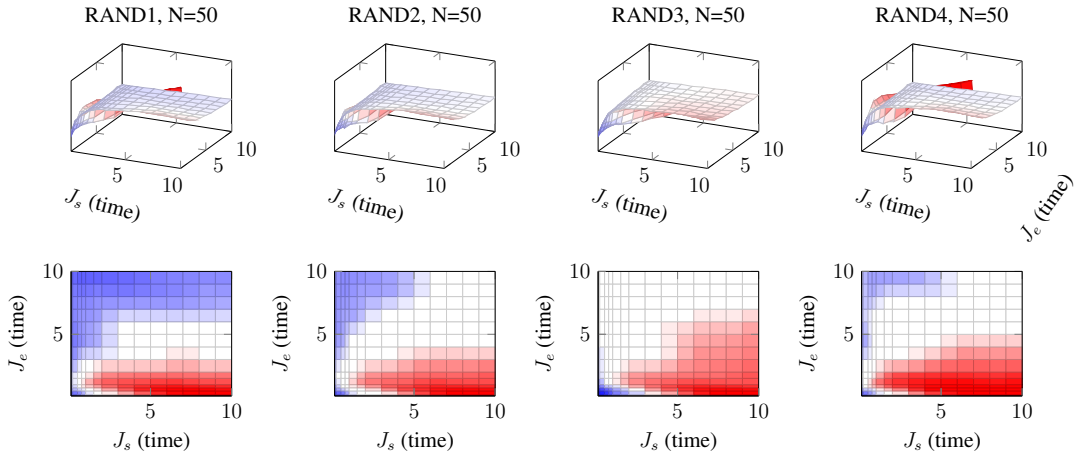


Figure 3.13: Foraging vs Uniform sampling, unbalanced arrival distribution. The region where Uniform performs better than Foraging becomes more diffuse for the settings of the underlying distributions used in RAND3 and RAND4.

We see similar behaviour for the Greedy algorithm when the classes of objects follow a Zipfian distribution (Fig. 3.14), as we do when following the unbalanced distribution. However, the Foraging algorithm is better able to preserve it's advantageous performance for small sampling

costs, across all settings of exploration costs. In all but the RAND1 setting, the Greedy algorithm is as good as or better than the Foraging algorithm for moderate to large sampling costs.

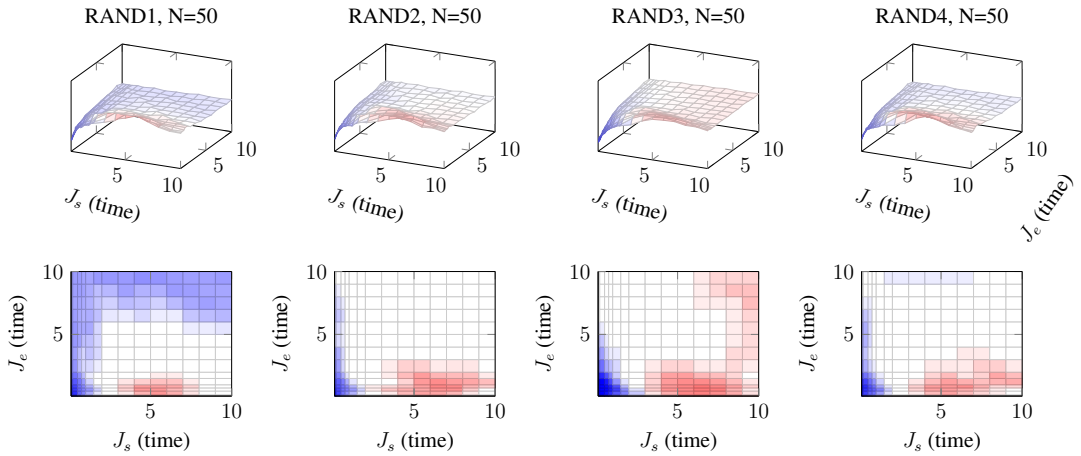


Figure 3.14: Foraging vs Greedy, Zipfian arrival distribution. The Foraging algorithm preserves the advantage for small sampling costs, but behaves as well as or worse than the Greedy algorithm for moderate to large sampling costs.

Using a Zipfian arrival distribution and comparing against the Uniform algorithm we see in Fig. 3.15 that the Foraging algorithm preserves its superior performance for sampling costs up to 3, for small to moderate exploration costs, compared to uniform underlying distributions as in Experiment 2. This is true across all four settings of the underlying distributions. However, Foraging loses the advantage when the exploration costs are large.

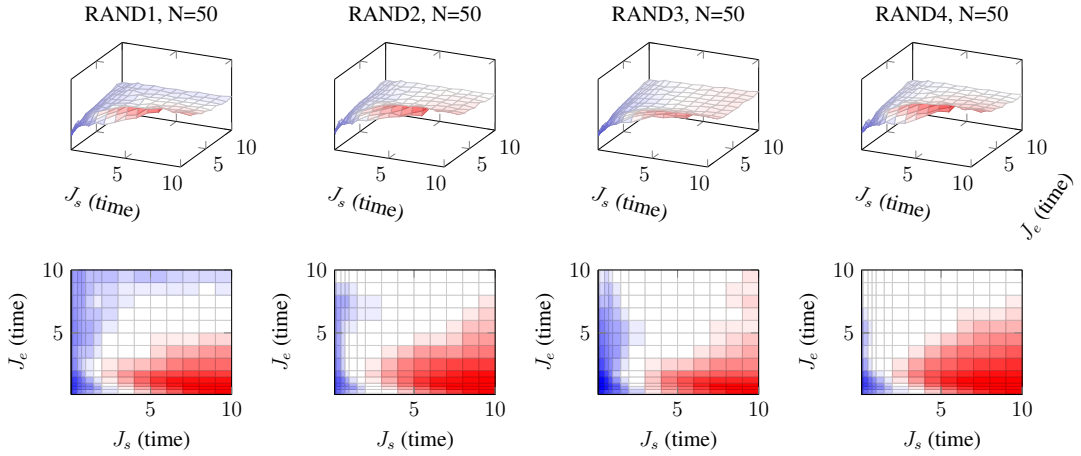


Figure 3.15: Foraging vs Uniform algorithm, Zipfian arrival distribution. Foraging maintains superior performance for small sampling and exploration cost, but loses the advantage for large exploration costs. This is most noticeable in the condition RAND4, where the Uniform algorithm is superior is over a much larger region.

What we can conclude from these experiments is that the Foraging algorithm is quite sensitive to the values of the underlying distribution. The main observation is that the performance of the

Foraging algorithm is much more variable for large exploration costs, but it preserves it's superior performance when the sampling and exploration costs are small, relative to the total budget.

When comparing to the algorithm which always samples, we observe the rare situation where the Foraging algorithm performs worse. Shown in Fig. C.15, when the arrival distribution is unbalanced and the test condition is RAND3, the Foraging algorithm is worse than an approach that always samples for large exploration and sampling costs. Otherwise Foraging is strictly as good as or better than always sampling. Again, Foraging avoids the penalty when the sampling costs are small that the Uniform (Fig. C.13) and Greedy (Fig. C.14) incur.

3.4.5 Experiment 5 Results - Underlying Distribution Change

Fig. 3.16 shows that the foraging algorithm with change detection performs substantially better than not using it. The leftmost bars show the performance of the algorithms with and without change detection immediately before the change in the underlying distribution. By employing change detection ("After" in Fig. 3.16) we see the error in estimating the underlying distribution is profoundly reduced. Notice also that the uniform sampling algorithm, while it performs better after the change detection than the standard foraging algorithm, it is still statistically significantly worse than the foraging with the change detection. Obviously, the uniform algorithm could also benefit the change detection mechanism.

If the number of opportunities to sample is not very large, then the agent will not be able to detect a change in an underlying distribution with any confidence. This must be considered when planning exploration missions. The number of samples that are needed to estimate the distribution with confidence can be determined through PAC learning bounds, but would require a confidence level determined by the designers of the mission.

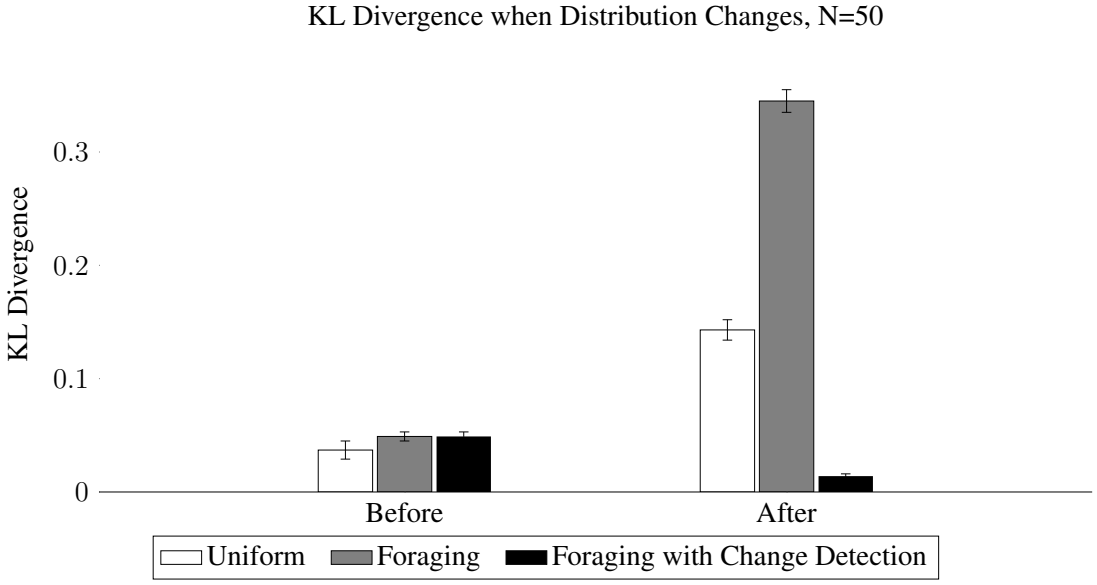


Figure 3.16: All algorithms perform comparably just before the underlying distribution change. At the end of the path the algorithm that detects changes performs substantially better than the ones that don't. Error bars represent a 95% confidence interval, estimated from 50 trials.

The sampling/exploration cost that this experiment takes place in corresponds to the region identified in the previous two experiments, and hence we see that the uniform algorithm outperforms the foraging algorithm. However, the fact remains that by recognizing a change in the underlying distribution the algorithm is able to better estimate the underlying distribution.

3.5 Discussion

While improving the modelling of field work, this approach still has some limitations. For example, the probability of encountering different classes of objects is considered to be independently and identically distributed (i.i.d). While this may be a valid first-order approximation, it could extend the utility of the algorithm to consider a Markov chain with a limited time horizon to estimate the value of continued sampling.

We also consider fixed sampling and search costs. While this work looked at different regimes of sampling and exploration costs, it could increase the fidelity of the simulation to introduce randomness into those costs.

The decision about whether or not to sample the currently available object is based on a hard comparison to the estimated value of exploring the environment. There are three underlying assumptions in this decision.

First, it is assumed that one bit of information gained is worth one unit of cost. It may be that a different relative weighting of rewards and costs would be more beneficial to behaviour. Since the decision being made by the robot is $\frac{\text{sampling value}}{\text{sampling cost}} \geq \frac{\text{exploring value}}{\text{sampling cost+exploration cost}}$, when the cost of sampling is very small, and in particular much smaller than 1, it could ensure that the algorithm will never give up on a sampling opportunity. However, when both sampling and exploration are cheap, this is not unacceptable behaviour.

Second, it is assumed that the value of classes of objects is determined exclusively by the information gained. While this protects against confirmation bias on the part of the robot, it does not permit scientists to impart preferences.

Third, it is assumed that a hard decision boundary between the value of current sampling opportunity and the estimated value of the environment is a suitable decision-making mechanism. A softer boundary could reduce the propensity of the foraging algorithm to give up on sampling opportunities.

In the final experiment in this chapter, the foraging with change detection algorithm kept fixed the number of old and new samples that are compared, and the confidence threshold for detecting a change. These parameters can greatly impact the ability to detect changes, and furthermore could conceivably mask changes in the underlying distribution which occur slowly over time. While hard changes in the underlying distribution are anticipated in the situations which inspired this work, in other scenarios this assumption must be re-examined.

The sampling budget that the algorithms were tested with was fixed arbitrarily at 100 units. While the costs of sampling and exploring were varied, the total budget was not. We would expect that this would change the topography of the surfaces of the KL divergence and confidence bounds for the algorithms.

We also consider that the robot is able to identify the different classes of objects without error. Many template-matching algorithms deployed *in situ* effectively make this assumption as

well. While this may be possible in some settings, when perception systems are deployed in unstructured environments this is almost certainly not true. Accounting for that confusion would make opportunistic sampling algorithms better suited to deployment in realistic systems.

This work can be extended in several ways. First, employ the same change detection of the sample values to the arrival probabilities. This way the exploring agent can detect when the composition of the environment changes, which may be interesting to remote scientists. Second, model more complex underlying distributions. Third, integrate site selection with a path planner in order to determine costs of different sampling actions. Finally, account for possible misclassification of the identified random variables in a scene. These additions will make progress towards robust autonomous planetary exploration.

The Foraging algorithm performs best relative to the Greedy and Uniform sampling algorithms when the sampling and exploration costs are small. Effectively, Foraging can recognize when it is sensible to engage with most of the encountered objects. However, it clearly loses out to the Uniform algorithm when the sampling costs are large and the exploration costs are small. While Foraging still out-performs a strategy of sampling everything for these cost settings, it would be good to enable Foraging to be more like Uniform sampling in these conditions, without losing Foraging's advantages when sampling costs are small.

It would be good to consider the relationship between environmental conditions and the results of sampling objects. A search procedure using one motivated by Quadratic Cauchy-Schwartz Mutual Information would be a natural method of valuing these samples - how independent are environmental parameters and the distribution underlying classes of objects. However, this does require knowledge about environmental conditions, and once we have this information in a map we then get into the notion of planning. Further, it assumes that the relevant environmental parameters are observable.

Finally, it was noted that different points of the sample/exploration cost space yield different performance for the competing algorithms. It would be valuable to develop an algorithm that can estimate the relative costs, and then use that estimate to modify which sampling strategy they should employ. This should yield greater flexibility and performance of the algorithms overall.

3.6 Summary

In this chapter we examined the case of a robot following a pre-determined trajectory while attempting to learn the distribution underlying a number of classes of objects. The fundamental idea underlying the work in this chapter is the recognition that an agent does not necessarily have the choice of which class of object to sample. We proposed a foraging algorithm to address this shortcoming. Additionally we consider a modified version of the algorithm which can identify changes in the underlying distributions.

When the likelihood of encountering the different classes of objects is uniformly distributed we see that the Foraging algorithm generally performs better than the Uniform sampling algorithm. When the costs of sampling and exploring were high the Greedy algorithm had a slight performance advantage over the proposed Foraging algorithm.

When the distribution governing which classes of objects were encountered was non-uniform we saw this performance gap narrow, for both the Greedy sampling algorithm and the Uniform

algorithm, with an increase in regions where the baseline algorithms out perform the Foraging algorithm. For larger settings of the sampling cost, the Uniform sampling algorithm out-performs the Foraging algorithm. However, the Foraging algorithm rarely under performs the Always Engage algorithm, as can be seen in Appendix C.

The result of the experimentation in this chapter demonstrate that there is not one clear winner between the Foraging, Greedy, and Uniform sampling. When exploration costs are small and sampling costs are large, the Uniform algorithm produces the best results. However, when sampling costs are small, the Foraging algorithm can yield improvements. There are also many settings of the parameters where the algorithms perform statistically indistinguishably from one another.

We must conclude that there are different scenarios when different sampling strategies have the best performance. When the arrival distribution of the classes of objects is near uniform, or the number of objects is small, it makes sense to choose the Foraging algorithm, however, once that distribution becomes unbalanced, the Foraging algorithm begins to lose out for large sampling costs, motivating the choice of the Uniform algorithm. One can consider the costs for sampling and exploration, and then decide which mode of behaviour it would be most beneficial to employ. However, if one must choose a single alternative to the approach of always engaging with samples, then we can say with confidence that the Foraging algorithm performs generally as good as or better than that strategy. Neither the Uniform or the Greedy algorithms can make this claim for the tested search and exploration costs, and arrival distributions.

The Foraging algorithm demonstrates its greatest advantage when the sampling and exploration costs are small. It can recognize when it is sensible to engage with most of the encountered objects. While, it clearly loses out to the Uniform algorithm when the sampling costs are large and the exploration costs are small, Foraging still performs better than the always sampling strategy. The Foraging algorithm can be viewed as a compromise point between the Uniform sampling algorithm's optimism, in the case of large sampling costs and small exploration costs, without losing the opportunistic zeal of a strategy that always chooses to sample.

An alternative view of this work is, if one has a secondary sensor where the cost of sampling can be controlled, and the mission has committed to a sampling strategy, then one can begin to reason about acceptable upper and lower bounds on that sampling cost. In this way the accuracy in individual samples can be traded against the overall accuracy in the estimates of the underlying distributions.

To this end, we constructed a map of where the different algorithms provide the best performance. The three algorithms were compared for all sampling and exploration costs and all experiment conditions, and the best algorithm was chosen for each point. Either the Foraging or the Uniform algorithms dominate performance, as seen in Fig. 3.17.

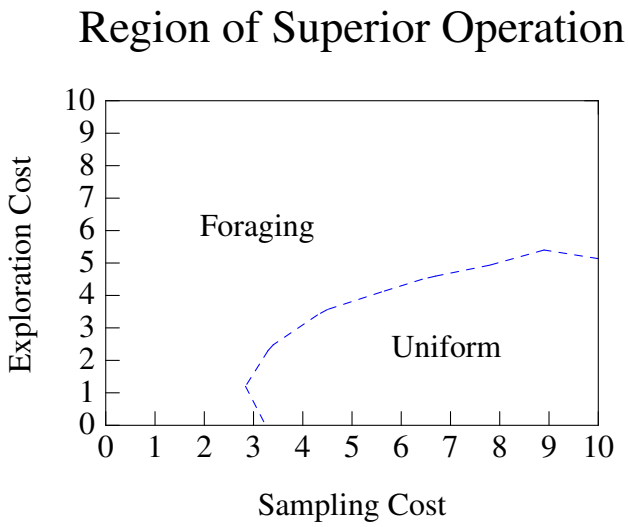


Figure 3.17: The dashed blue line gives the boundary between the regions where either the Foraging algorithm or the Uniform algorithm dominates. Note that where the Uniform algorithm performs better than the Foraging algorithm, both are better than an algorithm which samples every object it encounters. This plot is calculated only for budgets of 100 units of arbitrary time.

Designers of future missions could use the above figure to determine which style of sampling to employ. To figure out the location of the mission in the map one would have to determine the actual costs of sampling and exploration relative to the values tested in these experiments. To achieve this, given the maximum budget was 100 units of time, one would have to scale the estimated sampling and exploration costs by $\frac{100}{\text{budget}}$. Where "budget" is the mission budget expressed in units of sampling and exploration costs.

While this analysis is useful, it motivates a more rigorous analysis of the behaviour of the different algorithms. Without a theoretical relationship between the budget and the search and exploration costs, our map of relative algorithm performance remains a heuristic.

We also demonstrated that an algorithm which monitored samples for changes in the underlying distribution was better able to estimate the underlying distributions after the change than algorithms which did not. Additionally, identification of these changes offers a trigger for autonomous explorers to examine the environment for changes causing the shifts in the underlying distribution. Regardless of the sampling algorithm used, change detection would be a valuable addition to any opportunistic sampling system.

Chapter 4

Opportunistic Sampling in a Scalar Field (Prospecting)

Prospecting is a type of reactive sampling. One traverses a region with a sensor that is relatively cheap to use – the proxy sensor. The proxy sensor acts as a known indicator for some phenomenon to be investigated with an expensive, secondary, sensor. Prospecting is a distinct problem from the information foraging problem chiefly because it is not interacting with discrete targets, and because the relationship between the proxy sensor and the expensive sensor is already known.

During prospecting the agent is traversing a region with a cheap proxy sensor looking to identify places or regions to engage in sampling activities that we call Area of Interest Manoeuvres (AIMs). Examples of an AIM would be collecting soil samples with, e.g., a drill, or to drive a tight spiral pattern, interrupting the planned trajectory. Generally, the robot is navigating through a scalar or vector field, as opposed to a space littered with discrete objects, and it needs to find the best - per some objective function - location to deploy an action.

In this chapter we assume that a trajectory has been determined by some mechanism external to the agent. We can write that trajectory as $\tau(t)$ where $t \in [0, \infty[$. The prospecting sensor collects observations z_t that are drawn from some underlying distribution $G(t) = G(\tau(t))$. Which is to say that the distribution driving the readings from the proxy sensor is determined by where the agent is along the trajectory.

We seek to improve upon state of the art algorithms by being aware of the hypothesis about change points in the underlying distribution. Specifically, one change point when the distribution changes from $G(t) = G_1$ to $G(t) = G_2$ at some change point, t_{cp}. We build an algorithm upon the Sequential Probability Ratio Test (SPRT), developed by Wald (1945), to determine with confidence that a change in the underlying distribution has occurred. We demonstrate that being aware of the change in the underlying distribution can produce superior performance to state of the art approaches used in the field.

What we call prospecting is different from sample scheduling in that it is reactive to the observations collected *in situ* along the trajectory. Further we do not place a constraint on how many discrete actions should be deployed, we leave that determination to a higher-level component in a robot system. As previously stated we assume a pre-determined trajectory. While intelligently planning a trajectory could increase the performance of prospecting it does not eliminate the need

for a mechanism which makes opportunistic decisions to take discrete actions, so in this way the prospecting problem is different from planning for information gathering.

The work in this chapter was modelled on the Mojave Volatiles Prospector Project conducted in the Mojave desert in 2014 in part by the NASA Ames Intelligent Robotics Group. Here the prospecting instrument was a neutron spectrometer (NSS) and the high-cost intervention was an Area of Interest Mapping Manoeuvre (AIM). In this chapter we model an AIM as driving in a circular pattern when an area of interest is detected. The data used in this chapter was either data from the MVP project or simulated based on the distribution of NSS readings collected during this project. The MVP project is discussed in Section 4.1.

The state of the art in prospecting is best exemplified by the work of Ferri et al. (2010). As such this algorithm is used as a baseline for comparison in the experiments conducted in this chapter. This algorithm issues an AIM if an observation crosses a predetermined threshold. This and other related work are discussed in Section 4.2.

What we propose is an algorithm that attempts to determine if the source driving the readings of the prospecting sensor has changed. If a change has occurred then the robot can take any necessary actions. This algorithm is based around the sequential probability ratio test (SPRT)(Wald, 1945), which is used to determine if one of two hypotheses are more likely, given the observed data. The competing hypotheses are either that there has been no change in the underlying distribution or there has been one change in the underlying distribution.

The algorithms will be discussed in detail in Section 4.3.1. The experiments used to test them will be described in Section 4.3.3 and their results discussed in Section 4.4.

4.1 The Mojave Volatiles Prospector Project

To ground this work we consider the Mojave Volatiles Prospector (MVP) project conducted by NASA Ames in the Mojave desert in 2014 (Heldmann et al., 2015). The robot deployed in this project, pictured in Fig. 4.1, was equipped with a Neutron Spectrometer (NSS), which was used to estimate the abundance of subsurface water.

MVP served as an early test of the high-tempo scientific operations that would be required to support short-duration exploration missions like the anticipated Resource Prospector mission (Andrews et al., 2014). Robot-habitable conditions on the surface of the moon, (e.g. sunlight, temperatures above freezing) last for only two week periods at a time.

The concept of operations for Resource Prospector (RP) differs from typical planetary missions to date. Resource prospector is focused on prospecting for subsurface volatiles using point measurement instruments, such as a neutron spectrometer.

The neutron spectrometer (NSS) is an instrument that can infer the abundance of water molecules in its field of view. The NSS can sense water content up to a meter below the surface of the ground and in approximately a meter radius parallel to the ground plane. Since the NSS has a limited range and field of view, readings must be taken from the surface of the planetary body, and not from orbit.

The mode of operation for prospecting varies between long-range paths looking for variations or gradients in the field observed by the instrument. Periodically, at the discretion of the remote science team, the rover will deviate from the long-range path to engage in localized map-

Figure 4.1: KRex2 in the Mojave desert. The stainless steel and cadmium cylinders of the Neutron Spectrometer are visible on the rear of the vehicle.

ping. These localized mapping actions are called Area of Interest Manoeuvres (AIMs), and are designed to collect more information about the subsurface.

The measurement requirements for Resource Prospector differs from what is required of a field geology mission. In field geology one would focus on one region at a time, collect a wide variety of measurements thus thoroughly establishing the geological context of the site.

For Resource Prospector it would be the exception rather than the rule to spend time in one location. The time pressures of the mission drive the operations more towards long-range travel instead of repeated observations of one site.

Since the RP vehicle is designed to not survive lunar night, which saves mission costs, operations are limited to the lunar day. Consequently PR does not have the luxury of taking long periods of time to make decisions about deploying scientific instruments. Likewise, opportunities to return to previous locations may also be limited.

The work presented in this chapter is designed to identify when changes in the signal observed by the on-board instruments occur. The purpose is to act as an assistive tool for the remote science team, observing the prospecting sensor, detecting changes in the signal driving observations. The science team can use the identified changes to determine if an AIM should be deployed.

There are a number of benefits that come from separating the concerns of the path planner and the change detection algorithm. First, it reduces the search space the planner has to explore. Second, by separating the planner and the change detection algorithm the planner can produce paths which optimize one criterion, while the change detection algorithm can monitor unrelated signals. Finally, by separating path planning and the change detection algorithm, the change detection algorithm can be used even when paths are provided by human operators.

We take the conservative stance that precursor mapping information is not available to the

change detection algorithm. The algorithm itself is agnostic of the plan that is being executed, and simply monitors for changes which can be identified for remote operators. A principled way to incorporate prior information without breaking the separation between a planner and the change detection algorithm would be for either the remote science teams or a planning algorithm to use any precursor data to plan the trajectory that the robot is following.

The NSS reports counts of how many neutrons it estimates passed through the field of view of the instrument in one second blocks of time. These counts are modelled as a Poisson process. The average rate of the Poisson process is a function of the water abundance by weight and the depth of the water deposit. Generally speaking, a higher rate of counts implies a greater likelihood of water.

In the process of the MVP project it was discovered that the background rate was approximately 40 counts per second. A high number of counts, highly correlated with water is approximately 120 counts per second. A threshold for having detected water was approximately 80 counts per second. As such we used 80 as the threshold for the threshold algorithm used in this chapter.

The NSS is always on and does not consume resources other than power. Therefore we assume no sampling budget is consumed when using the NSS. We consider the fixed power budget of the NSS while prospecting as non-negotiable.

Prospecting is an important part of exploration, especially for missions focussed around *in situ* resource utilisation. An NSS is planned to be part of NASA's upcoming Resource Prospector mission. The instrument will be the primary prospecting sensor in the search for subsurface deposits of solid water. The vehicle will drill at where the water density is most likely highest. The tight time constraints of operating in one lunar day motivate effectively finding likely water density maxima as observed by the neutron spectrometer.

4.2 Prior Work

There are three major non-adaptive methods for determining where and when to deploy AIMs: Delegate to a human; scout out the area and return to relative maxima; or to use a threshold to determine when an AIM should be employed. However, human judgement is not always available, or if it is, it requires demands be placed on unbiased experts whenever the prospecting sensor is deployed.

Pre-scouting an area is a viable approach should the exploration budget have sufficient margin that it can traverse the length of the path at least twice. But this approach does make the assumption that the interesting observations that are made along a traverse have no temporal component and will still be there when the robot returns to the site. This is not necessarily a valid assumption, and as such decisions that are made reactively would be more effective.

Yoerger et al. (2000) use a robot to scan an area by searching a grid. This approach is principled and avoids collecting data. However, erroneous readings can be over-represented in the data, as there is no attention paid to the statistical confidence in the reported data. Likewise, any anomalies that do appear are uninvestigated, as this is a non-adaptive approach. The work presented in (Yoerger et al., 2007) improves on their prior approach by having the vehicle follow a search pattern, then identify anomalies – readings that pass a certain pre-determined threshold

– and then return along the path to gather more observations of those anomalies.

The third approach of using a threshold to determine when to conduct an AIM is, in principle, feasible, but only if the statistics of the environment are already well understood going into the exploration site. A static-threshold approach does not adapt to the statistics of an environment, and as such it may execute AIMs either too frequently, in sufficiently abundant territories, or not frequently enough, should sites with high concentrations of the reading of interest be missed, or the higher concentration regions be not as highly concentrated as the remote scientists hoped.

The paper (Ferri et al., 2010) describes a robot that is attempting to localize undersea hydro-thermal vents. They track the vents by looking at the concentration of chemicals in the water which are emitted from hydro-thermal vents. Should the observed concentration pass a threshold they initiate a spiral area of interest mapping manoeuvre in order to build a more informed map of chemical concentrations.

In the case of the Resource Prospector project, the purpose is to localize maxima of the neutron signal in order to drill in a location likely to be most abundant with water. In the case of the work of Ferri et al. (2010) the objective was to localize geothermal vents, places likely to contain new and exciting ecosystems.

The work in (Ferri et al., 2010) is a method for deploying AIMs in an online fashion, improving on (Yoerger et al., 2007). In the previous work the vehicle had to travel to the end of the trajectory before it determined where to deploy the AIMs, where as (Ferri et al., 2010) the vehicle can determine to deploy AIMs as it is travelling. The AIMs that the algorithm deploys are spiral motions which collect more observations with the chemical sensor.

In (Ferri et al., 2010) they present an algorithm which uses an adaptive threshold, and compare it to a baseline of a fixed threshold algorithm. They employ their algorithms on pre-recorded data from transects collected with the ABE robot.

As the robot travels along the length of the transect they increase and decrease the threshold based on how many AIMs have been deployed vs how many AIMs it was suggested to be deployed. The suggested number of AIMs deployed is not a hard upper limit on how many can be deployed.

The threshold, γ, is updated when a "patch" is ended. A patch is a region of the transect where readings come close to triggering an AIM, but do not necessarily trigger an AIM. A schematic of the decision rule for updating the threshold is given in Algorithm 4.1.

Algorithm 4.1 The algorithm checks to see if the number of AIMs deployed is keeping pace, as a fraction of the suggested number of AIMs, with the fraction of the transect covered. The algorithm starts off with a threshold, γ determined by a default threshold, specified by the scientists, which is denoted γ_B. There is also defined a minimum threshold, γ_{min}.

transect_frac $\leftarrow \frac{\text{distance travelled}}{\text{transect length}}$
$K \leftarrow 2$
if $\frac{\text{aims deployed}}{\text{suggested aims}} \leq$ transect_frac **then** ▷ The threshold should be lowered
  $\gamma_{low} \leftarrow$ median$(\{\max(p) : \forall\, p \in \text{patches}\ s.t.\ \max(p) < \gamma\})$
  $\gamma' \leftarrow \max(\gamma_B * (1 - \text{transect\_frac}) + \gamma_{low} \times K \times \text{transect\_frac}, \gamma_{min})$
  $\gamma' \leftarrow \min(\gamma', \gamma_B)$
else ▷ The threshold should be raised
  $\gamma_{up} \leftarrow$median$(\{\max(p) : \forall\, p \in \text{patches}\})$
  $\gamma' \leftarrow \max(\gamma_B * (1 - \text{transect\_frac}) + \gamma_{up} \times K \times \text{transect\_frac}, \gamma_B)$
end if
$\gamma \leftarrow \gamma'$

However, the actual fraction of the distance travelled was estimated by the number of track lines in the lawnmower pattern that have been completed. In effect the algorithm attempts to ensure that at least the number of suggested AIMs are deployed by the end of the transect. As it gets closer to the end of the transect the threshold will be lowered towards the minimum allowed threshold, γ_{min}. However, this approach is focused more on ensuring the AIMs are deployed, and less on ensuring that it has successfully detected changes in the distribution that is driving the sensors.

The reliance on an *a priori* threshold, in the case of the static policy in (Ferri et al., 2010) means that the vehicle cannot adapt itself to operations in environments that have not been previously characterized. Ferri et al. (2010) proposed their adaptive algorithm addresses this by modifying the threshold in response to the data observed.

Comparing their adaptive algorithm to the fixed algorithm they classified the AIMs they deployed into one of three categories: AIM with confirmed vent, AIM with likely nearby vent, and AIM with no vent. They also counted the number of vents that were missed by the algorithms, the performance of which is summarized here in Table 4.1 and found on p27 of (Ferri et al., 2010).

Table 4.1: The success of the AIMs deployed by the adaptive and fixed threshold algorithms. Also reported the number of missed vents. Numbers come from Table 2 on p27 of (Ferri et al., 2010).

	Adaptive Threshold	**Fixed Threshold**
Confirmed Vents	11	9
Likely Vents	5	4
No Vent	2	3
Missed Vents	2	4

If we include all likely vents in the count of successful vent discoveries, we can fit beta distributions to the precision and recall of the algorithms. We compute precision by dividing

the number of true positive identifications of vents by the sum of the true positives and the false positives. We compute recall by dividing the number of true positive identifications by the sum of the true positive identifications and the missed vents.

From this analysis, summarized in Table 4.2, we find the degree of belief in the improvement does not meet a 95% confidence level, but is still quite large at 84% probability of improvement. From a "gambler's ruin" perspective, one should overwhelmingly choose the adaptive threshold over the fixed. The effect size of the improvement from using the adaptive algorithm is large, with a Cohen's $d = 0.99$.

Table 4.2: A Bayesian statistical analysis of the improvement due to the adaptive threshold.

	Adaptive	**Fixed**	$P(\text{Adaptive} > \text{Fixed})$	**Cohen's** d
Precision	0.61	0.56	84%	0.99
Recall	0.85	0.69	84%	0.99

It should be noted that the behaviour of the algorithm proposed by Ferri et al. (2010) is dependent on a number of parameters determined by scientists in the field. There are opportunities to fine tune this algorithm to be more or less aggressive in how it deploys AIMs, but it also requires knowledge of the environment in the form of the default threshold, the minimum threshold, and the suggested number of AIMs to be deployed.

The work developed by Girdhar focuses on processing images *in situ*. The images that are collected by the robot are subjected to topic analysis to create meaningful clusters of images. As the robot explores, it scores the perplexity of the scene, or in the case of (Girdhar and Dudek, 2016) sub-regions of the scene. Topic perplexity in this case is a measure of how well the image is described by any one of the topics already discovered in the image database. When the topic perplexity is high for an image, the vehicle slows down in order to collect more observations of the perplexing scene. These vehicles do not plan with respect to overall mission objectives or goals, but simply employ their resources greedily.

There is an analogy between topic perplexity and the entropy of an image. Girdhar *et al.* model images as Bag of Words (BoW) vectors. Their topic content can be viewed as the dot product between the topics BoW vectors and their individual image BoW descriptions. The product between the images and all the topic models produces a distribution over the topics. An image that has a high entropy distribution over the topic models represents an anomaly that is not adequately described by the code book developed from the previous observations and as such warrants further investigation. Additional observations get added into the database that are used for producing the topic models, and as such there is an adaptation to the topic model to describe the environment the robot is operating in.

In (Thompson et al., 2013) the focus is on deploying samples when there is a limited sampling budget. Here the explorer is carrying an instrument that is sampling at a predetermined rate and the only thing the robot can do to collect more samples of a phenomenon of interest is to slow the vehicle down. Further, the vehicle is forbidden from back-tracking. The focus of the work was constructing a non-stationary Gaussian Process kernel function in order to have the vehicle adapt to anomalies on-line. That being said, generating two distributions to model the data, one stationary and one not, could be used as the competing hypotheses that are considered in this work, should one need to make a discrete decision.

In the case of using an instrument such as a neutron spectrometer, as in this chapter, there is no reason to believe that there will be an explicit limit on the number of samples that can be collected. However, there may be a limit on overall mission time and/or power resources that can be deployed at any given time. Additionally, unlike (Thompson et al., 2013), our algorithm needs to make a decision to deploy a discrete action, and not control a parameter in a continuum.

Lee et al. (2018) engage in the automatic localization of gamma-ray sources using a single ground vehicle. The experimenters had the robot following one of three different paths, a straight line, a spiral, or a lawnmower path, and used a Compton gamma camera to localize the sources of radiation. The Compton camera is capable of reporting three-dimensional data about incident gamma-rays, and thereby enable localization of sources much more effectively. The work hinges on having a good estimate of the 3D location of the robot and directional information from the Compton camera.

Unfortunately the Mojave Volatiles Prospector and Resource Prospector projects only have access to a point sensor in the form of the NSS. Having such an instrument for neutrons would be highly valuable, and would spur an interesting extension of this work - automatically detecting changes in the number of neutron sources with confidence.

Wilson and Williams (2017) present an approach for designing trajectories for surveying an environment to maximize the observations of values, in their case depth, within a certain range. The algorithm operates within an prescribed boundary, and uses a Gaussian process to model the depth observed by the robot. The robot then follows the gradient of the depth model until it intersects with the boundary. Then the robot follows the contour of the depth measurements until it has identified a closed region within the proscribed boundary, and then decomposes that region into polygons that are searched using a lawnmower pattern.

The approach developed by Wilson and Williams (2017) uses the Gaussian process to reactively change directions in order to identify the region to be decomposed. After the decomposition is designed the robot then deploys the lawnmower pattern without further reactive planning. This approach could be used to design the trajectories that the robot in our algorithm is following, at the overall mission level.

The problem setting discussed in this chapter does not permit control of the robot's path, the robot can simply deploy short-term deviations from the path. In settings where the robot can control its global trajectory, planning more intelligent trajectories is likely to be more effective. Below we discuss ergodic trajectory planning and trajectories designed by Bayesian optimisation.

Ergodic trajectory planning uses an existing density function over a closed region. The objective of the planner is to produce trajectories which cover regions of a space in proportion to the density function over that space (Mathew and Mezić, 2011).

Mathew and Mezić (2011) developed a greedy planner for multiple robots in order to achieve uniform coverage of an enclosed space. This approach was extended to plan optimal trajectories with finite time horizons by Miller and Murphey (2013b) over the course of three papers (Miller and Murphey, 2013b,a; Miller et al., 2016).

Miller and Murphey (2013b) introduce an optimal trajectory planner to produce trajectories, $x(t)$ that are ergodic with respect to some distribution over space, $\phi(x)$. The planner works with continuous time trajectories, and does not require a discretization of the operating domain. This is achieved by transforming the density map the robot is investigating into the frequency domain.

The ideal trajectory is scored by examining the difference between the coefficients of the Fourier decompositions of the trajectory and the density function. This metric is given by Eq. (4.1).

$$\mathcal{E} = \sum_{k=0}^{K} \frac{1}{(1+|k|^2)^{\frac{n+1}{2}}} |c_k - \phi_k|^2 \tag{4.1}$$

Where c_k is the k^{th} Fourier coefficient of the time-averaged trajectory of the robot. ϕ_k is the k^{th} Fourier coefficient of the distribution. The ergodicity metric is summed over the number of Fourier coefficients, K, determined before planning time, and n is the number of dimensions of the region that is being explored by the robot.

Minimizing this metric ensures that the trajectory of the robot has approximately the same spatial distribution of the distribution $\phi(x)$. The higher-level terms are discounted by the square of the component number, meaning higher frequency terms have less of an effect on the planned trajectory.

In (Miller et al., 2016), Miller *et al.* develop the planner in the face of incomplete information. Here the algorithm starts with an estimated spatial density $\phi(x)$, and executes a trajectory for a time horizon, T. Then the algorithm updates the map and plans a new trajectory. Miller *et al.* note that the ergodic planning objective would be compatible with a fast re-planning algorithm and could react to sensor readings as they are observed.

The advantage of the ergodic objective function may initially appear unclear, compared to simply planning a trajectory that maximizes the integral of the trajectory through the density function. What ergodic planning can reject, without additional constraints, is a trajectory that travels to maximum value locations and never leaves. In this way the ergodic metric drives trajectories to explore more of the environment while focusing on high-value locations.

Bayesian Optimization relaxes the requirement to have an initial distribution. Bayesian Optimization algorithms find maxima in a function while exploring the space to determine where those maxima are located. An approach which employs Bayesian optimisation is presented by Marchant et al. (2014). They use the Bayesian optimisation approach to find the maxima in an unknown function over a two-dimensional space. They apply their method to static and time-varying functions. They have a robot travelling at a fixed speed, and with a fixed sampling rate.

They speed up the solution to their POMDP problem by using the Monte Carlo Tree Search (Abramson, 1987). The operations of the robot in question is talked about in terms of days, and it takes the algorithm approximately 8 days to learn the dynamic function, which itself has a period of 1 day.

The work reported in (Marchant et al., 2014) was conducted in simulation. Morere et al. (2017) implement the BO-POMDP algorithm with a UAV mapping an indoor environment. Because of the increased efficiency of the MCTS algorithm they are able to re-plan quickly on a commodity computer. Motion plans consist of a fixed number of cubic splines, the robot collects samples while executing the trajectory. When the trajectory endpoint is reached a new trajectory is planned, incorporating the most recent observations.

In the work of Morere et al. (2017) there is no intra-trajectory reaction to the samples collected. But as the capacity to re-plan increases, approaches like those described by Morere et al. (2017) will obviate the need for algorithms like the one proposed in this chapter, if only to trigger

replanning. While the algorithm only considers planning trajectories and not deploying distinct sampling actions, this is something that could be formulated into a POMDP framework, and benefit from the efficient planner they employ.

In the situation dealt with in this chapter, the robot is making decisions about deploying discrete actions (Area of Interest Manoeuvres) to observe more of the subsurface water density. The objective in deploying these AIMs is to observe maxima in the subsurface water density, achieved by spending more of the robot's trajectory over the high density regions.

An ergodic trajectory planner would naturally spend more of its trajectory in high value regions, removing the need for an explicit representation of the discrete AIM actions. However, this plan would not translate into other types of discrete actions, such as collecting subsurface samples by drilling. Further, the trajectories produced tend to produce back-tracking and oscillation between peaks in the density map, which is not amenable to the Resource Prospector concept of operations. For a comparison of an ergodic planner to the SPRT-based method discussed in this chapter, see Section D.6.

Bayesian optimization approaches are capable of dealing with situations where the underlying function is either known or unknown. Like ergodic planners, Bayesian approaches attempt to maximize observations of high value regions. The planners like those described by Morere et al. (2017) have the capability to reason about discrete actions like drilling or deploying AIMs. Arguably this would make them a more extensible tool for mission planning than the ergodic planner.

When the ergodic planner does not have knowledge of the density map it needs to execute multiple passes to learn the density map. This is also true for the Bayesian optimisation planner.

When prior data about the terrain being explored is limited or unavailable the ergodic and Bayesian optimization planners will produce trajectories with, possibly substantial amounts of, backtracking. A strategy which requires multiple passes in order to plan paths is at odds with the mode of operations of the Resource Prospector mission, where backtracking is actively discouraged due to the mission time pressures. However, these tools could be useful in designing the Area of Interest Manoeuvres, instead of relying on fixed patterns of motion.

Given that the ergodic planner is simply an optimization routine around the ergodic reward function (Eq. (4.1)), it could be integrated with the Bayesian optimization approach. It remains to be investigated the utility of combining an ergodic reward function with a Bayesian optimisation planner.

Since radiation tolerant hardware lags the performance of commodity hardware by at least a decade, there is still a need for fast-executing reactive behaviour while the robot is executing plans designed off-board the robot. While they can't necessarily be executed on-board current space rated hardware, even today algorithms like those described in (Wilson and Williams, 2017), (Marchant et al., 2014), and (Miller et al., 2016) can be immediately useful as ground-based mission planning tools.

(Alcantarilla et al., 2016) present an approach for using convolutional neural networks (CNNs) to do change detections on images sequences captured at two different times. The approach relies on having hand-labelled masks for where change has occurred. They do not construct a notion of confidence for the change.

Adams and MacKay (2007) present a Bayesian approach to change detection. Their model attempts to estimate the run length of different distributions which are driving the observations

presented to the algorithm.

With every new observation they update the belief that the run of observations from the same distribution has continued. Their baseline algorithm has a computational complexity that is linear in the number of observations collected. They propose an improvement of throwing away history prior to the previously detected change point.

Throwing out data prior to a change mirrors the behaviour of our algorithm, which focusses on only detecting one change at a time. Our algorithm, however, has a computational complexity that grows with the square of the number of observations.

Two things distinguish our work from (Adams and MacKay, 2007). First, we estimate the confidence that a change has occurred, whereas Adams and MacKay (2007) only report the probability of the change. Second, the algorithm of Adams and MacKay (2007) is an online algorithm that updates the probability of the run length based only on the current state (a Markovian assumption) and a fixed transition distribution. At the cost of greater computational complexity, our algorithm looks at the relative evidence for one change vs none, fitted to all observed data, we do not make assumptions about the transition probability distribution.

Thompson et al. (2008) present a means of selecting subsets of images for downlink to remote scientists. They used a hidden Markov model to track the probability that the robot is in one of two terrain types. By accumulating evidence from multiple observations it is able to estimate which class of terrain the robot is currently occupying, and hence identify when changes have occurred. Using this knowledge the robot is able to send representative datasets which bracket the change point in the underlying distribution.

Like (Adams and MacKay, 2007), Thompson et al. (2008) puts a prior on the transition of one regime to another. Their transition distribution was tuned to the datasets to produce ideal behaviour. While this is not necessarily amenable to operation in unknown environments, this distribution could conceivably also be learned online.

Hidden Markov Models are an approach that could be employed in our setting as well, however, instead of having two terrain classes we could have up to an uncountable number of states. These states would be the parameter(s) of the distribution driving observations collected by the robot.

One could discretize the parameter space to reduce the number of states that would need to be updated. Further, if the parameter space can be bounded, then the computational complexity of the update at each time step can likewise be bounded. However, this approach wouldn't directly address the question of when the change point occurred with a measure of confidence.

Likewise, as in (Adams and MacKay, 2007), this work does not directly give a measure of confidence that a change has occurred. To detect a change, one could start with an initial parameter estimation, λ_i, and use the Sequential Probability Ratio Test for each parameter setting $\lambda_j \in \Lambda,\ j \neq i$. After $P(\lambda = \lambda_i | Z = z_t)$ is updated, given an observation z_t, we could determine a change has occurred if $\max_{j \neq i} \sum_{t=1}^{T} \frac{P(\lambda=\lambda_j|Z=z_t)}{P(\lambda=\lambda_i|Z=z_t)}$ crosses a pre-determined level of confidence.

Scargle (1998) presents the Bayesian Blocks algorithm for detecting changes in Poisson data. Like the algorithm presented in this chapter, it uses the ratio of two likelihoods to determine if a change has occurred. They explicitly place a prior on the change point and the rates of the Poisson distributions driving the observations.

Their approach integrates over the possible driving rates of the distributions, similarly, we

use the Maximum Likelihood Estimator to fit the parameters for the Poisson processes driving observations. Scargle (1998) determine the likelihood of there having been a change in the underlying distribution as being the average likelihood over all change points. In contrast, we consider each point individually, and select the maximum likelihood of all change points.

The threshold for detecting a change was based on whether the likelihood of a change, relative to no change, crossed a threshold, ρ. ρ was defined as a prior odds ratio which discouraged detecting changes. They observe that their choice for ρ may not be justifiable in a Bayesian formalism, but they do report good results. They do not explicitly mention likelihood ratio tests (e.g. (Wald, 1945)) in the description of their formalism, however, that is precisely what they have developed.

Steyvers and Brown (2006) present an efficient method for detecting changes. They test a Monte Carlo Markov Chain (MCMC) sampler as well as their proposed "Fast and Frugal" change detection algorithms. Steyvers and Brown test the ability of these algorithms to model human change detection behaviour, and find the performance acceptable..

In this experiment the t^{th} observation, y_t, contains a bundle of Bernoulli results, i.e. $y_t = \{y_{t,1}, \ldots, y_{t,D}\}$. Here $y_{t,i} \in \{0, 1\}$, and D is the number of trials in an observation.

The fast and frugal change detection algorithm collects a window of at most M data points and fits a Binomial distribution to those data, which we refer to as $f_A(\cdot)$. An alternative distribution is fit to the new observations when it is collected, which we will call $f_B(\cdot)$.

Changes are detected when the ratio between the likelihood of the latest observation is more likely given the distribution f_B, than the likelihood of the data given the distribution fit to the previously observed data, f_A. They place a threshold C which is tuned in experiments to better model the decision making behaviour of the humans in their experiments.

$$\log\left(f_B(y_t)\right) - \log\left(f_A(y_t)\right) > C \tag{4.2}$$

The decision making rule used in (Steyvers and Brown, 2006) is given in Eq. (4.2). Once a change has been detected, the window of M observed data points is emptied, and new observations are added to it until another change is detected. While in the paper they frame it as a difference of the log of two likelihood functions, it is equivalent to a likelihood ratio test, which the work in (Scargle, 1998) and in this chapter are also examples of.

Scargle (1998) report that the likelihood of there being a change in the underlying distribution is "astronomically large in favor of segmentation". This may be due to the fact that in practice, they omit the probability of the individual change points and compute a sum of the likelihood of the change points. Since they assumed a uniform prior on the change points, these two quantities were equivalent.

The Bayesian Blocks algorithm was then extended in (Scargle et al., 2013) to use a non-parametric approach. They found their approach was able to detect changes in a number of data sets, and also detect multiple changes. The non-parametric approach let them deal with a wider variety of data types, and they proposed additional work with higher-dimensional data.

Worpel and Schwope (2015) use the Bayesian Blocks algorithm to detect transient events in X-ray astronomy data. They found that the approach reduced the error in transient detection.

At first glance, situation addressed in (Ferri et al., 2010) is a perfect application for algorithms that solve the secretary problem. The purpose of the secretary problem is to give the decision

maker some data on which to build a distribution, and only seeks to make one decision instead of taking multiple actions, and does not consider if the underlying distribution is changing.

The secretary problem is a problem setting from statistical literature (Ferguson, 1989) which was constructed to determine when was the optimal time to stop collecting samples. The principle set up was this: A person is trying to hire a secretary. They are presented with a sequence of N candidates to consider for the position. Each candidate has a knowable score, relative to all the other candidates which can be determined by interviewing the candidate. The interviewer then can either choose to hire the candidate or dismiss them and go on to the next candidate. Once a candidate has been dismissed they cannot be recalled. The problem is to determine when is the optimal time to stop interviewing candidates. It was determined that the optimal solution for this problem was to interview the first N/e candidates, where e is Euler's constant, and reject each one in turn. The interviewing process stops either when the first candidate who scores as well as or better than the best candidate in the first N/e candidates, or take the last candidate. This ensures finding the best candidate 37% of the time.

In principle this decision rule makes sense: Determine the distribution of the scores of candidates, then find an outlier with a high score. This problem does line up nicely with automatic AIM execution, the sensor readings from our proxy sensor maps nicely to the score of the candidates. As the robot continues to move we are encountering the sensed phenomena much like the sequence of candidates.

Work like that described in (Das et al., 2015) applies what is called the submodular secretary problem to deploying a fixed sampling budget across a transect. The submodular secretary problem deploys k samples across a transect by dividing it up into k segments, and running a standard secretary algorithm in each segment. This approach does come with the assumption that the value of all the encountered sampling opportunities come from the same distribution.

Distributing samples uniformly across sub-regions of an environment is a principled approach. But due to the nature of secretary algorithms each sample must be deployed on its region of the transect. This could result in ineffectual deployment of resources, should some regions of the environment have a substantially lower amount of the quantity of interest.

Since the objective of Das et al. (2015) is to learn the relationship between the encountered environmental characteristics, that is a reasonable assumption. This approach does not necessarily translate to our setting. We know the relationship between the proxy sensor readings and the quantity we are interested in sampling, and we understand that the underlying distribution is changing as we traverse the environment.

There are other reasons why the secretary problem setting is not an exact match for our proposed problem. First, it does not consider noise in the sensor. However, this objection could be overcome through the use of sensor filtering.

Second, the candidates are considered to be drawn identically and independently from the same distribution. In our setting there can be strong spatial effects on the population from which sensor readings are derived. That is to say, near a location of a water deposit, the readings will come from a distribution with a higher mean than those from farther away from the water deposit.

Third, it assumes that there is no cost to continuing to the next candidate, and that candidate cannot be recalled. This is not the case for autonomous exploration, moving between locations takes time, and any traverse comes with risk associated with the execution of tasks in an unknown environment. While there are algorithms for secretary problems that permit recalling past

opportunities, such as the work in (Rocha, 1993), these do not address changes in the distribution.

The strong spatial dependence creates a need to not only determine how many distributions there are, but also to determine when the vehicle should be confident that it is operating in either one mode or the other. Just modelling the distribution alone is not sufficient, as simply executing AIMs when the probability of being in one regime is higher than the others could result in excessive deployment of AIMs and an unnecessary use of mission resources.

In summary, there are applications for robots that need to determine when to deploy expensive activities, namely AIMs. To date the algorithms deployed are either greedily ignoring resources, using arbitrary and non-adaptive thresholds, or simply control speed of a vehicle proportionally to perceived anomalies without considering if there is statistical confidence that an anomaly has been observed. This chapter proposes an adaptive algorithm that determines when to deploy resources in response to statistical confidence that there has been an event that warrants observation.

4.3 Method

We consider three algorithms for detecting changes in the distribution driving the observations in the proxy sensor, which are described in Section 4.3.1. There are four experiments that are used to test the algorithms. The first two experiments test the effect of the magnitude of the change and the delay in the onset of the change in the distribution driving the observations from the proxy sensor. The third experiment tests the algorithms on data collected as part of the MVP project. The fourth and final experiment simulates the performance of the algorithms in deploying AIMs while operating in a two-dimensional environment. We also compare the performance of the proposed algorithm to an MCMC sampling model, and find the execution time does not warrant further investigation at this point. Otherwise, the results of the experiments are presented in Section 4.4

4.3.1 Algorithms

We used three algorithms in this experiment. The state of the art algorithm, as used in (Ferri et al., 2010) is to issue an AIM whenever a reading crosses a threshold. We modify this algorithm slightly to make for a fairer comparison to the proposed algorithm. The threshold algorithm with memory described in Section 4.3.1 is the baseline algorithm that is employed in the experiments of this chapter.

We employ a second baseline algorithm, which detects a change if the probability of the current observation is not within the 95% high confidence interval of a Poisson distribution based on previous data. This algorithm permits relative changes in the signal to be detected, and should be able to identify changes that occurred below the threshold of the Memory Threshold algorithm.

Additionally, in Section 4.3.2 we compare the proposed algorithm to a similar MCMC based approach. Although performance is comparable to the proposed algorithm, and the implementation of the algorithm could be considered more intuitive than the proposed algorithm, we find that the time to compute solutions is significantly higher and does not warrant that adoption without additional hardware speed-ups.

Detecting Changes with a Threshold

This algorithm compares every observation z_t to a threshold, γ and if $z_t > \gamma$, or conversely if $z_t < \gamma$ then a change is considered to have happened. This algorithm is memoryless and only considers one observation at a time, and each in isolation.

When the process being observed is well characterized this is a useful algorithm. However, the threshold needs to be tuned to every different environment that a robot is operating in. This makes the algorithm very brittle, but it remains a state of the art algorithm.

An analogous version of this algorithm is given a probability distribution, consider a change has occurred when a reading has a probability $P(z_t|\theta) < \delta$ then consider a change has occurred. For every such setting of δ there is a corresponding γ to threshold the sensor values. Again, this algorithm has no memory, and once one starts trying to accumulate data in favour of a change, we get into the proposed algorithm, the Sequential Probability Ratio Test.

Algorithm 4.2 Threshold Change Detection. This algorithm reports that a change has occurred whenever the observed value, z_t, is greater than the threshold, γ. Because the algorithm has no memory, it simply reports that the change has occurred at the current time step.

```
function INIT_THRESHOLD_CHANGE_DETECTION(γ)
    γ ← γ
    t̂_cp ← ∅
end function
function DETECT_CHANGE(z_t)
    if z_t ≥ γ then
        t̂_cp ← t
    else
        t̂_cp ← ∅
    end if
    return t̂_cp
end function
```

This algorithm is in line with the approach taken by Ferri et al. (2010).

Detecting Changes with a Threshold and Memory (Baseline Algorithm)

With little additional effort it is possible to have a slightly smarter algorithm based on thresholding the observed data. We add additional memory such that the first time an observation crosses the threshold, γ, we mark the current time as the start of the change. The algorithm reports the change point as being the time of the first observation that has crossed the threshold until the observation goes below the threshold again. This approach is coded in Algorithm 4.3.

Memory of when a change first occurred helps stabilize the algorithm's estimation of when a change has occurred. Without memory the algorithm will think that every time the threshold has been crossed is a new instance of a change in the underlying distribution.

Algorithm 4.3 Threshold Change Detection with Memory. This algorithm reports that a change has occurred at the first observed value, z_t, is greater than the threshold, γ. Once the observed value goes below the threshold the change period is reset. This prevents the algorithm from constantly reporting that a new change has occurred even though it has been consistently occurring for a number of time steps.

function INIT_MEMORY_THRESHOLD_CHANGE_DETECTION(γ)
    $\gamma \leftarrow \gamma$
    $\hat{t}_{cp} \leftarrow \emptyset$
    in_change $\leftarrow false$
end function
function DETECT_CHANGE(z_t)
    if $z_t \geq \gamma \wedge \neg$ in_change **then**
        $\hat{t}_{cp} \leftarrow t$
        in_change $\leftarrow true$
    end if
    if $z_t < \gamma$ **then**
        in_change $\leftarrow false$
        $\hat{t}_{cp} \leftarrow \emptyset$
    end if
    return $\hat{t}_{cp}$
end function

Detecting Changes with Adaptive Threshold (Baseline Algorithm)

The approach in (Ferri et al., 2010) had two major aspect: First they recorded the anomalousness of observation, and used an adaptive threshold. With this algorithm we employ a threshold adaptation mechanism similar to, but not identical to, that employed by Ferri et al. (2010). We give our implementation of the algorithm in Algorithm 4.4.

An important part of the algorithm is the identification of patches. For our purposes a patch is created when counts cross over the lower bound on the threshold, γ_{min}. Observations over γ_{min} are added to the patch as they are observed. If observed values drop below γ_{min} for more than N_{patch} contiguous observations then the patch is closed. Whenever a patch is closed the threshold is updated.

Algorithm 4.4 Adaptive Threshold Change Detection with Memory. This algorithm reports that a change has occurred at the first observed value, z_t, is greater than the threshold, γ. Once the observed value goes below the γ for more than N_{patch} time steps, the change period is reset. This prevents the algorithm from constantly reporting changes due to momentary fluctuations around γ. Patches are closed when observations go below γ_{min}.

function INIT_ADAPTIVE_THRESHOLD_CHANGE_DETECTION($\gamma_B, \gamma_{min}, N_s$)
    $\gamma \leftarrow \gamma_B$
    $\gamma_B \leftarrow \gamma_B$
    $\gamma_{min} \leftarrow \gamma_{min}$
    $\hat{t}_{cp} \leftarrow \emptyset$
    in_change,in_patch $\leftarrow false$
    patches $\leftarrow \emptyset$
    current_patch $\leftarrow \langle\cdot\rangle$
    patch_count, $N_{AIM}, N_s, N_{patch} \leftarrow 0$
end function
function DETECT_CHANGE(z_t, s) ▷ detect_change requires,s, % of transect completed
    if $z_t \geq \gamma \wedge \neg$ in_change **then**
        $\hat{t}_{cp} \leftarrow t$
        in_change $\leftarrow true$
        $N_{AIM} \leftarrow N_{AIM} + 1$
    end if
    if $z_t < \gamma$ **then**
        in_change $\leftarrow false$
        $\hat{t}_{cp} \leftarrow \emptyset$
    end if
    if $z_t \geq \gamma_{min}$ **then**
        patch_count $\leftarrow 0$
        current_patch $\leftarrow$ current_patch $\cup \langle z\rangle$
        if $\neg$ in_patch **then**
            in_patch $\leftarrow true$
        end if
    else
        patch_count $\leftarrow$ patch_count $+1$
        if in_patch $\wedge$ patch_count $\geq N_{patch}$ **then**
            in_patch $\leftarrow false$
            patches $\leftarrow$ patches $\cup \{$current_patch$\}$
            current_patch $\leftarrow \langle\cdot\rangle$
            update_threshold(s)
        end if
    end if
    return $\hat{t}_{cp}$
end function

The algorithm takes as parameters a default threshold, γ_B, a minimum threshold γ_{min}, and a suggested number of AIMs to be deployed N_s. The algorithm also keeps track of how far along the transect it has completed, and if the number of AIMs deployed is not keeping pace with the fraction of the transect covered, the threshold is adjusted. The algorithm used for modifying the threshold, γ is given in Algorithm 4.5.

Algorithm 4.5 The algorithm compares the number of AIMs that have been deployed to the fraction of the transect that has been completed. If a smaller fraction of the suggested AIMs have been deployed than the fraction of the transect have been completed then the threshold is lowered to encourage a greater number of AIMs to be deployed. If a larger fraction of AIMs have been deployed than transect covered, then the threshold is raised to discourage future AIMs from being deployed.

function UPDATE_THRESHOLD(s)
  if $N_{AIM}/N_S \leq s$ **then**
    low_thr $\leftarrow$ median $(\{\max(p) : \forall p \in \text{patches } s.t.\ \max(p) < \gamma\})$
    $\gamma' \leftarrow \max(\gamma_B \times (1-s) + \text{low\_thr} \times s, \gamma_{min})$
  else
    raise_thr $\leftarrow$ median $(\{\max(p) : \forall p \in \text{patches}\})$
    $\gamma \leftarrow \max(\gamma_b \times (1-s) + \text{raise\_thr} \times s, \gamma_B)$
  end if
end function

In our experiments we set γ_B to be the same value as γ in the Memory Threshold algorithm, and keep the number of suggested AIMs, $N_s = 50$. The minimum threshold, γ_{min} is set to 53 for experiments 1,2, and 4, and set to 45 for experiment 3.

We do not employ the same techniques for identifying anomalous readings as that used by Ferri et al. (2010), consequently our algorithm doesn't detect changes that occur below the minimum threshold, γ_{min}. However, as will be seen in Section 4.4, we get similar improvements in performance from using the adaptive vs a fixed threshold as in (Ferri et al., 2010).

Detecting Changes with Relative Changes (Baseline Algorithm)

The threshold-based algorithms described above only detected changes if observations cross a pre-determined level. An obvious other approach is to examine relative changes of the readings, and if readings cross a boundary then consider that a change has occurred. In this case we take a Poisson distribution and detect a change if the probability of the new reading is outside of the interval that contains δ percentage of the distribution. We set the confidence level to 95% for all experiments in this chapter. The implementation of the algorithm is given in Algorithm 4.6.

Algorithm 4.6 The relative change detection algorithm considers that the underlying distribution has changed. When a change has been detected the new reading is taken as the mean of the new distribution.

function INIT_RELATIVE_CHANGE_DETECTION(δ)
  $\delta \leftarrow \delta$
  $\lambda \leftarrow \emptyset$
  $\hat{t}_{cp} \leftarrow \emptyset$
end function
function DETECT_CHANGE(z_t)
  if $\lambda = \emptyset$ **then**
    $\lambda \leftarrow z_t$
  end if
  if $z_t \notin$ interval$(P(Z), \delta)$ **then**
    $\lambda \leftarrow z_t$
    $\hat{t}_{cp} \leftarrow t$
  end if
  return $\hat{t}_{cp}$
end function

Detecting Changes with the Sequential Probability Ratio Test (Proposed Algorithm)

The sequential probability ratio test considers two hypotheses, h_1 and h_2. Each hypothesis implies some probability distribution about observations that were collected. The sequential probability ratio test asks the question which distribution better explains observed data $z_1, \ldots, z_t$. We determined the relative probability of the data given the hypotheses is:

$$R_t = \frac{P(z_1, \ldots, z_t|h_2)}{P(z_1, \ldots, z_t|h_1)} \tag{4.3}$$

If at any given time $R > 1$ then h_2 is more likely. If $R < 1$ then h_1 is more likely. However, if observations are considered to be independently and identically distributed then we can write the following:

$$R_t = \prod_{i=1}^{t} \frac{P(z_i|h_2)}{P(z_i|h_1)} \tag{4.4}$$

And if we take the log of this quantity we find:

$$\Lambda_t = \sum_{i=1}^{t} \left[\log\left(P(z_i|h_2)\right) - \log\left(P(z_i|h_1)\right)\right] \tag{4.5}$$

$$\Lambda_t = \sum_{i=1}^{t} \log\left(P(z_i|h_2)\right) - \sum_{i=1}^{t} \log\left(P(z_i|h_1)\right) \tag{4.6}$$

The log form of the equations has the advantage of letting us work with sums, which are easier to keep a running total of than products. Since likelihoods can become very small very quickly it is more numerically stable to work with the log of the likelihood function.

Wald determined thresholds α and β based on the acceptable false positive and false negative rates for the test. If $\Lambda_t > \alpha$ then h_2 is accepted and if $\Lambda_t < \beta$ then h_1 is accepted and the experiment stops. For specified levels of error probabilities the SPRT is the test with the minimum expected stopping time (Wald, 1945).

In this chapter h_1 is the hypothesis that no change has occurred and h_2 is the belief that a change has occurred. We consider the hypothesis that a time has occurred at every time point between 1 and t, the current time. We select the most likely time change. If that potential change point crosses a threshold α then we consider that a change in the underlying distribution has occurred.

Fitting two distributions to a data set should always fit the data better than one distribution. The implication of this is that $R_t \geq 1$ and hence $\Lambda_t \geq 0$. So we can ignore checking if we become certain that there has been no change. In fact, because we don't do anything until confidence has built that there has been a change, no further action is required of the robot except to continue driving. The aim is only necessary when we are confident that a change has occurred.

Because $\Lambda_t \geq 0 \,\forall\, t \geq 0$ there is the risk that with enough observations the quantity will eventually cross the confidence threshold, even if no underlying change has occurred. To mitigate this risk we set α to be larger than prescribed by Wald. The task of determining the confidence level at which an AIM should be deployed is left to the people who designed the mission and be a function of their tolerance of risk and the expected duration of the entire transect.

There are a number of assumptions we make here in order to implement this algorithm. First, we assume the data from a given distribution are all independently and identically distributed (IID) and coming from Poisson distributions. We consider that in the case of no change in the underlying distribution that it was drawn from a Poisson distribution with rate θ_0. In the case where there has been a change in the distribution we call the rate before the change θ_1 and the rate after the change θ_2.

Since we do not know θ_0, θ_1, or θ_2, we estimate them by the maximum likelihood estimates from corresponding samples, $\hat{\theta}_0 = MLE(\langle z_0, \ldots, z_T \rangle)$, $\hat{\theta}_1 = MLE(\langle z_0, \ldots, z_t \rangle)$, and $\hat{\theta}_2 = MLE(\langle z_{t+1}, \ldots, z_T \rangle)$, respectively, where T is the current maximum time step.

After an AIM has been conducted the window of samples gets reset to the samples after the change point, t, and the process continues on as before. The implementation of the algorithm is given in 4.7.

Algorithm 4.7 Sequential probability ratio test-based change detection. This algorithm considers all the data that has been collected up to the present time, t, and determines whether or not there has been a change in the underlying distribution.

function INIT_SPRT_CHANGE_DETECTION(s)
    $s \leftarrow s$
    data $\leftarrow \langle \cdot \rangle$
end function
function LIKELIHOOD($z_1, \ldots, z_n, \lambda$) ▷ Computes log likelihood for a Poisson distribution
    return $\log\left(\prod_{k=1}^{n} P_{poisson}(z_k|\lambda)\right)$
end function
function DETECT_CHANGE(z_t)
    data $\leftarrow$ data $+ \langle z_t \rangle$
    $l_0 \leftarrow$ likelihood(data, $mean$(data))
    likelihoods $\leftarrow \langle \cdot \rangle$
    for all $k \in \{1, \ldots, t\}$ **do**
        $\lambda_1 \leftarrow mean(z_1, \ldots, z_k)$
        $\lambda_2 \leftarrow mean(z_{k+1}, \ldots, z_t)$
        likelihoods$[k] \leftarrow$ likelihood$(z_1, \ldots, z_k, \lambda_1)$ + likelihood$(z_{k+1}, \ldots, z_t, \lambda_2)$
    end for
    $\hat{t}_{cp} \leftarrow \underset{k \in \{1,\ldots,t\}}{\operatorname{argmax}}$(likelihoods$[k]$)
    if likelihoods$[change_point] - l_0 > s$ **then**
        data $\leftarrow$ data $\left[\hat{t}_{cp}, \ldots, |\text{data}|\right]$
        return $\hat{t}_{cp}$
    else
        return $\emptyset$
    end if
end function

This algorithm simply detects the changes in the underlying distribution. It is up to other mission objectives to make value judgements about the meanings of those values. For instance, observations that contradict remote sensing data may warrant a different level of interaction than data that is variable, but consistent with what is observed from orbit.

We can achieve a $2\times$ speed up in the algorithm by not recomputing the log likelihood of the first $1, \ldots, t$ data points for each t. The naive implementation is given above, for ease of communication.

4.3.2 Execution Time Comparison of SPRT to MCMC-Bayesian Change Detection

A principled method to approach this problem would be to employ an MCMC sampling method to determine if there has been a change in the underlying distribution. Modern approaches would have one construct a probability distribution over the change point in the underlying distribution,

then feed the model data and use, e.g., an MCMC sampling algorithm to determine the probability that a change has occurred for each time step, for example as conducted in (Adams and MacKay, 2007). The MCMC model will produce a distribution over change points. One can use that distribution to select the most likely change point and act upon it, or pass the whole distribution to another decision making algorithm. The setup for such a problem is given as follows:

$$C_t \sim Poisson(r_t); \tag{4.7}$$

$$r_t = \begin{cases} late \text{ if } s < t \\ early \text{ if } s \geq t \end{cases} \tag{4.8}$$

$$s \sim U(1, t_{max}) \tag{4.9}$$

$$early \sim exp(1) \tag{4.10}$$

$$late \sim exp(1) \tag{4.11}$$

Where s is the switch point between the $early$ and $late$ arrival rates. The arrival rates are drawn from an exponential distribution. They, in turn, drive the Poisson distribution that produces the counts observed by the NSS. This model, along with the observed data up until the current time, t_{max}, is estimated with an MCMC sampler.

To illustrate the relative performance of our proposed algorithm vs the Bayesian-MCMC approach in Fig. 4.2 we report the time it takes to execute an update in response to the latest sample for transects with up to 100 samples.

We executed the two algorithms on an Intel Core i7 CPU 860, which has eight cores and a clock speed of 2.8GHz, with 16 GB of RAM. The algorithms were both implemented in Python version 2.7 running in Ubuntu 16.04 LTS. The MCMC algorithm was implemented using the pymc3 library (Coyle, 2016).

The time to complete an update is substantially larger for the MCMC approach than the SPRT algorithm. The average time for the MCMC algorithm to process a sample was 61.1 seconds, the average time for SPRT to process a sample was 0.0719 seconds. The reduction in processing time was approximately 61 seconds, with a 95% High Density Interval (HDI) of $[58, 62]$ seconds and effect size (Cohen's d) of 16.8, which is an very large effect. Because of the increased time execution we exclude MCMC-based solutions.

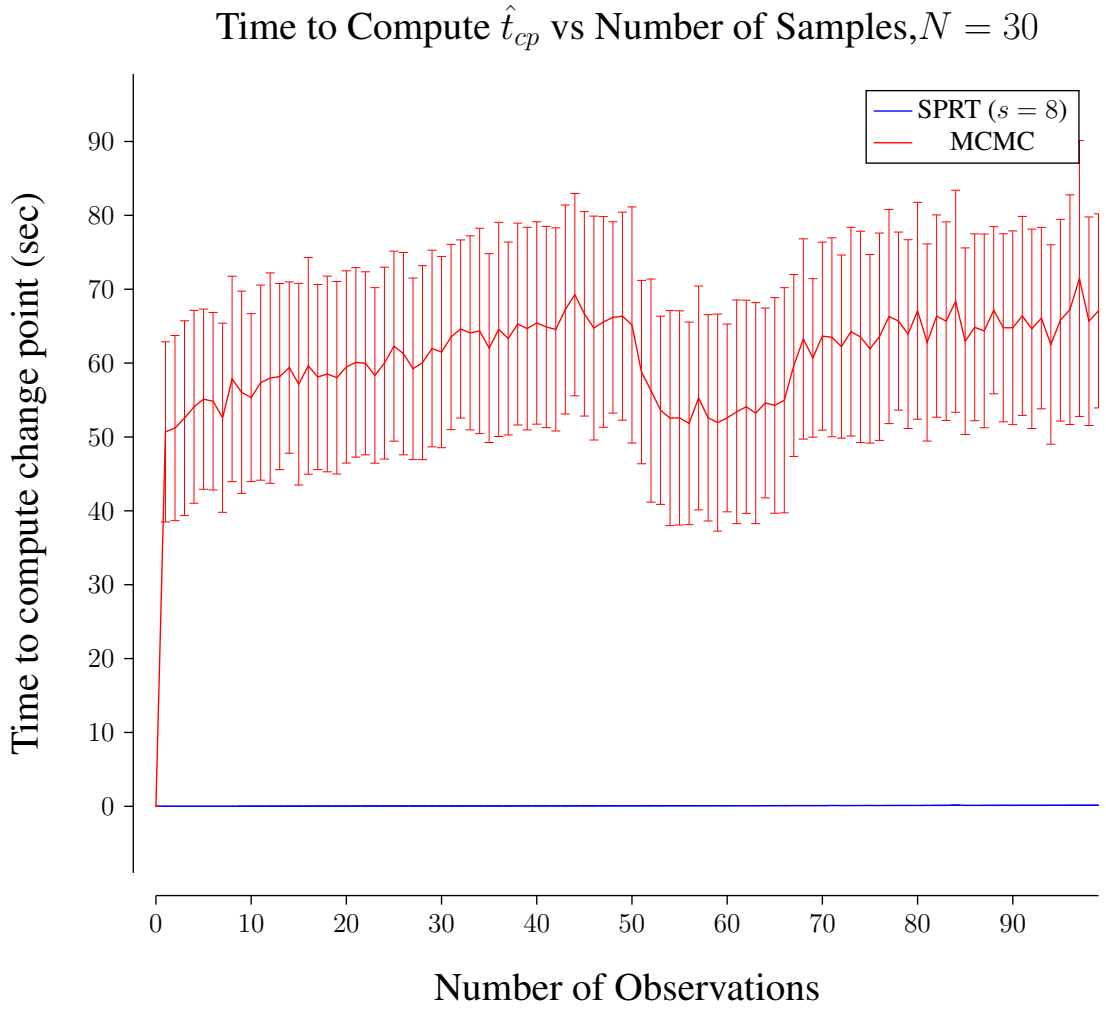


Figure 4.2: The average execution time for the SPRT and Bayesian-MCMC algorithms as a function of the number of observations collected. Error bars represent one 95% confidence interval based on $N = 30$ trials. The SPRT algorithm is on the order of 100x faster than the MCMC algorithm.

We also considered the execution the two different algorithms. We compare the performance of the two algorithms by looking at the error in identifying when the change in the underlying distribution occurred, E_{cp}. We define that error as:

$$E_{cp} = |\hat{t}_{cp} - t_{cp}| \tag{4.12}$$

Where $\hat{t}_{cp}$ is the estimated time of the change and t_{cp} is the actual time of the change. In the ten trials we performed the change all occurred at the halfway point, $t = 50$ and the underlying distribution changed from $\lambda_1 = 40$ to $\lambda_2 = 80$. We looked at the error that occurs before the change in the underlying distribution ($t = 25$), immediately after the time change ($t = 52$), a short time after the time change ($t = 60$) and at the end of the transect ($t = 100$). The change point detection performance is shown in Fig. 4.3. Notice that immediately after the change in the distribution has taken place, the variability of the MCMC algorithm is very large.

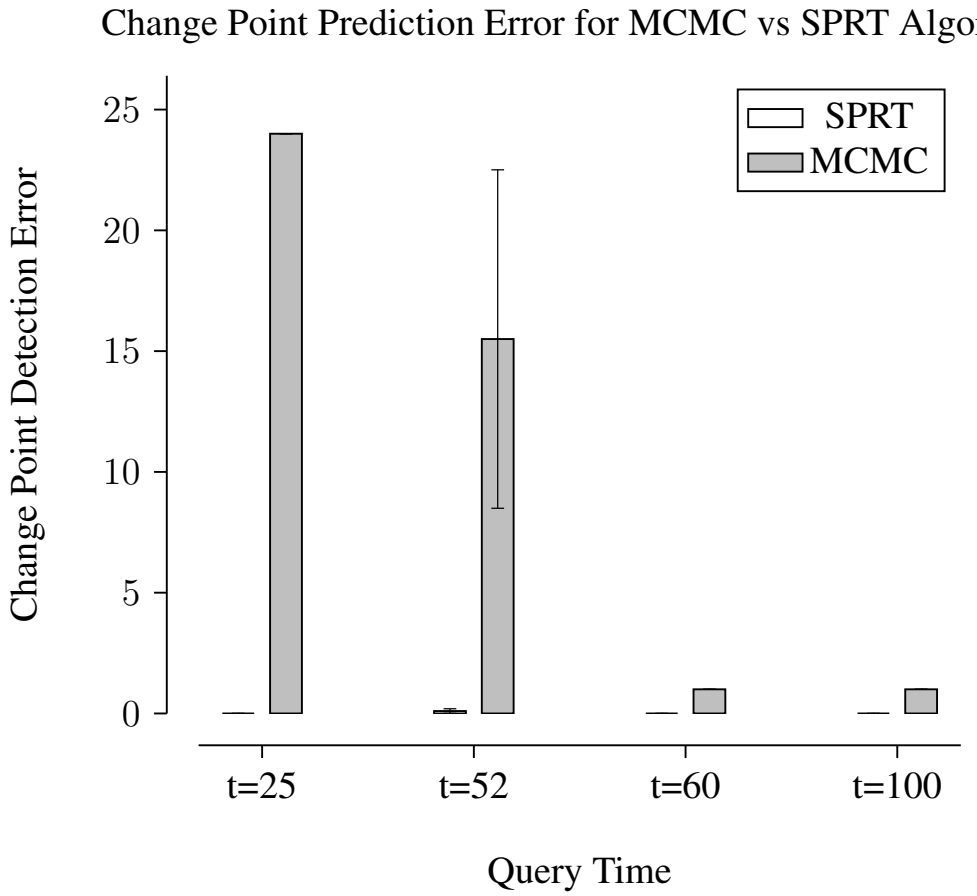


Figure 4.3: Considering the performance of the two algorithms over 10 trials. The performance is measured as the error between the estimated time of change in the underlying distribution and the true change time. We consider their performance before the change in the underlying distribution (t=25), shortly after the change in the underlying distribution (t=52 and t=60) and at the end of the transect (t=100).

Table 4.3 shows the relative performance of the MCMC algorithm and the SPRT algorithms in detecting the change point in the underlying distribution. In all cases the SPRT and MCMC had statistically significantly different performance.

Something we observed is that after the change had occurred at $t = 50sec$ the MCMC approach reported the change happened at $t = 49sec$ while the SPRT algorithm reported that the change occurred at $t = 50sec$. This difference can be viewed as a slightly different interpretation of how to report when the change happened, when is the last time step before the new data is observed or when is the first time step corresponding to the new data. So while the difference between the two algorithms is statistically significantly different, and with a substantial effect size it can be viewed as demonstrating comparable behaviour.

More telling is that before the change has occurred the SPRT algorithm is superior to the MCMC algorithm to correctly reporting that no change has occurred. This is a statistically significant difference with a profoundly large effect size. This alone could be taken as motivation for not using the MCMC approach.

Granted, the MCMC algorithm described here assumes that there is at least one change in the underlying distribution. Making it more complex would enable it to identify that there might be zero changes in the distribution in a scene. However, making the model more complex would not make it less computationally intensive.

Table 4.3: In all cases there is a statistically significant reduction in the error in detecting the change point. Before the time change has occurred the SPRT algorithm is better at detecting there has been no change in the underlying distribution than the MCMC approach. However for all the cases after the change has occurred there is an error of 1 time step, which is a negligible error. We can say that the SPRT and MCMC algorithms have indistinguishable performance after a change has occurred.

T (sec)	$\mu_{MCMC} - \mu_{SPRT}$ **(sec)**	**95% HDI**	**Effect Size (Cohen's d)**
25	24	$[24, 24]$	1.87×10^3
52	1	$[0.97, 1.03]$	48.2
60	1	$[1, 1]$	1.88×10^3
100	1	$[1, 1]$	1.86×10^3

Because the performance of the SPRT algorithm is indistinguishable from that of the Bayesian algorithm after the change, and superior to it before the change in the distribution, and because the time performance is substantially better, we choose to use the SPRT algorithm in favour of the Bayesian/MCMC approach.

4.3.3 Experiments

We test the SPRT algorithm against the Memory Threshold and Relative Change algorithms in four different experiments. All of the experiments revolve around detecting a change from one underlying distribution to another. All experiments consider a sensor with noisy readings that follow a Poisson distribution. The experiments compare the performance of the algorithms:

1. As a function of the magnitude of change in the underlying distribution rate.
2. As a function of the delay in the onset of the distribution change.
3. On real MVP data.
4. On a simulated 2D environment using data drawn from the observations from the MVP project.

In all the experiments a time step was one second. For the Memory Threshold algorithm a threshold of $\gamma = 80$ was used. For the Relative Change algorithm we used a confidence interval of $\delta = 0.95$.

Experiment 1 - Effect of Magnitude of Change in the Underlying Distribution

In the first experiment the agent is presented with sequential data. There is a change in the underlying distribution half way through the simulated transect. The initial rate for the Poisson distribution is $\theta_1 = 40$ counts per second. The second rate is given by $\theta_2 \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

We ran 50 trials of each setting of θ_2. The duration of the transect was $T = 100$ time steps. Algorithm parameter settings are given in Table 4.4

Table 4.4: The parameter settings for the algorithms used in this experiment. γ_{min} was chosen to be 53 because that is the upper end of a 95% confidence interval of a Poisson process with a rate of 40.

Algorithm	Parameters
Memory Threshold	$\gamma = 80$
Adaptive Threshold	$\gamma_B = 80, \gamma_{min} = 53, N_S = 50$
Relative Change	$\delta = 0.95$
SPRT	$s \in \{2, 4, 8\}$

Experiment 2 - Effect of Delay in Onset of Change in the Underlying Distribution

In experiment 2 we kept the two different rates constant with $\theta_1 = 40$ and $\theta_2 = 100$. The duration of the transect $T = 400$ time steps. The time step at which the distribution changed was varied over $t \in \{0.1T, 0.2T, 0.3T, 0.4T, 0.5T, 0.6T, 0.7T, 0.8T, 0.9T\}$. We ran 50 trials for each budget. The algorithm parameter settings are given in Table 4.5.

Table 4.5: The parameter settings for the algorithms used in this experiment. γ_{min} was chose to be 53 because that is the upper end of a 95% confidence interval of a Poisson process with a rate of 40.In this experiment we down selected the SPRT confidence level settings to just $s = 8$. This was done in response to the performance from the previous experiments.

Algorithm	Parameters
Memory Threshold	$\gamma = 80$
Adaptive Threshold	$\gamma_B = 80, \gamma_{min} = 53, N_S = 50$
Relative Change	$\delta = 0.95$
SPRT	$s = 8$

Experiment 3 - Performance on Real-World Data

We tested the algorithms on real-world data collected as part of the MVP project. The objective of the test was to detect changes in the underlying distribution that were annotated by humans. We tested on 9 days of data collected during October of 2014. Algorithm parameter settings are given in Table 4.6.

Table 4.6: Algorithm parameters were the same as in Experiment 2, except the lower threshold for the Adaptive algorithm was lowered to 50, because otherwise the algorithm did not identify changes in a majority of the transects.

Algorithm	Parameters
Memory Threshold	$\gamma = 80$
Adaptive Threshold	$\gamma_B = 80, \gamma_{min} = 50, N_S = 50$
Relative Change	$\delta = 0.95$
SPRT	$s = 8$

To determine the sensitivity of the algorithm performance to different parameters we constructed a Receiver Operating Characteristic (ROC) curve. This reports the true positive and false positive rate of the algorithms. The parameters we used to determine those rates are given in Table 4.7. We selected the initial threshold for the Adaptive Threshold algorithm based on the best performing threshold for the Memory Threshold algorithm.

Table 4.7: Parameter settings used to construct ROC curve. The initial threshold for the Adaptive Threshold algorithm was selected from the best performing setting of the Memory Threshold algorithm.

Algorithm	**Parameters**
Memory Threshold	$\gamma \in \begin{Bmatrix} 10, 20, 30, 33, 35, 37, 40, 43, 45, 47, 50, \\ 52, 55, 57, 60, 70, 80, 90, 100, 110 \end{Bmatrix}$
Adaptive Threshold	$\gamma_B = \{43\}$ $N_s \in \{10, 30, 50\}$ $\gamma_{min} = \begin{Bmatrix} 20, 25, 30, 33, 35, 37, 40, 43, 45, 47, \\ 50, 53, 55, 60, 65 \end{Bmatrix}$
Relative Change	$\delta \in \begin{Bmatrix} 0.85, 0.90, 0.925, 0.95, 0.975, \\ 0.99, 0.995, 0.997, 0.999 \end{Bmatrix}$
SPRT	$s \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

Experiment 4 - Effect on Performance in 2D Operations

Finally, to explore what effect the algorithm would have on performance in a mission scenario we simulated exploration by a rover in an environment modelled on data from the MVP project. We consider the objective to be the same as in (Ferri et al., 2010) - to localize the local maxima of the underlying distribution. We considered a maxima to have been localized if the vehicle drove over that point.

We add another algorithm, **No AIMs**, simply follows the pre-defined trajectory. This was added to be able to gauge an absolute improvement of the baseline and proposed algorithms. The algorithm parameter settings are given in Table 4.8.

Table 4.8: Parameter settings were returned to the values used in experiment 2.

Algorithm	**Parameters**
Memory Threshold	$\gamma = 80$
Adaptive Threshold	$\gamma_B = 80, \gamma_{min} = 53, N_S = 50$
Relative Change	$\delta = 0.95$
SPRT	$s = 8$

We tested the performance of the algorithms on 30 randomly generated maps and compared them against an additional algorithm which simply followed a lawnmower pattern across the landscape. A map of the subsurface water density was produced by randomly placing 1000 samples in a $50m \times 50m$ map and the constructing a Voronoi map from those samples. The map was approximated as a 100×100 cell grid, where cell width was $0.5m$.

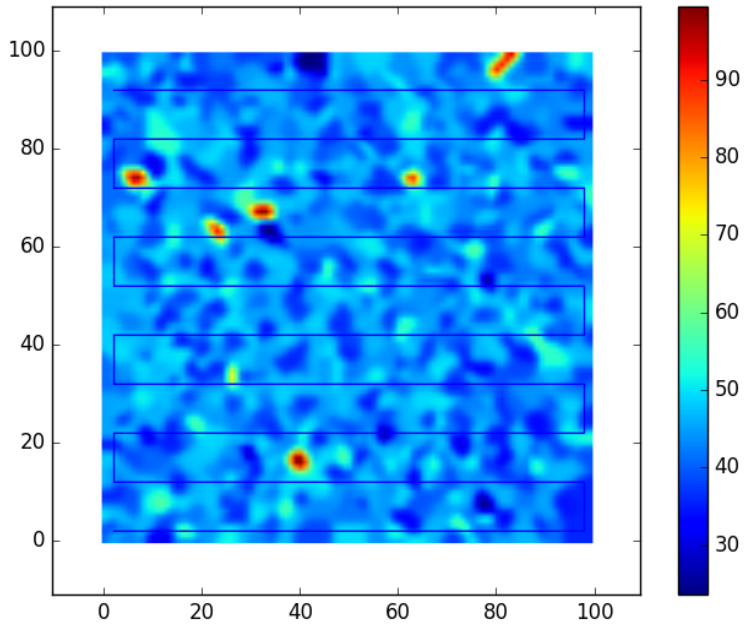


Figure 4.4: Here we see a simulated map with the planned lawnmower path superimposed on it. The rate of the Poisson distribution that samples were drawn from was determined by the location of the vehicle in the map.

In this experiment we did not consider trafficability or obstacle avoidance and simply followed a lawnmower pattern that was 50m wide with 5m spacing between switchbacks. An example of the map with the path superimposed is shown in Fig. 4.4. At any point in time,t, when the rover was at location (x, y) the rate of the Poisson distribution from which observations were sampled was given by $\theta_t = M(x, y)$, where $M(x, y)$ was the value of the map at location (x, y).

As with Experiment 3 we also constructed a ROC-like curve to show the sensitivity of the algorithms to their parameters. Table 4.9 gives the parameters that were explored in evaluating the algorithms' performance.

Table 4.9: Parameters explored to test the sensitivity of the algorithms when operating in 2D environments.

Algorithm	**Parameters**
Memory Threshold	$\gamma \in \left\{ \begin{matrix} 10, 20, 30, 33, 35, 37, 40, 43, 45, 47, 50, \\ 52, 55, 57, 60, 70, 80, 90, 100, 110 \end{matrix} \right\}$
Adaptive Threshold	$\gamma_B \in \{40, 45, 50, 55, 60, 65, 70, 75, 80\}$ $N_s \in \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60\}$ $\gamma_{min} \in \left\{ \begin{matrix} 20, 25, 30, 33, 35, 37, 40, 43, 45, 47, 50, \\ 53, 55, 60, 65 \end{matrix} \right\}$
Relative Change	$\delta \in \left\{ \begin{matrix} 0.85, 0.90, 0.925, 0.95, 0.975, 0.99, \\ 0.995, 0.997, 0.999 \end{matrix} \right\}$
SPRT	$s \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

4.3.4 Performance Metrics

In experiments 1 and 2 we used three performance metrics. The first was the false positive rate. This is the number of time steps before the change in the underlying distribution that were reported as changes. The second metric was the false negative rate. This is the number of time steps after the change in the underlying distribution where no change was reported. The third metric was the mean error in the reported time change. This is the average difference the reported time of change and the actual time of change, for all time changes reported.

In experiment 3 we recorded true positives, false positives, and false negatives. We identified changes in the NSS signal manually, and recorded the ability of the competing algorithms to identify them. We recorded this data for each data set, corresponding to data collected on different days of the MVP field work.

In the fourth experiment we considered the number of local maxima in the map of the water map captured by the vehicle during navigation. This metric is inspired by Ferri et al. (2010), as the objective of the robot was to capture the local maxima of the chemical densities it was tracking. Ferri et al. (2010) placed an upper limit on the number of AIMs that the robot could deploy, which we do not. To remove the number of maxima that would have been observed regardless of how many AIMs were deployed, we subtracted the number of maxima observed by the No AIM algorithm from the performance of all the other algorithms. In order to determine

how effectively the algorithms use the AIMs we consider the ration of maxima observed to AIMs deployed.

4.4 Results

Below we present the results in the experiments described above. In the first three experiments we found that the proposed SPRT based algorithm outperforms the baseline algorithms. In the fourth experiment we find that the Relative Change algorithm successfully captures more maxima in the water map, but we find that it is less productive in its deployment of AIMs than the SPRT algorithm. We compare the performance of the algorithms using pooled Bayesian hypotheses tests for experiments 1-3, and a paired Bayesian hypothesis test in experiment 4. We set a confidence threshold of 95% probability of a difference in performance. When statistical confidence is achieved we report the effect size of the improvement in performance.

In order to understand how the changes are reported by the algorithms we direct the reader's attention to Fig. 4.5. This illustration shows that the change point determined by the Memory Threshold and Relative Change algorithms are unstable. They continue to increase their estimate of the time change in response to noisy observations, and after the time change has occurred. The SPRT with a confidence level set at 2 erroneously detects changes before the actual change in the underlying distribution. The SPRT algorithm with the confidence level set to 8 does not report changes before the actual change has occurred, and after the change the estimate is stable. This behaviour is typical in all the experimental settings.

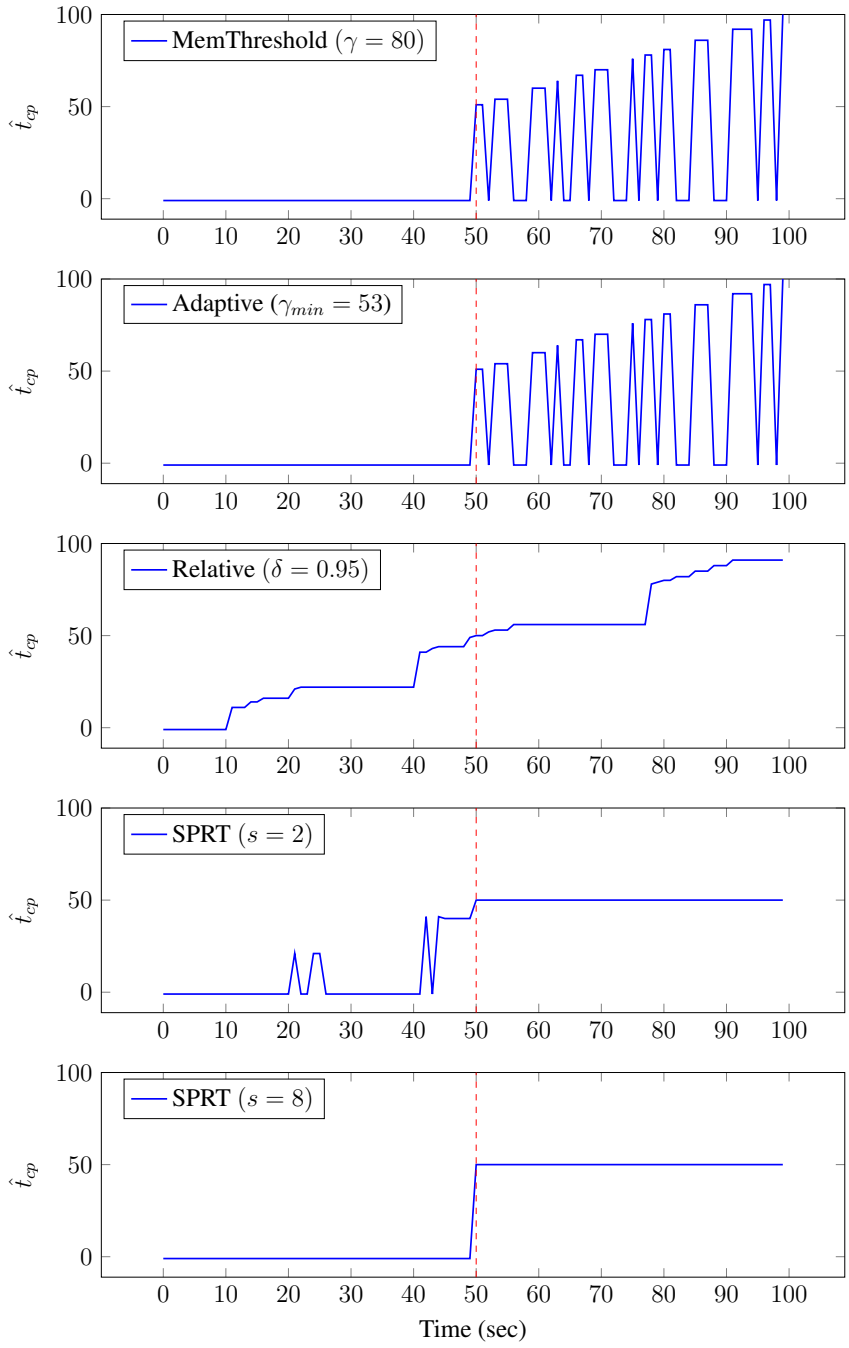


Figure 4.5: The detected change points vs time. Both the Memory Threshold and SPRT $(s = 8)$ report no change before the change has occurred. After the change has occurred both SPRT algorithms have stable predictions of the change point. The Memory Threshold, Adaptive, and Relative Change algorithms have unstable predictions, with the Relative Change algorithm predicting changes before the change actually occurs, which the Memory and Adaptive Threshold algorithms do not.

4.4.1 Experiment 1 Results - Effect of Magnitude of Change in the Underlying Distribution

In Fig. 4.6 we see that with the exception of where there is no change between the two arrival rates the SPRT algorithm detects the change with high accuracy. The Relative Change algorithm has approximately a zero false negative rate, this is because it reports changes with high frequency, regardless of whether or not a change has occurred. We see that the Adaptive algorithm has similar performance to the Memory Threshold algorithm, with a slight improvement over the fixed threshold algorithm.

When there is no change in the arrival rates the SPRT algorithm has a much higher false negative rate. This, however, is desirable, as if there is no change in the arrival rate then we don't want the algorithm to detect a change. In this experiment that it was the SPRT algorithm with a parameter set to 8 that had the best false negative rate.

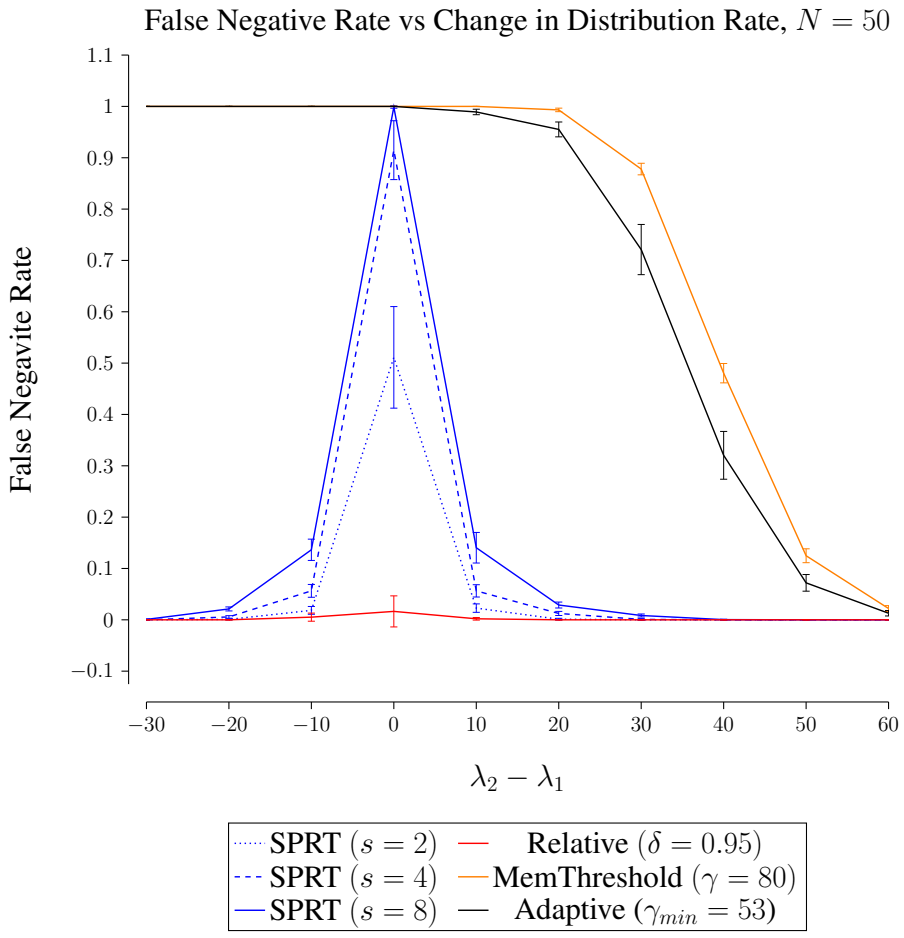


Figure 4.6: Showing the rate of false negatives as a function of the difference between the arrival rates as the confidence parameter of the SPRT algorithm is tuned. A peak at $\lambda_2 - \lambda_1 = 0$ is expected because this represents no actual change in the distribution. Error bars represent a 95% confidence interval, $N = 50$.

If we look at Table D.1,Table D.2,and Table D.3, we can see that all of the SPRT algorithms

perform statistically significantly different from the Memory Threshold algorithm.

In Fig. 4.7 we see that the false positive rate is independent of the change in the distribution driving the sensor readings. It is clear to see that the Relative Change algorithm is performing significantly worse than any of the other algorithms. The SPRT algorithm with a confidence level of $s = 8$ is the best performing SPRT algorithm.

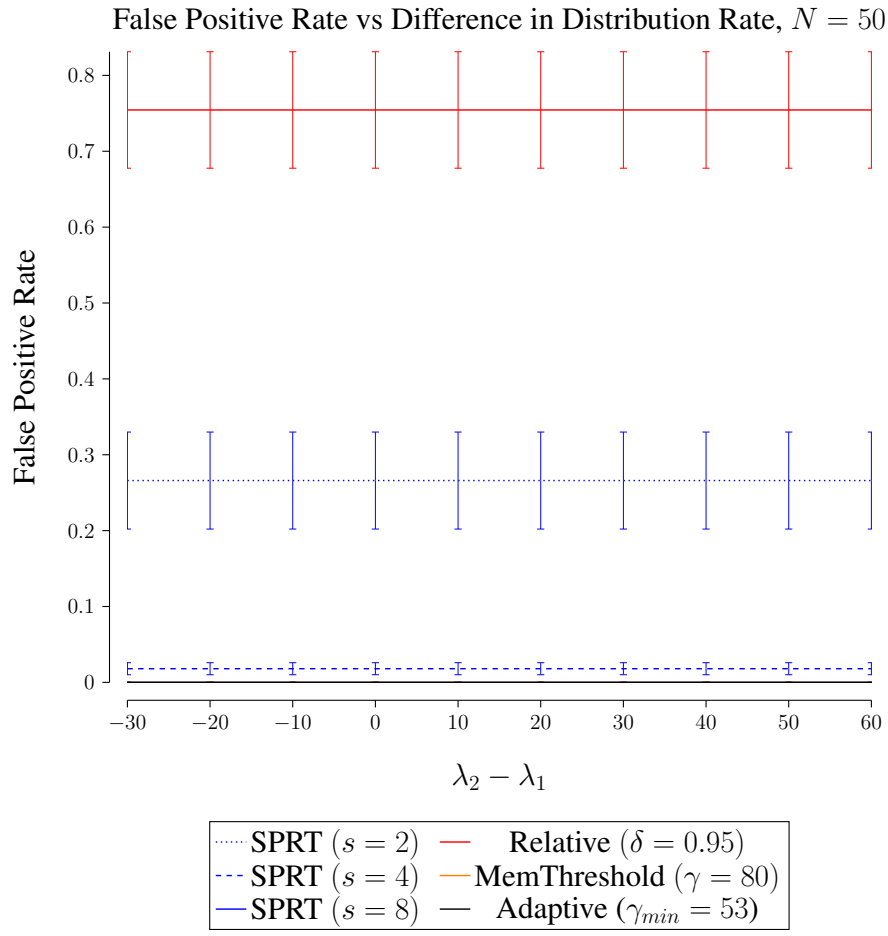


Figure 4.7: Showing the rate of false positives as a function of the difference between the arrival rates as the confidence parameter of the SPRT algorithm is tuned. The SPRT algorithm's performance is independent of the change in background arrival rate. This graph reveals that it is possible to control the false positive rate through the SPRT confidence parameter. Memory and Adaptive Threshold algorithms have identical performance. The Relative Change algorithm has an exceptionally large false positive rate. Error bars represent a 95% confidence interval, $N = 50$.

In Table D.4 shows that the SPRT algorithm with the confidence level set to 2 performs demonstrably worse than the Memory Threshold algorithm. When the SPRT confidence threshold is set to 4 (Table D.5) and 8 (Table D.6) we see that the false positive rate of the SPRT is indistinguishable from that of the Memory and Adaptive Threshold algorithms, and in turn, indistinguishable from 0.

In Figure 4.8 we see that the Memory Threshold algorithm does not correctly identify a

change until the second distribution begins to approach the threshold. By virtue of having an adaptive threshold, the Adaptive algorithm improves on the Memory Threshold algorithm. The Adaptive algorithm was able to detect smaller changes in the driving rate than the Memory algorithm, and it had an improvement on the error in predicting the change. However, like the Memory Threshold algorithm, its estimation of the change point were not stable, post-distribution change.

The SPRT algorithm with a confidence level of 2 tends to make premature predictions about the change, hence the average negative error. SPRT with confidence levels of 4 and 8 both have a negligible prediction error. Again, with there being no difference between λ_1 and λ_2, an error rate of 50 is ideal. It means the algorithms are identifying the change point as being either at the start or the end of the transect, either being a valid statement. SPRT with a confidence level of $s = 8$ is the best performing algorithm.

The Relative Change Algorithm has a high and fairly consistent mean error in the estimated change point. This is a function of the high number of false positives and the unstable performance after the distribution change.

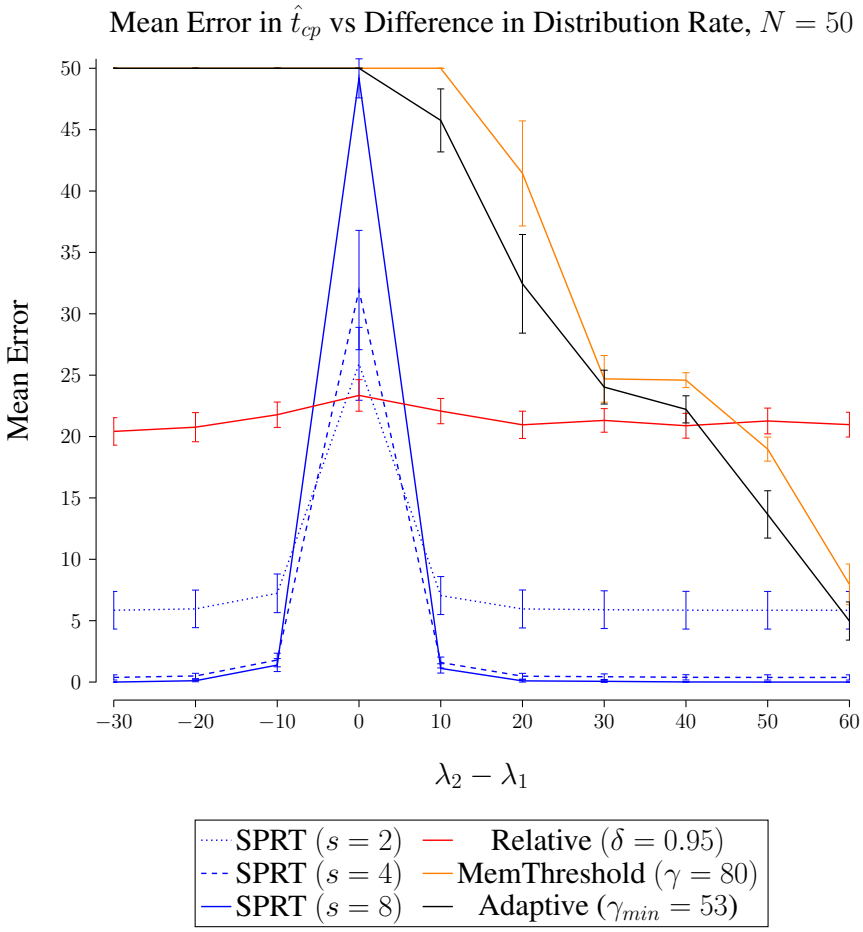


Figure 4.8: The average error between the reported time step and the true change point, for every reported change point. Error bars represent a 95% confidence interval, $N = 50$.

As shown in Table D.7, Table D.8, and Table D.9, the SPRT algorithms maintain statistically significant difference from the Memory Threshold algorithm for all cases except for there being no difference in the underlying distribution when the SPRT confidence level is 8.

The conclusion we draw is that the SPRT algorithm with a confidence level of 8 overall has the best performance over the range of different changes in the magnitude of the background rate. When the change in the underlying distribution is small, it has a slightly higher false negative rate than SPRT with confidence level 4, but this is more than compensated for by the lower false positive rate, and the better performance when there is no change in the distribution. From here on out we exclude other settings of the SPRT confidence level other than $s = 8$.

4.4.2 Experiment 2 Results - Effect of Delay of Change Onset

Based on results from the previous experiment, we only test the SPRT algorithm with confidence level $s = 8$. The SPRT algorithm outperforms the threshold based algorithms. SPRT's false positive rate is largely unaffected by the change point. Threshold-based algorithms have a statistically significantly higher rate, see Figure 4.9. The Relative Change algorithm has a substantially larger false positive rate than the Memory Threshold, Adaptive, or SPRT algorithms.

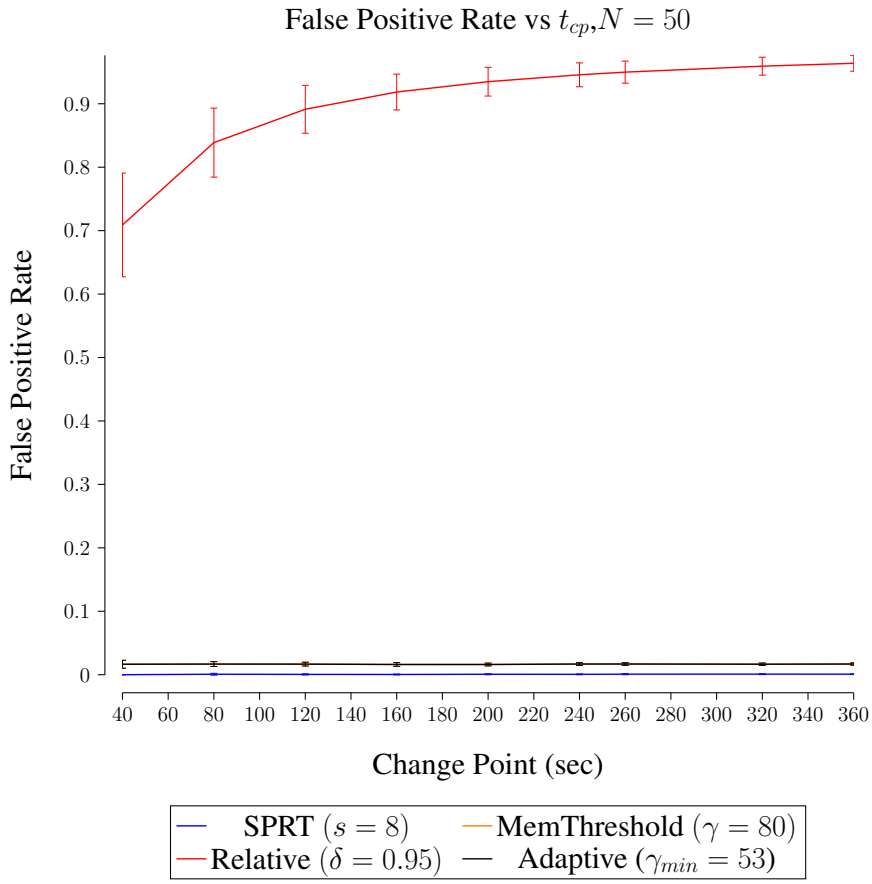


Figure 4.9: As the change point increases the false positive rate of the Relative Change increases towards 97%. Memory Threshold and Adaptive had identical performance. SPRT false positive rate is approximately 0. Error bars show a 95% confidence interval, $n = 50$.

The false negative rate of the SPRT algorithm is lower than that of the Memory Threshold and Adaptive algorithms, with statistical significance and almost entirely large effect sizes, as per Figure 4.10. The false negative rate of the SPRT algorithm does increase as a function of the time change of the distribution, Figure 4.10. Simply, if there is an overwhelming amount of data from the first distribution, it becomes more difficult to identify a change late in the observation scheme.

The Memory Threshold and Adaptive algorithms maintain an approximately constant performance as a function of change in the transition point. We expect an average false negative rate close to zero because the background rate is changed to be above the threshold of the Memory Threshold and Adaptive algorithms, the probability of dropping below the threshold is low. The Relative Change algorithm has a higher false negative rate, and the variability tends to increase with the increase in change onset.

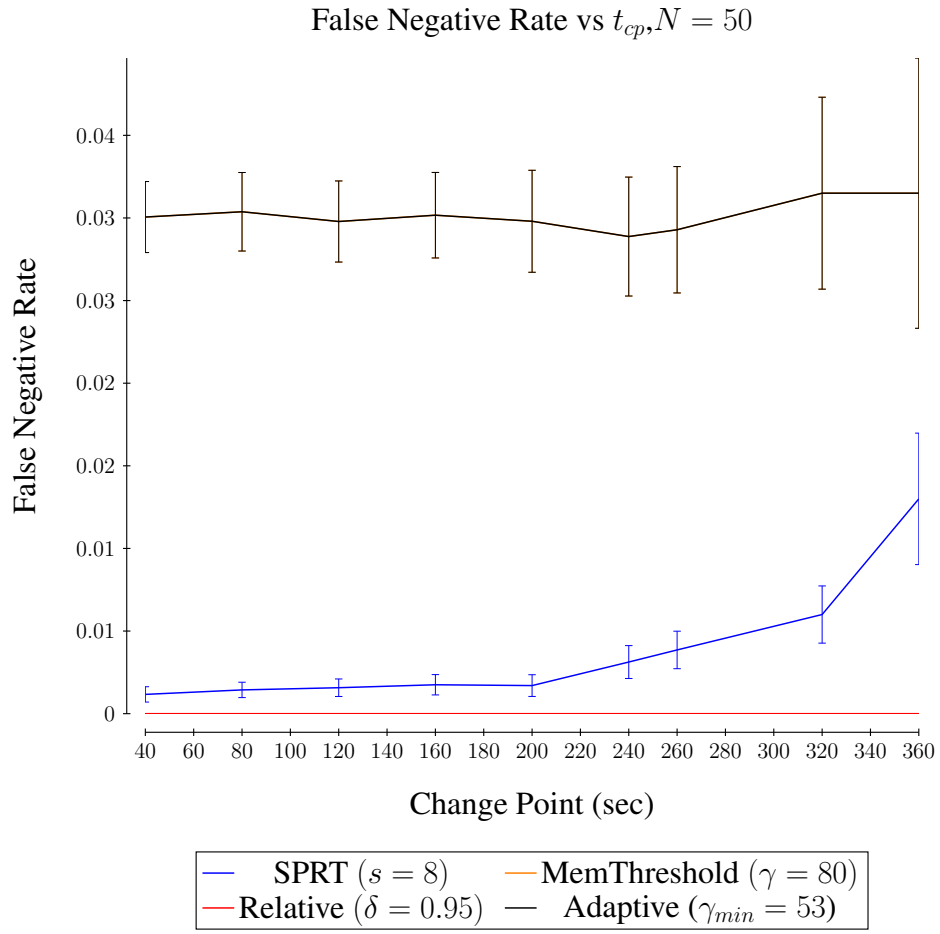


Figure 4.10: False negative rate vs onset time of change. Error bars represent a 95% confidence interval, $N = 50$.

The mean error in the detected time change for the SPRT algorithm is very close to zero, as we see in Figure 4.11. The mean error of the Memory Threshold decreases linearly with the delay in the change, whereas the Relative Change follows a roughly parabolic curve.

The Adaptive algorithm has better performance than the Memory Threshold algorithm. This is consistent with the results found by Ferri et al. (2010). The Memory Threshold algorithm has unstable predictions post-distribution change, as does the Adaptive algorithm. However, because the Adaptive algorithm increases its threshold as more changes are detected, it will reduce the likelihood that a change will be identified. This makes the change estimation more stable, after the distribution change, hence the reduced error rate.

The reason for the linear behaviour of the Memory Threshold algorithm is that it isn't really, at it's heart, designed to detect changes in the underlying distribution, it is designed to detect anomalies. Because of this, whenever the readings rise above or drop below the threshold the algorithm determines that an event worth investigating has happened, and it has no confidence that this is the case.

As a result of this, at different points in the transect the algorithms will receive an observation at t_1 and report that a change has occurred, that $\hat{t}_{cp} = t_1$. Then at a later time, $t_2 > t_1$, the observation will go below the threshold, indicating that it has left the regime of interest but at $t_2 + dt$ the observation may cross the threshold again, causing the algorithm to report that another change in the distribution has occurred, that $\hat{t}_{cp} = t_2 + dt$. As observations keep dipping below the threshold, the more likely that the Memory Threshold algorithm is to report additional changes in the distribution.

The closer the new background rate is to the threshold, the more frequently this is going to happen. The result is that the estimated time change grows with the number of observations collected by the Memory Threshold algorithm. As the time change increases there are fewer opportunities to trigger the Memory Threshold algorithm, hence the reduction in error.

Similar reasoning applies to the Relative Change algorithm. However, where the Memory Threshold algorithm's estimate of the change point is only unstable after the change has occurred, the Relative Change algorithm has unstable estimates before and after the change, due to its high false positive rate. This error reaches a minimum when the change point is in the middle of the transect. The SPRT algorithm, however, maintains a stable estimate of the time change once it has detected the change.

We see this in Figure 4.11. The average error of the change estimate, whenever a change is reported, is considerably higher for the Memory Threshold and Relative Change algorithms than the SPRT algorithm. The Adaptive algorithm has better performance than either the Memory Threshold and Relative Change algorithms, but still inferior to the SPRT algorithm. The error in the SPRT algorithm remains effectively constant, and small. Table D.12 shows that at all times the error in prediction for the SPRT was statistically significantly different from the Memory Threshold algorithm.

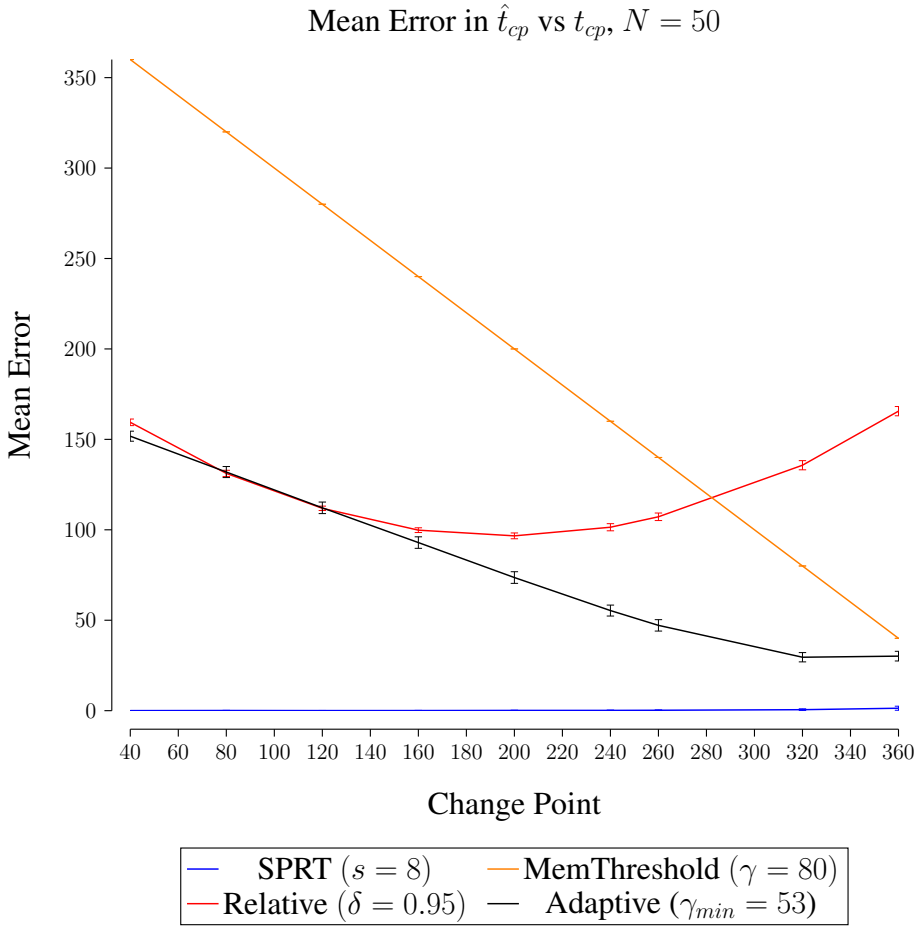


Figure 4.11: The average error in predicting the time point of the distribution change. The SPRT algorithm has a very small error in predicting the change point, independent of the onset of the change. The threshold algorithm, however, has a linear relationship in the change point prediction error. Relative Change algorithm has a roughly quadratic relationship, and never gets below 100 time steps error in estimating the change point. Error bars represent a 95% confidence interval, $N = 50$.

4.4.3 Experiment 3 Results - Real MVP Data

We tested the competing algorithms on data collected during the 2014 MVP operations in the Mojave desert. We are using the NSS data that was collected from the 17^{th} to the 25^{th} of October, 2014.

In this experiment we consider only counts from the stainless steel channel. Figure 4.12 to Figure 4.20 shows the data that were collected during the MVP project. We set the threshold and memory threshold algorithms with the same threshold that was used to identify hot spots in the field. The SPRT algorithm was run with a confidence level of 8, the Memory Threshold was run with a threshold of 80 counts, and the Relative Change algorithm used a threshold of 0.95.

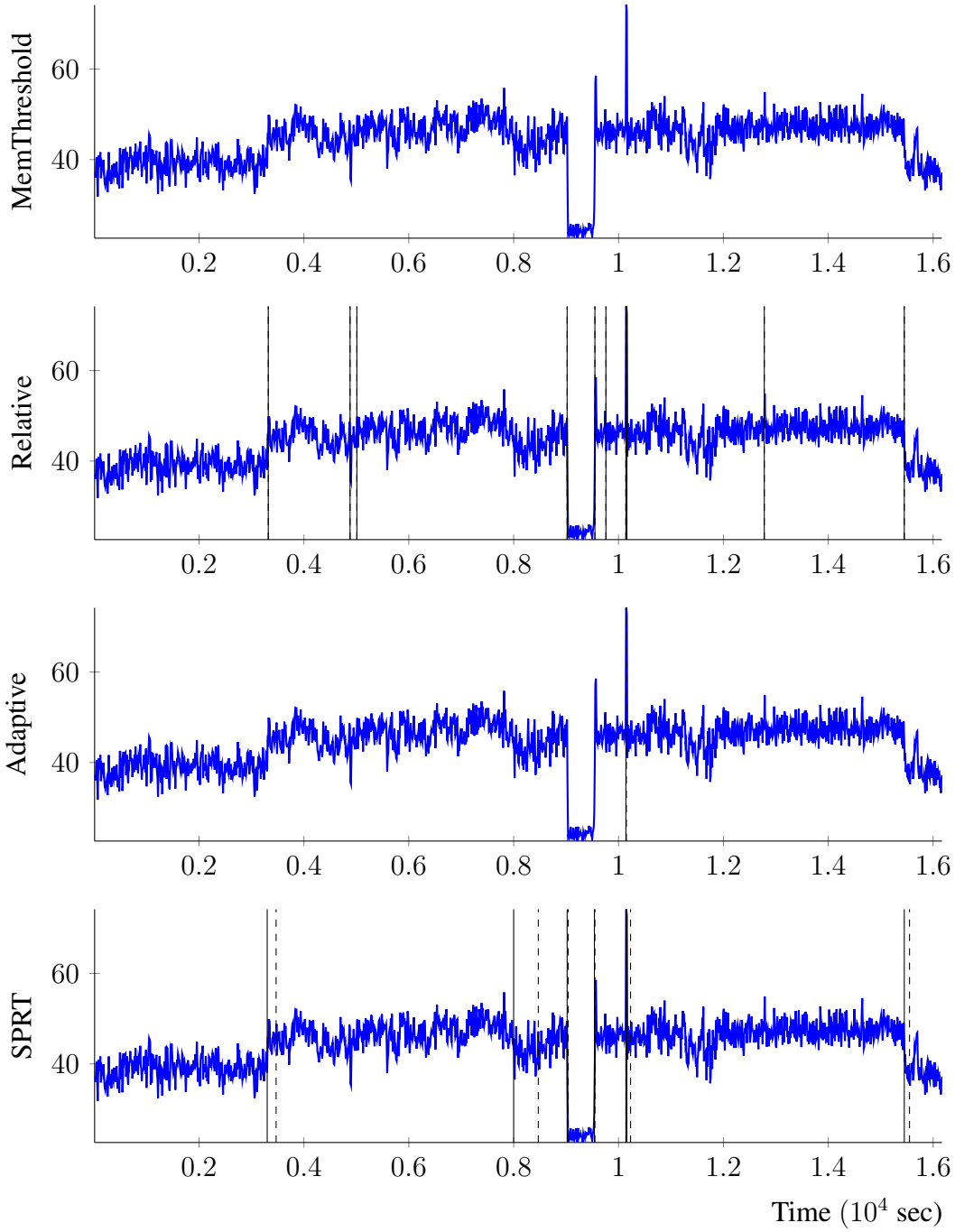


Figure 4.12: The blue line indicates the NSS counts recorded on 17 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.

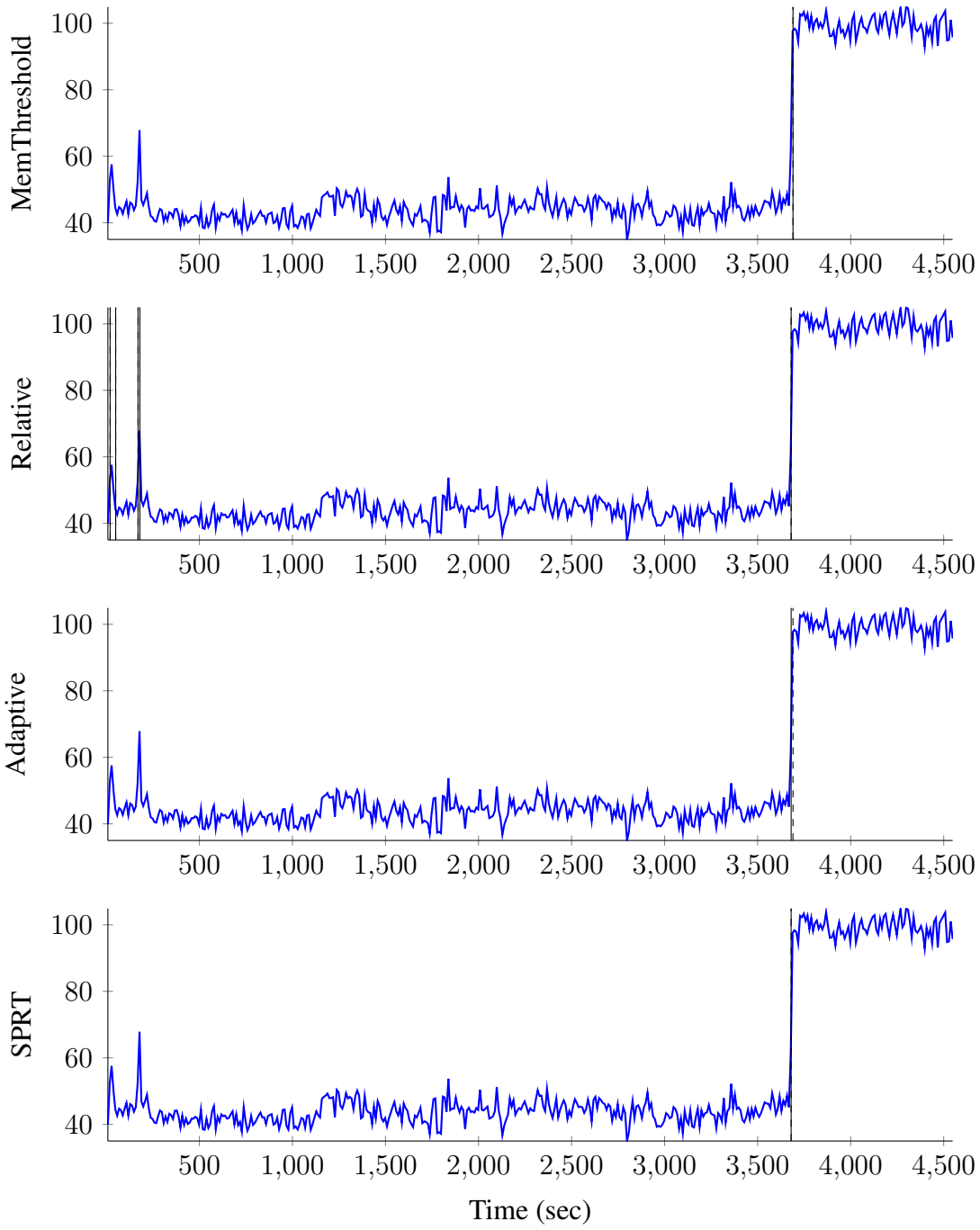


Figure 4.13: The blue line indicates the NSS counts recorded on 18 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.

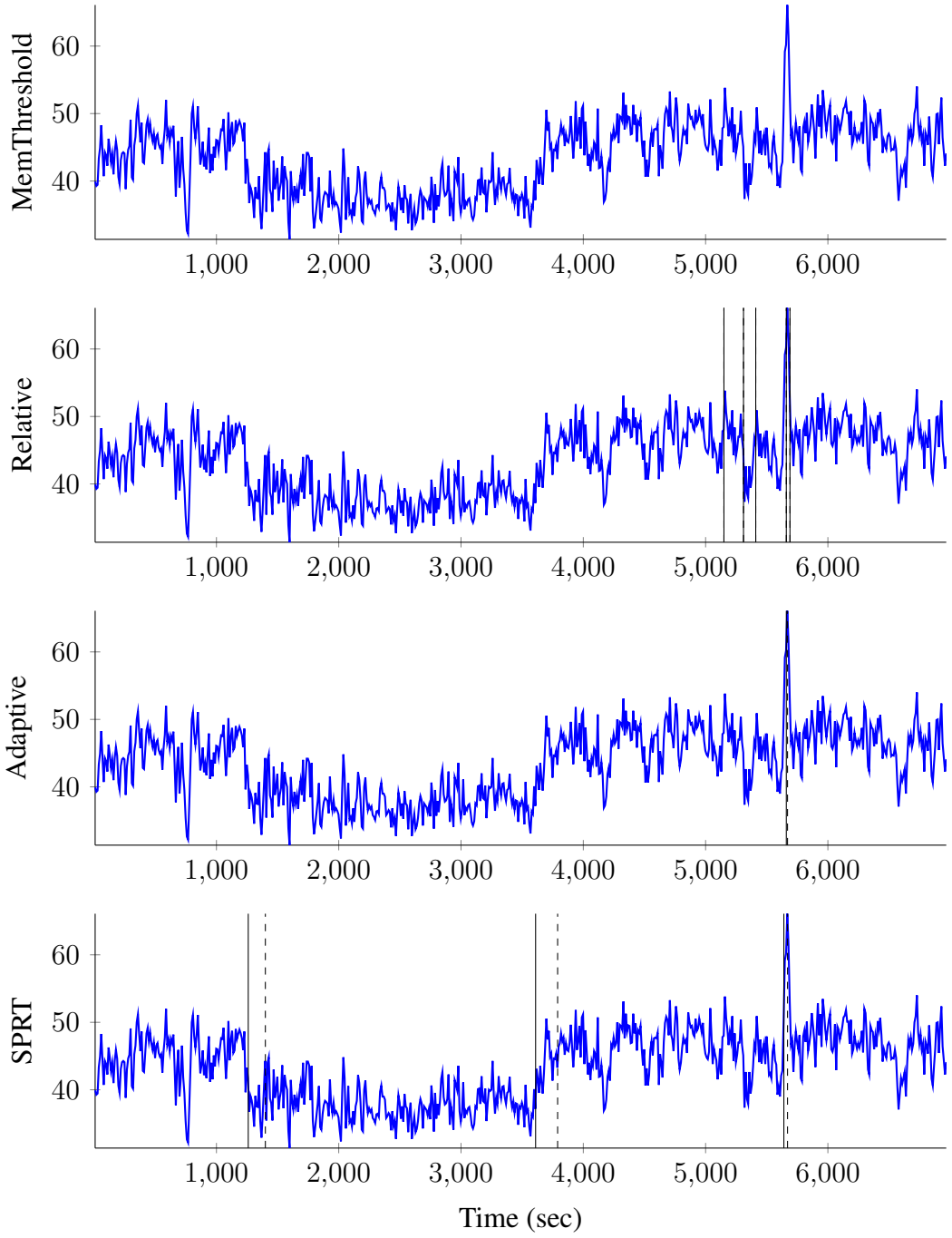


Figure 4.14: The blue line indicates the NSS counts recorded on 19 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.

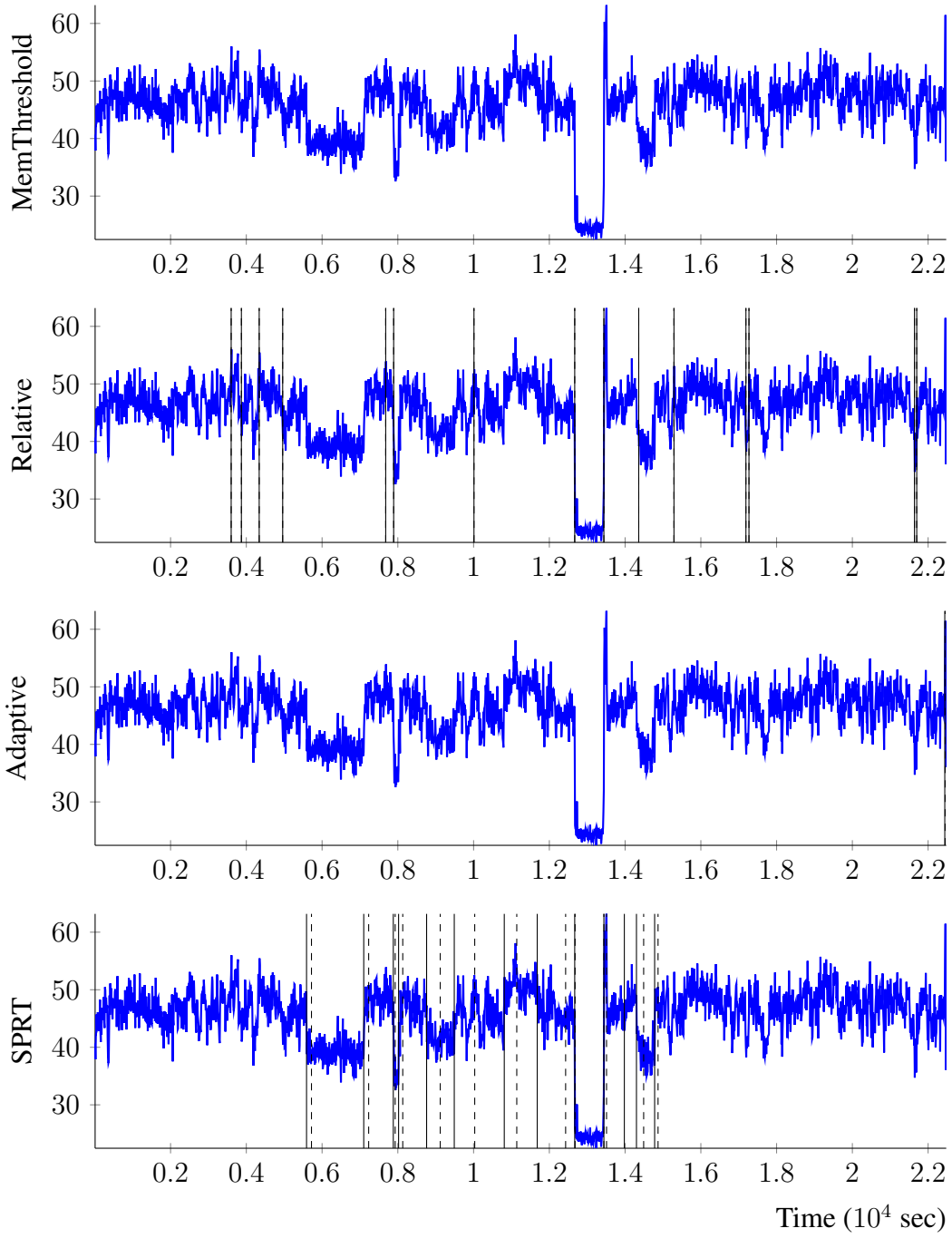


Figure 4.15: The blue line indicates the NSS counts recorded on 20 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.

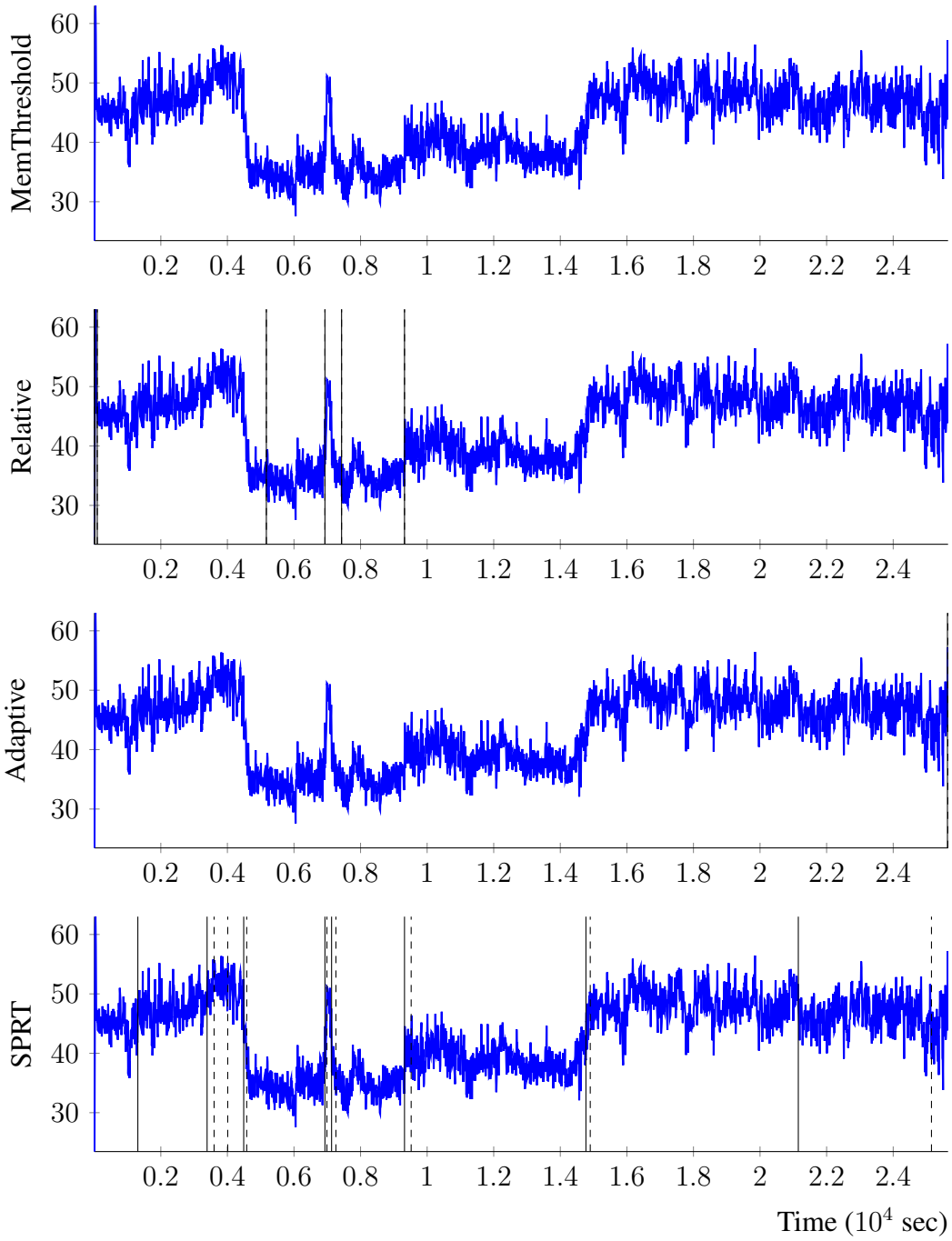


Figure 4.16: The blue line indicates the NSS counts recorded on 21 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.

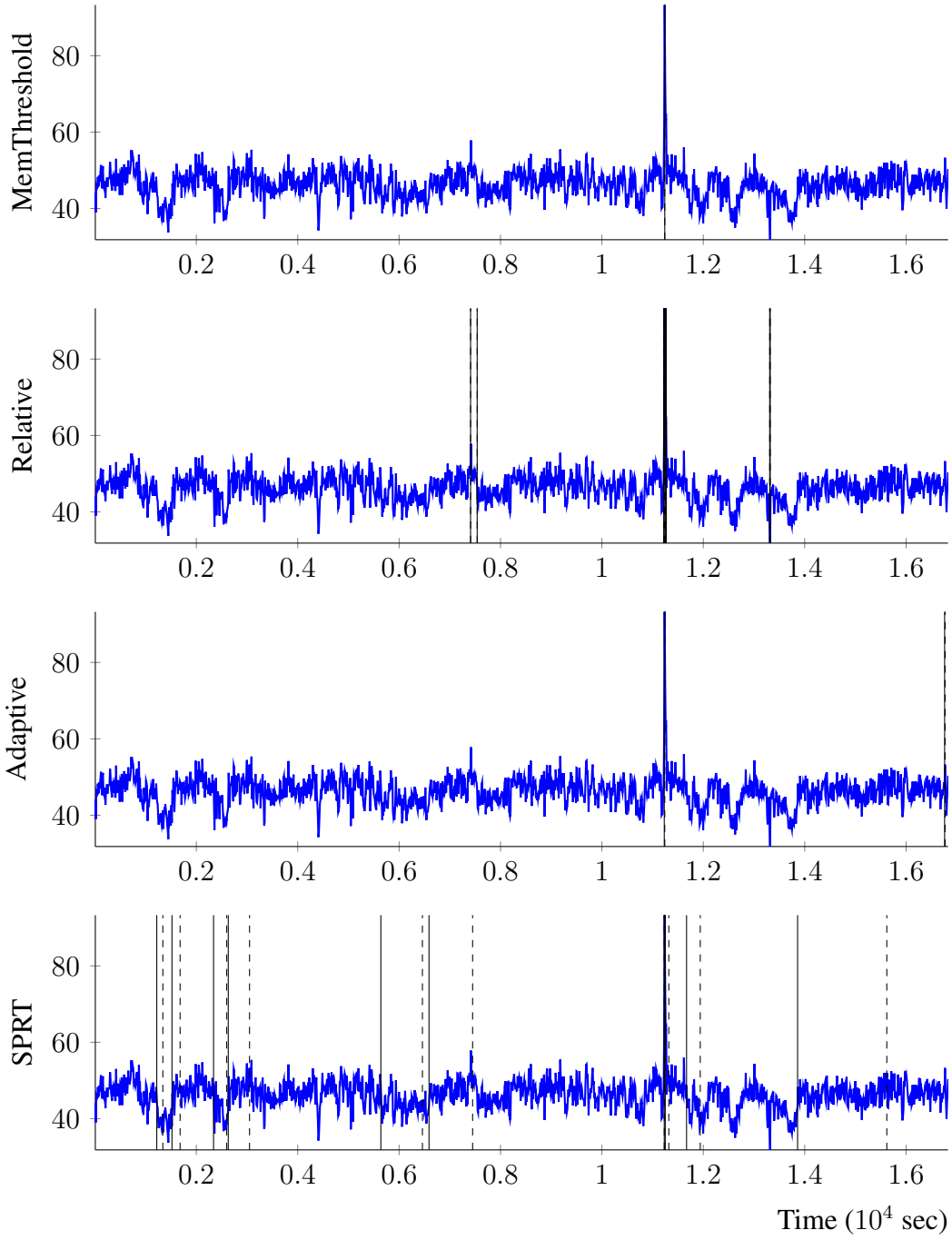


Figure 4.17: The blue line indicates the NSS counts recorded on 22 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.

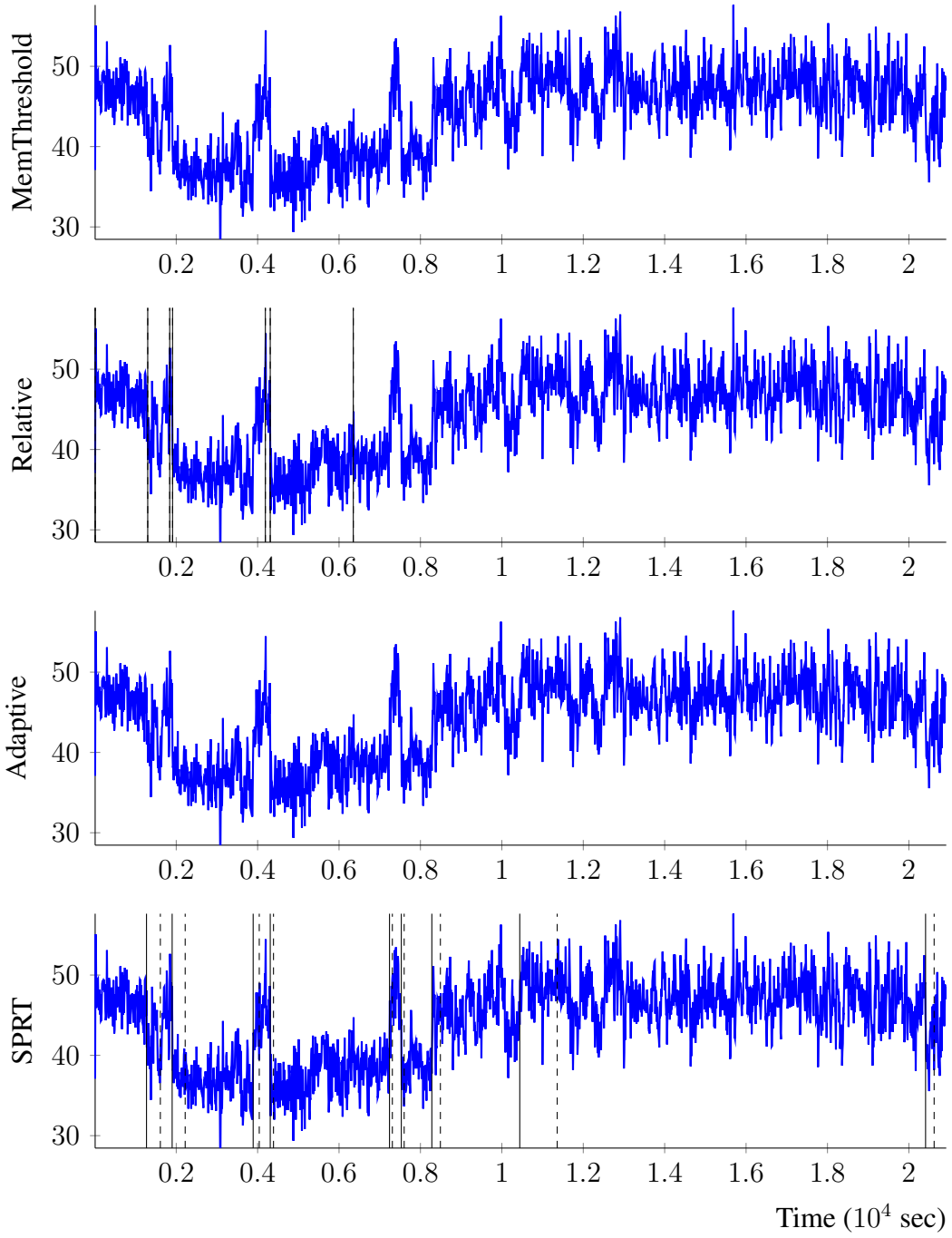


Figure 4.18: The blue line indicates the NSS counts recorded on 23 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.

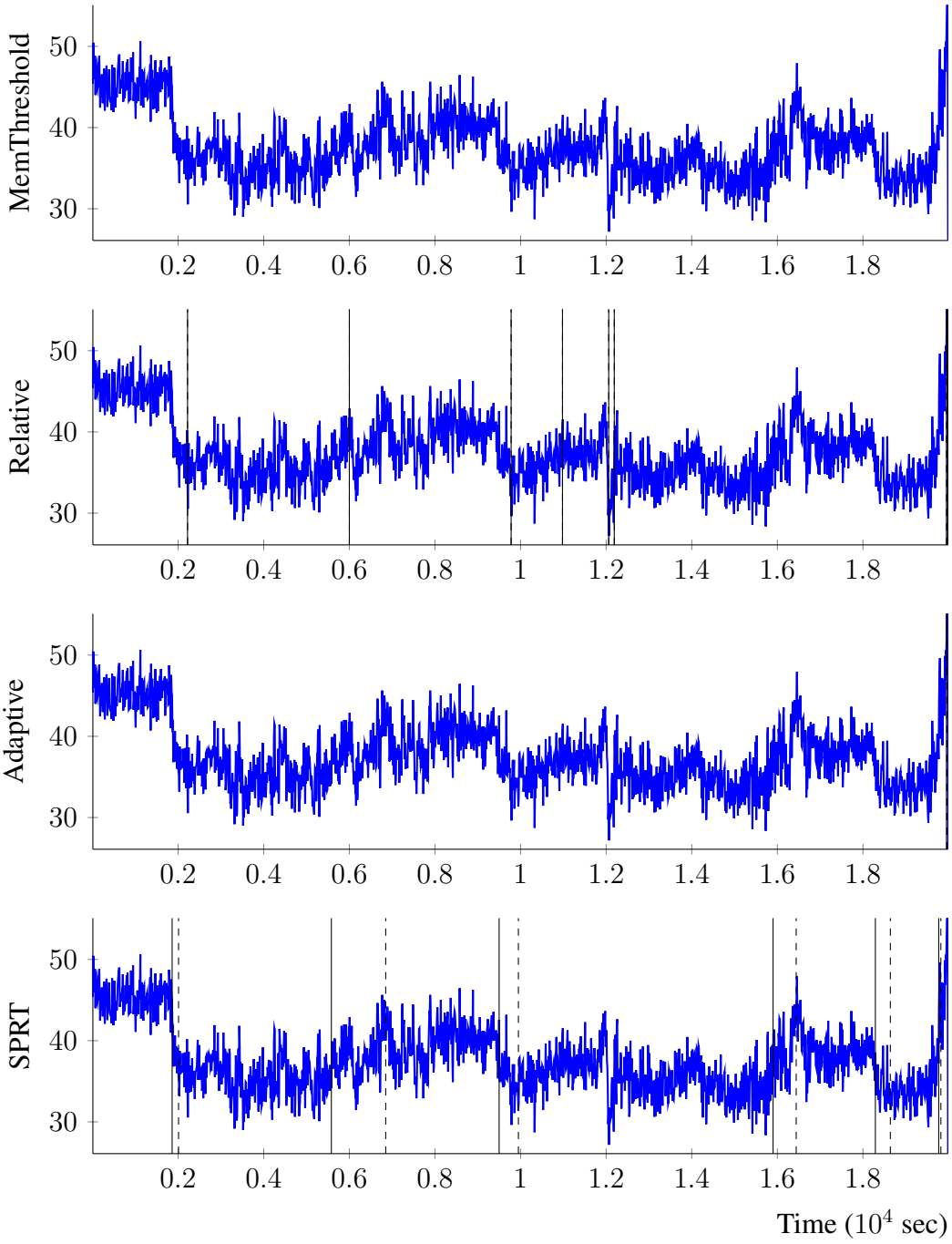


Figure 4.19: The blue line indicates the NSS counts recorded on 24 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.

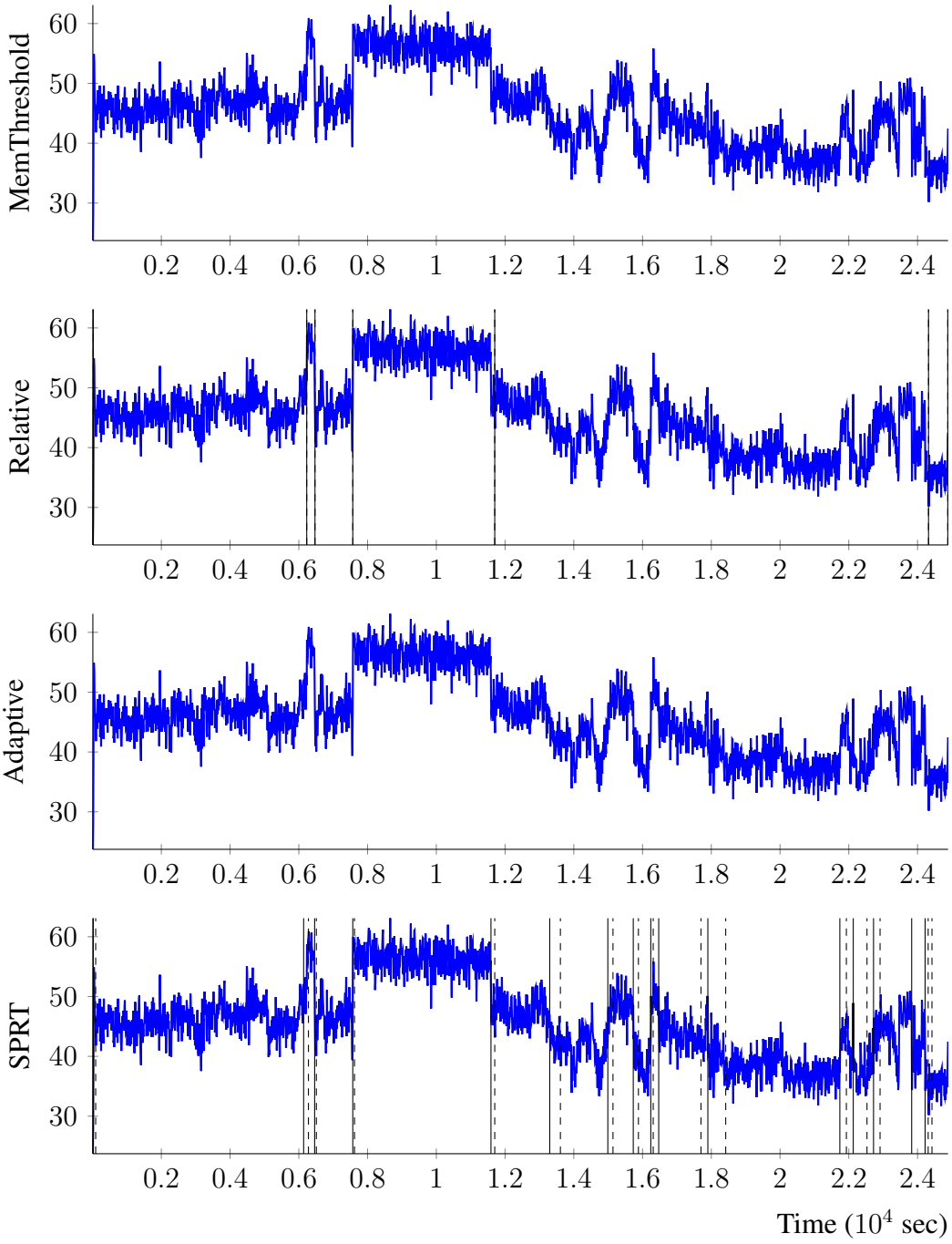


Figure 4.20: The blue line indicates the NSS counts recorded on 25 October 2014. The vertical dashed lines show when the different algorithms detected a change. Solid vertical lines indicate where the algorithm determined the change occurred.

Table 4.10: In this table we report the results of the Memory Threshold, Adaptive , Relative Change, and SPRT algorithms on real-world data. While SPRT does report some false positives, they are rare. We also see that SPRT notices changes that are not observed by the Memory Threshold algorithm simply because they are happening below the threshold. This is done without adapting the algorithm to the environment.

Date	**Memory Threshold**			**Relative**			**Adaptive**			**SPRT**		
	TP	**FP**	**FN**	**TP**	**FP**	**FN**	**TP**	**FP**	**FN**	**TP**	**FP**	**FN**
2014-10-17	0	0	7	5	3	2	1	0	6	6	0	1
2014-10-18	1	0	2	3	0	0	1	0	2	1	0	2
2014-10-19	0	0	10	4	0	6	1	0	9	3	0	7
2014-10-20	0	0	13	8	6	5	1	0	12	12	1	1
2014-10-21	0	0	13	4	1	9	1	0	13	8	0	5
2014-10-22	1	0	8	3	1	6	1	1	8	8	1	1
2014-10-23	0	0	11	3	1	8	0	0	11	9	0	2
2014-10-24	0	0	9	4	1	5	1	0	8	5	1	4
2014-10-25	0	0	17	5	1	12	0	0	17	16	1	1
Total:	2	0	90	39	14	57	7	1	86	68	4	24

We can use these aggregate data to fit Beta distributions over the precision, computed $\frac{TP}{TP+FP}$, and recall, computed $\frac{TP}{TP+FN}$, of the different algorithms. We can also use these distributions to determine the probability that any one algorithm's performance is higher than the others.

Table 4.11: We estimate the precision and recall of the algorithms estimated from the data in the final row of Table 4.10. The Memory Threshold algorithm has a high precision, but the overall level of detection is quite poor, as seen in its recall score.

Algorithm	**Precision**	**Recall**
SPRT $(s = 8)$	0.94	0.74
Memory Threshold $(\gamma = 80)$	1.00	0.02
Adaptive $(\gamma_{min} = 50)$	0.88	0.08
Relative $(\delta = 0.95)$	0.74	0.41

The precision and recall of the algorithms are given in Table 4.11 and the statistical significance of the relative performance is given in Table 4.12. As we can see the SPRT algorithm has a statistically significant improvement over the baseline algorithms, except for the precision of the Memory Threshold algorithm. While the Memory Threshold algorithm does have superior precision, we can see that its rate of detecting changes is sufficiently low that the improvement in precision doesn't warrant adopting the algorithm.

Table 4.12: The comparison of the SPRT algorithm to the Relative and Memory Threshold algorithms. We see that the SPRT algorithm has superior recall compared to either algorithm, with probability $\geq 95\%$, and huge effect sizes. The SPRT algorithm also has statistically significantly larger precision than the Relative change algorithm with probability $\geq 95\%$ and huge effect size.

Comparison	Metric	$p \geq 95\%$	Effect Size
SPRT $>$ MemThreshold	Precision	N	N/A
SPRT $>$ MemThreshold	Recall	Y	**14.83**
SPRT $>$ Relative	Precision	Y	**3.15**
SPRT $>$ Relative	Recall	Y	**4.92**
SPRT $>$ Adaptive	Precision	Y	**22.58**
SPRT $>$ Adaptive	Recall	Y	**12.51**

Sensitivity of Parameters

The above experiment demonstrates the performance of the algorithms with parameters that were tuned to the environment. When operating in an unknown environment one will not have the opportunity to tune the parameters in the fashion that we conducted above. To demonstrate the sensitivity of the algorithms' performance with respect to parameter settings we constructed a Receiver Operating Curve demonstrating the change detection abilities of the algorithms as a function of parameter change.

The ROC curve for the algorithm performance is given in Fig. 4.21. We can see that the SPRT algorithm generally has superior true positive and false positive rates compared to the competing algorithms. Data supporting the ROC curve is given in Tables D.13 to D.15 and Tables D.16 to D.27. Note that the false positive rates are very small, less than 10^{-2}. The number of samples recorded by the sensor is on the order of $\sim 10^3 - 10^4$, and most of those data points do not represent a change in the underlying distribution. So while the false positive rate may be quite small for an algorithm, it could easily be reporting as many false positives as true positives.

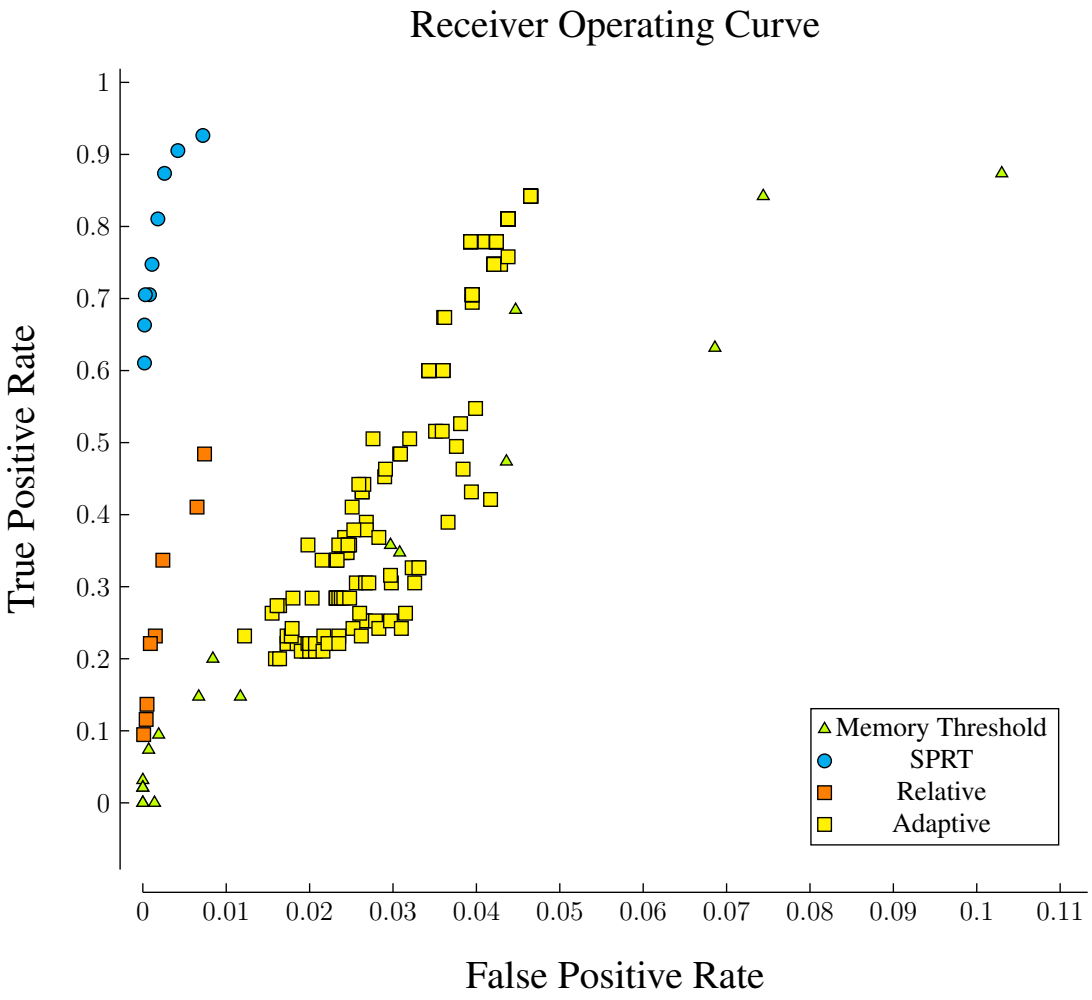


Figure 4.21: In the above figure we can see that the SPRT algorithm is capable of producing better true positive rates than the other algorithms while maintaining lower false positive rates.

In Fig. 4.21 we see that the SPRT algorithm produces generally superior performance to the other algorithms. We can control the false positive rate by increasing the confidence level, s. While the relative change algorithm can produce comparable performance to the SPRT algorithm with respect to the false positive rate, we see that it is inferior with respect to the true positive rate.

Both the Adaptive Threshold and the Memory Threshold algorithms suffer higher rates of false positive in exchange for better true positive rates. Their ROC curves also notably loop back on themselves. As thresholds are lowered, more of the true changes will be identified, increasing the true positive rate, and likewise increasing the false positive rate. However, once the threshold reaches a certain level, for the Memory Threshold algorithm, only one change will be detected, reducing the true positive rate as well as the false positive rate.

The Adaptive Threshold algorithm is sensitive to the γ_{MIN} parameter. We see that by changing γ_{MIN} that the false positive rate can be reduced, but at a cost to the true positive rate. Changing the γ_{MIN} parameter can have as big an effect on performance as lowering the threshold, γ,

for the Memory Threshold algorithm, but it preserves the true positive rate. However, this performance is still inferior to the SPRT algorithm with respect to the false positive rate. A more thorough examination of the effect of the γ_B parameter is found in Fig. D.1.

4.4.4 Experiment 4 - Effect on Performance in 2D Operations

In (Ferri et al., 2010) the objective was to localize maxima in the map. To determine the effectiveness of the SPRT algorithm we have simulated an experiment where a rover is following a lawnmower pattern across a terrain that has pockets of water density that will drive different average rates of the Poisson process for the NSS sensor.

The algorithm does not know that it is operating in a two dimensional environment. This choice was made in order to gauge the performance of the algorithm as a trajectory-agnostic component to a robot scientist. Futher, this mirrors the deployment of the threshold algorithm employed by Ferri et al. (2010). There are higher-level concerns that are ignored by the algorithm which can be delegated to a higher-level module of the robot, much like the CASPER system in OASIS/AEGIS.

We took data from the NSS readings from the MVP field experiment and fit a distribution over the number of counts. We sampled 100 observations from this distribution and used them as seeds to generate a Voroni map, called M, representing a 50m by 50m area, with a grid size of 0.5m.

The map was then smoothed with a Gaussian blur with a scale factor of $\sigma = 0.8$ in order to mimic the effect of the field of view of the sensor. At each point (x, y) in M the robot is given a sample from a Poisson distribution with $\lambda = M(x, y)$. We generated 30 maps and ran ten trials on each map for each algorithm.

The robots follow a trajectory that follows a lawnmower pattern across the width of the map and with the swaths of the path having a spacing of 5m. The trajectory is illustrated in Figure 4.4. We compare three algorithms. The first simply follows the lawnmower path without conducting any AIMs. The second algorithm conducts an AIM when an observation crosses a threshold, as per (Ferri et al., 2010). The third algorithm uses the SPRT algorithm described above, and when a transition is detected going from a lower background rate to a higher background rate it conducts an AIM, in the hopes of being on a gradient towards a local maxima.

Because the objective of the experiment in (Ferri et al., 2010) was to localize the source of chemicals, we set the robot in the simulation the task of collecting high-value observations. High value observations were defined to be local maxima in the water map. To score the performance of the robot we counted the number of instantaneous observations that were above the threshold over the course of the transect. To score the performance on localizing maxima we count the number of maxima that fell within the robot's field of view while driving. Higher scores are better. A threshold of 80 counts was established in conversation with member of the MVP team.

To identify the number of maxima that are captured by virtue of using the different algorithms we first subtracted the number of peaks that were captured by not doing any AIMs. Figure 4.22 plots the number of unique maxima captured vs the number of AIMs that were deployed by the algorithms. The raw data is available in Table D.31 and Table D.30.

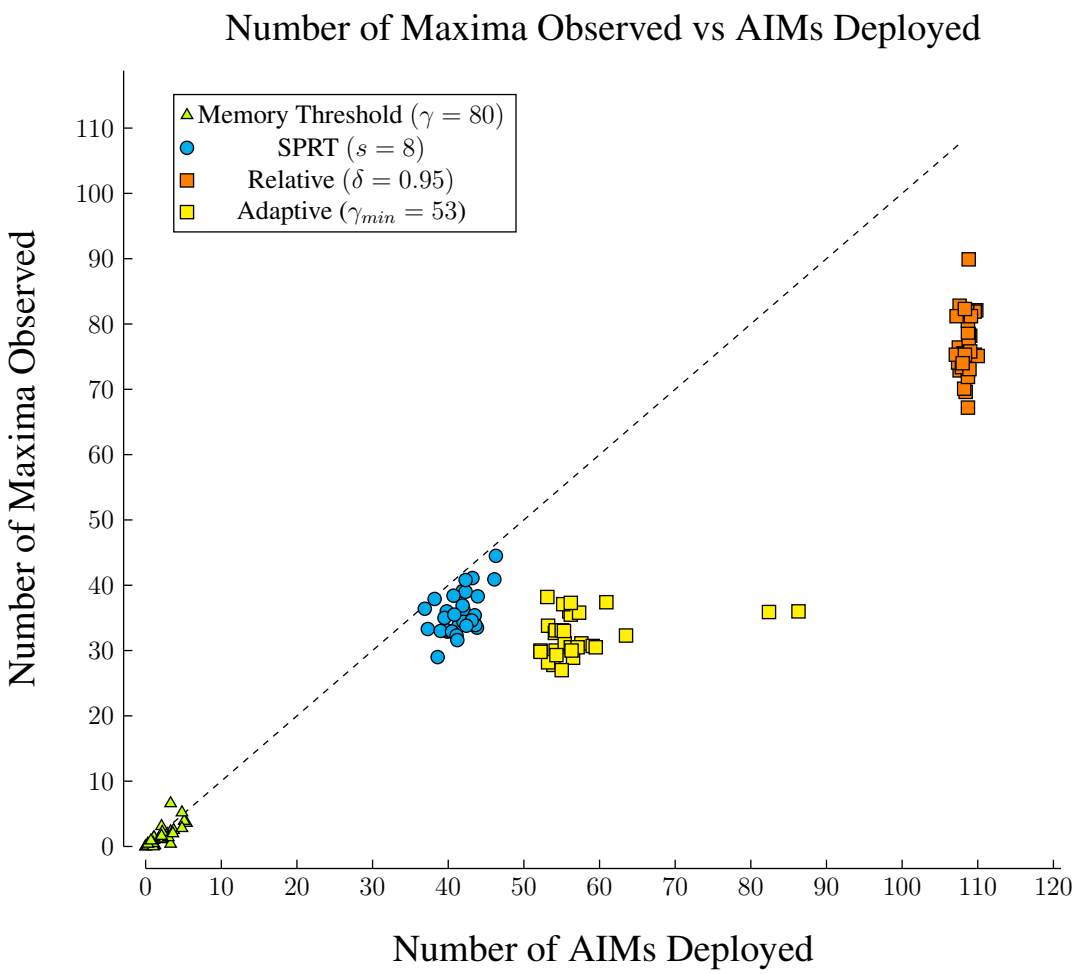


Figure 4.22: The number of maxima observed by the algorithms versus the number of AIMs deployed for the 30 different simulated maps. The Relative Change algorithm captured a greater number of maxima than the SPRT algorithm, and the SPRT captured a greater number of maxima than the Memory Threshold algorithm. The Adaptive algorithm deployed slightly more AIMs than the SPRT algorithm, and observed a similar number of maxima. The dashed line has a slope of 1. The further below the line, the less efficient the algorithm is in deploying AIMs. Each point represents $n = 10$ trials on each map.

The Relative Change, SPRT, and Memory Threshold algorithms were all statistically significantly different from one another in terms of both the number of AIMs deployed and the number of maxima captured. The Adaptive algorithm deployed a significantly different number of AIMs from the other algorithms, but the number of maxima captured was statistically indistinguishable from the SPRT algorithm at a confidence level of 95%.

The number of AIMs deployed and maxima observed by the algorithms are given in Table 4.13. The statistical significance of the differences in performance is given in Table 4.14. We ordered the comparisons by decreasing number of AIMs deployed. With the exception of the number of maxima observed by the SPRT and Adaptive algorithms, all other quantities were distinct at a confidence level of 95%.

Table 4.13: The number of AIMs deployed and maxima observed by the different algorithms averaged over the 30 simulated maps. Relative Change deploys the most AIMs and observes the most maxima, followed by the Adaptive and SPRT algorithms, then the Memory Threshold algorithm.

Algorithm	**AIMs Deployed** ($\mu \pm \sigma$)	**Maxima Observed** ($mu \pm \sigma$)
Relative Change ($\delta = 0.95$)	108.5 ± 0.7	76.8 ± 4.7
Adaptive ($\gamma_{min} = 53$)	57.7 ± 7.5	32.4 ± 3.14
SPRT ($s = 8$)	41.4 ± 2.2	35.8 ± 3.3
Memory Threshold ($\gamma = 80$)	2.2 ± 1.5	1.7 ± 1.6

Table 4.14: Relative Change deploys more AIMs than SPRT, and it observes more maxima than SPRT with an extremely large effect size at a confidence level of 95%. Similarly the SPRT algorithm deploys more AIMs than the Memory Threshold algorithm, and it also observes more maxima in the map, at a confidence level of 95% and a very large effect size. The number of maxima observed by the Adaptive threshold is statistically indistinguishable from the SPRT algorithm, while deploying more AIMs.

Comparison	**AIMs Deployed**		**Maxima Observed**	
	$p \geq 95\%$	Cohen's d	$p \geq 95\%$	Cohen's d
Relative Change > Adaptive	Y	**6.83**	Y	**10.4**
Adaptive > SPRT	Y	**2.33**	N	N/A
SPRT > Memory Threshold	Y	**14.79**	Y	**8.86**

However, if we look at how effective the AIM deployments are, the picture changes somewhat. We can measure the effectiveness of the algorithms by the ratio of maxima captured to AIMs deployed. If every AIM deployed resulted in discovering a unique maxima in the field, then the ratio would be 1. Values greater than 1 would be preferred, and the further below the dashed line in Figure 4.22, the less effective the algorithm is.

The Memory Threshold algorithm and the SPRT algorithm are statistically indistinguishable in their relative effectiveness at a confidence level of 95%. The uncertainty for the Memory Threshold algorithm is much higher than the other algorithms because there were two trials where the Memory Threshold algorithm did not deploy any AIMs, and so they were excluded from the analysis. The effectiveness of the different algorithms is reported in Table 4.15.

The Adaptive algorithm had the worst effectiveness. We conjecture that is because, as the transect progresses, the threshold for deploying an AIM is reduced, should AIMs have not been deployed. If the threshold is lowered then AIMs are likely to be deployed in less favourable environments.

Table 4.15: The ratio of maxima observed to AIMs deployed. An effectiveness of 1 would mean that every AIM deployed observes a unique maxima in the underlying map.

Algorithm	**Effectiveness** ($\mu \pm \sigma$)
SPRT ($s = 8$)	0.86 ± 0.07
Memory Threshold ($\gamma = 80$)	0.77 ± 0.45
Relative Change ($\delta = 0.95$)	0.70 ± 0.04
Adaptive ($\gamma_{min} = 53$)	0.57 ± 0.07

Using a paired significance test the SPRT algorithm has a higher effectiveness than the Relative Change algorithm at a confidence level of 95% and an effect size of $d = 2.56$. The SPRT algorithm has a greater effectiveness than the Adaptive algorithm at a confidence level of 95% and effect size of $d = 3.80$. The average difference in effectiveness between the algorithms is given in Table 4.16.

Table 4.16: The difference in effectiveness for the different algorithms. At a 95% confidence level it is not possible to distinguish the performance of the Memory Threshold from the SPRT algorithm. There are two transects where the Memory Threshold algorithm did not deploy any AIMs, omitting these transects makes the performance of the Memory Threshold algorithm sufficiently variable as to make its performance indistinguishable from the other algorithms. The SPRT algorithm has superior effectiveness than either the Relative Change or Adaptive algorithms.

Comparison	Δ **Effectiveness** ($\mu \pm \sigma$)	$p \geq 95\%$	Cohen's d
SPRT > Relative Change	0.16 ± 0.06	Y	**2.56**
SPRT > Adaptive	0.30 ± 0.08	Y	**3.80**
SPRT > Memory Threshold	0.09 ± 0.46	N	N/A

What we can conclude from these results is that the SPRT algorithm is as efficient as the Memory Threshold algorithm, at a confidence level of 95%, but it collects a greater number of maxima in the underlying map. The Relative Change algorithm captures more maxima, but comes at the cost of significantly reduced efficiency, this is the result of the false positives observed in subsection 4.4.1 and subsection 4.4.2.

We see that the Adaptive threshold was able to capture more maxima than the Memory Threshold algorithm, which is consistent with the results reported by Ferri et al. (2010). However, with our formulation of the adaptive threshold, we see that the adaptive algorithm is less productive, largely because of how it lowers its standards for deploying an AIM as the threshold is lowered.

Fig. 4.23 shows the sensitivity of the algorithms to changes in the controlling parameters. For the Memory Threshold, Relative Change, and SPRT algorithms, all seem to follow trend lines with similar slopes. The Adaptive Threshold algorithm follows a different trendline with a lower

slope than the other algorithms. As with the ROC curve for Experiment 3 the results for the Adaptive Threshold algorithm was selected to show the best performance, and the performance for different settings of the intial threshold are given in Fig. D.2 and Fig. D.3.

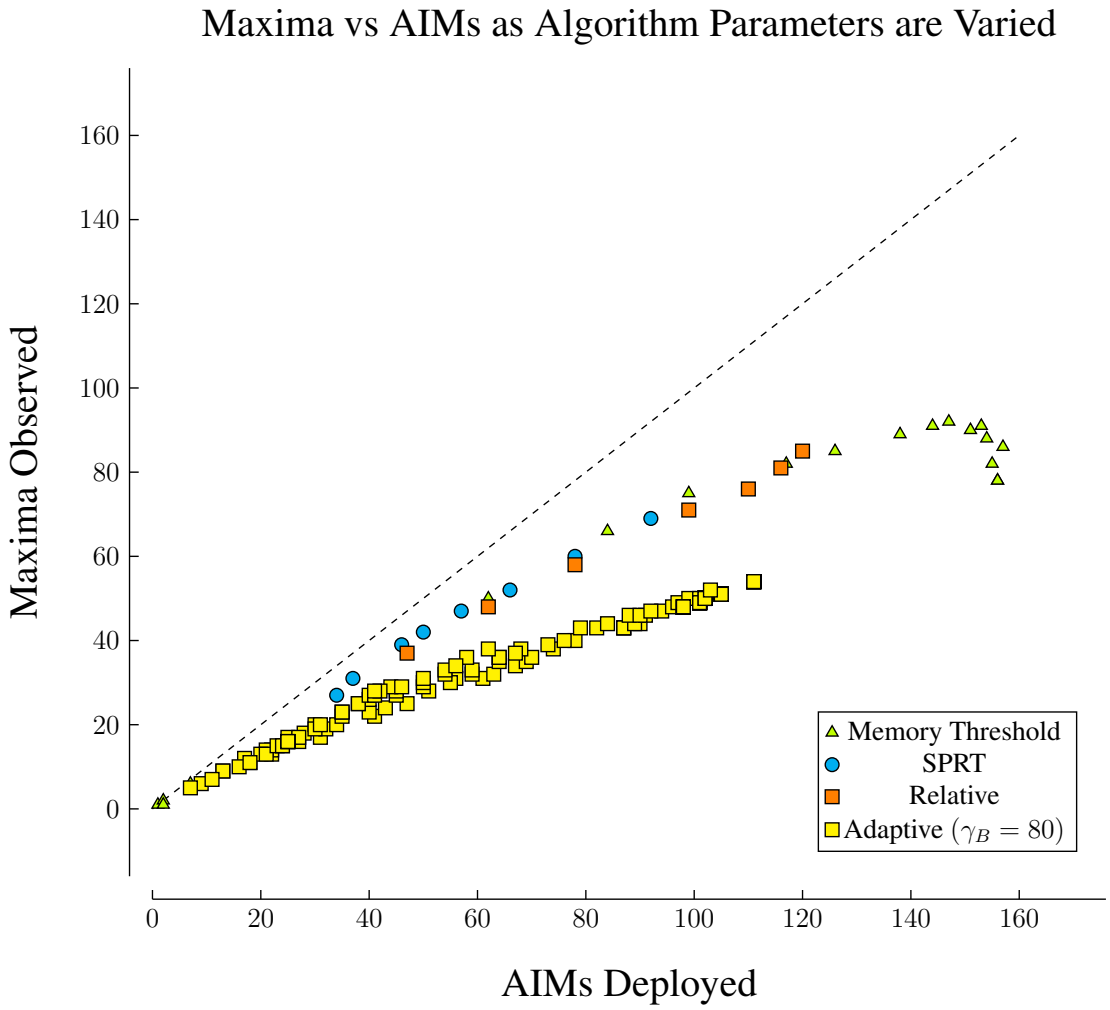


Figure 4.23: The individual points in the plot represent the performance of the algorithms averaged over all 30 water maps. As in Fig. 4.22, the y-axis is the number of maxima observed by the algorithm less the maxima that would be observed by following the lawnmower path without deploying any AIMs. The dashed line has a slope of 1.

For the Memory Threshold algorithm, when the threshold γ is high, the number of AIMs deployed is low, but they generally result in observations of maxima. As γ is lowered the Memory Threshold algorithm deploys more AIMs but, at the extreme, the number of maxima observed saturates.

The Relative Change algorithm with the higher settings of δ performs comparably to the SPRT algorithm. At the lowest settings of δ the Relative Change algorithm deploys more AIMs and observes more maxima in the water map. However, when the number of AIMs deployed by the Relative Change algorithm are similar to that of the SPRT algorithm, they observe slightly fewer maxima in the water map.

The SPRT algorithm does not deploy as mainy AIMs as the Relative Change or Memory Threshold algorithm. At its lowest setting of s, the SPRT remains safely out of the region of saturation that the Memory Threshold algorithm enters, but at the cost of observing fewer maxima in the water map. With the highest setting of the s parameter the number of AIMs deployed is lower, but at the benefit of higher effectiveness.

The Adaptive Threshold algorithm does not make as effective use of its AIMs as the other algorithms. This can be ascertained from the lower slope of the data points. As can be seen in the Fig. D.2 and Fig. D.3 the performance of the algorithms does not significantly vary. The most effective use of AIMs is when the γ_{min} parameter is at the highest value, but this results in fewer AIMs being deployed and hence fewer maxima being observed.

The advantage of the SPRT algorithm is that it gives a measure of confidence that can be used in mission-level decision making to deviate paths, which is not something that can be done with the Relative Change algorithm alone. However, the SPRT algorithm can be viewed as an integration of the observations from the Relative Change algorithm to the point where such a decision can be made.

4.5 Discussion

The algorithms presented in this chapter assumed that only one change in the underlying distribution occurred. We were able to mitigate this in the 2D exploration simulation by caching any history of previously collected samples. However, it is possible that multiple changes could have occurred in the time series collected by the robot. Representing this, and dealing with an unknown number of changes would require more sophisticated machinery, but could be addressed by techniques like the multi-hypothesis developed by (Baum and Veeravalli, 1994), making for a logical extension of the work.

Similarly, we assume that the functions driving the background rate of the observations is constant in between changes. If that assumption does not hold up, then the hypotheses that are used to model the changes may not be able to do so correctly. For example, if the background rate was determined by a periodic function then a change detection algorithm assuming constant background rates would most likely frequently report that changes have occurred.

We assumed that the boundaries of changes in the underlying distribution are sharp, an assumption motivated by mission context in which this work was developed. There is a distinct boundary between different settings of the underlying distribution. A change that occurs more slowly over a long period of time may not be identified by the approach described in this chapter and would require a different competing hypotheses in the decision making algorithm. This is where an approach like the one described in (Thompson et al., 2013) could be used as input to the decision making algorithm.

In the experiments we conducted in this chapter we assumed that the values reported from the sensor had infinite certainty. Accounting for uncertainty in the values would be a more principled (read: Bayesian) approach to change detection.

The changes in the distribution underlying the readings collected by the robot were considered to to be a function of distance along the robot's pre-determined path. This is considering the path to be one dimensional. In actuality the robot is following a trajectory through (at least)

two-dimensional space, and it should be possible to incorporate knowledge about the spatial relationships between readings to make better decisions. This does come at the cost of the simplicity of the algorithm discussed in this chapter and so trade-offs in computing resources should be considered in a mission context.

Because the scientists employing the neutron spectrometer have well characterized the relationship between the readings of the instrument and the abundance of subsurface water, there isn't a need to learn that relationship on-line. However, one might be in the situation where the relationship between the prospecting sensor and the secondary sensor is not characterized and needs to be determined on-line. In this setting an upper confidence bound value function (Auer, 2003) like the one used by Das et al. (2015) would be more appropriate.

The decision rule used to determine if an AIM should be employed was a fairly trivial exemplar to illustrate the efficacy of the approach. Actual missions should employ AIM deployment decision rules that reflect mission risk and vehicle resources. Likewise the threshold for confidence in the change in the underlying distribution implies a certain risk tolerance. The confidence threshold should also be tuned to mission risk.

We do not consider the case of having a limited sampling budget. Should that become a constraint then the authors would recommend combining the change detection algorithm described in this chapter with the submodular secretary described in (Das et al., 2015).

The algorithm assumes that a path is pre-determined for the vehicle. In the 2D simulation we considered a lawnmower path of fixed width. The success and failure of the mission could depend on the spacing of switchbacks along that path. An informative path planner could improve the overall performance of the algorithm, reducing the risk of missing locations of interest in the world.

The SPRT algorithm as implemented searches through every time step in the data that have been collected from the last detected change. This is a linear search in the number of time steps, T, which in turn requires an order T computation, yielding $O(T^2)$ complexity. However, we can reframe this problem as searching for the roots of the equation $\lambda(t) - s$, where s is the confidence level threshold for choosing one hypothesis over the other. With this reframing we could employ, for example, a bisection search which would have a worst-case complexity of $O(T \log T)$.

4.6 Summary

The objective of the work in this chapter was to use a change detection algorithm to improve the deployment of discrete secondary sampling actions. We chose an Area of Interest Manoeuvre as the discrete, secondary sampling action. We designed an algorithm which is a straightforward application of the sequential probability ratio test and compared it against a hard threshold technique which is used, in the author's experience, by planetary scientists.

We have shown that the SPRT-based algorithm is able to outperform the Memory Threshold, and Relative Change algorithms on real data from the MVP project, and in the simulated 2D environment. Most importantly, it is able to detect sub-threshold changes that would be invisible to an algorithm using a hard decision-making threshold.

By recognizing trends that occur below the threshold, the autonomous science agent can make decisions about deployment that the competing algorithm can't. By being able to detect

changes with confidence the SPRT algorithm can reduce the rate of false positives, thus reducing waste of sampling resources.

The Relative Change algorithm is easily distracted by anomalies, resulting in a much higher false positive rate than the other algorithms. This behaviour could also be tuned to different scenarios, but again, it may need to be modified before every deployment. If the environment is truly unknown then this proposition is untenable.

In the simulated experiments we found that the Adaptive algorithm increases the number of changes detected, and reduces the error in estimating when the change occurs, compared to a fixed threshold algorithm. However, we found on the MVP data the improvements in performance were not maintained.

In the simulated 2D environment we found the SPRT algorithm was able to observe more maxima in the water maps than the Memory Threshold algorithm was, with potentially greater productivity. The number of maxima observed by the Adaptive Threshold was not statistically significantly different from the SPRT algorithm, but the Adaptive algorithm comes with the burden of additional parameters which must be tuned to the environment, the γ_{min} and number of suggested AIM deployments. We found with the MVP data that the γ_{min} parameter of the Adaptive Threshold algorithm still prevents sub-threshold activity from being detected.

It is difficult to recommend the use of the Adaptive algorithm, as the SPRT algorithm had better performance on efficiency, and because the Adaptive algorithm introduces more parameters than either the Memory Threshold or the Relative Change algorithm. There may certainly be benefits to encouraging the algorithm to deploy samples as a function of how much of the transect remains to be explored. To the author this seems to be conflating two notions, the first, determining how aggressively AIMs should be deployed, and second, what is the standard for an anomaly given environmental conditions. In our proposed algorithm we attempt to determine with some statistical confidence whether or not a change has occurred in the driving distribution, which seems to be a more direct solution to the problem.

Because the SPRT algorithm combines multiple measurements to calculate the likelihood that something has changed, rather than simply measuring their magnitudes independently, it is less dependent on prior knowledge for operation. With a reduced dependence on prior knowledge there should be greater confidence in performance when sending prospecting robots into unknown environments.

Chapter 5

Global Planning for Hypothesis Falsification

The previous chapters discussed methods for reactive autonomous science. Both the foraging and prospecting algorithms assumed that the trajectory to be followed already existed. In this chapter we develop an algorithm which selects goal locations in two-dimensional space, in order to collect samples that determine which of a set of competing hypotheses is most likely to be correct.

The hypotheses we consider in this chapter are functions that map observations in a domain, X, to a probability distribution over observations, Z. The task then set out for the learner is to collect sufficient data to determine the accuracy of a hypothesis, $h : X \to (Z \to [0, 1])$.

We present this work in the context of a physical robot moving to collect samples, however, the concepts should generalize to more abstract applications. Before we discuss the technical contributions of this chapter, we briefly frame the problem of selecting the most true of a set of hypotheses in the context of the philosophy of science.

To say a hypothesis is correct relies on having a notion of truth. In a world with noisy sensors and variable processes having an objective notion of truth can be challenging, to say the least. In this work we use F.P. Ramsey's notion of truth, which is a gambler's notion of truth. That is to say, the thing which makes the predictions we can gamble on with greatest reward is the truth (Ramsey, 1931).

Previous approaches to science autonomy generally considers zero or one hypothesis at a time. Informative path planning attempts to collect observations in the domain of a hypothesis in question in order to build accurate models. The prototypical example of this is to find the distribution of some quantity over a region of a map. The algorithms that are used to design the sampling points come with certain assumptions about the smoothness and the continuity of the hypothesis over the domain.

In many respects this mirrors the notion of confirmation theory or verificationism, due to Carnap (1936). Verificationism was a "bottom-up" theory of science (Hacking, 1983), the experimenter was expected to collect data and generate hypotheses which fit those data. The belief was that sufficient observations enable one to generate hypotheses about the world from true statements regarding empirical data. Hypotheses would be proven true by the irrefutable weight of the experimental data.

Really this does not seem an objectionable stance. The belief that with sufficient data one can uncover rules which generalize to unseen cases, is fundamental to the field of machine learning. One starts with a dataset and then attempts to identify the functions that best fit the data.

The space of functions searched is constrained by the representations used in the function learning process, and by the optimization and search techniques used to fit the function. Constraints, like regularization terms, are also added to mitigate overfitting, which is to say, to avoid functions that make accurate predictions where data has been collected, but that make wildly inaccurate predictions where data have not been collected.

Generalization is the holy grail of many a machine learning algorithm. The attempt to curtail overfitting, to ensure that predictions on unseen data are reasonable, is a pre-emptive defense against David Hume's problem of induction (Vickers, 2016). The problem of induction is that when extracting a law from observed data one only really knows that the law holds for those data, it says nothing about what may be encountered in the future.

Directly addressing the problem of induction is what motivates Karl Popper's scheme of falsificationism (Popper, 2005). Falsificationism is based on the notion that no predictive statement about the world can be proven true, it can only be proven false. That is to say, that we may uncover that its predictive power is less useful in all circumstances than previously thought.

Under Falsification we can only consider hypotheses that make predictions about the state of the world. Further, those predictions must be, either directly or indirectly, observable. Then we can take the hypothesis, test its predictions and assess its relative truth.

With the best performing hypothesis we can then go about the world making predictions and reaping our rewards. Take for example the progression from Kepler's Orbits to Galileo's, or Newton's physics to Einstein's. Of course, it is possible that there might be a set of hypotheses which have different levels of predictive power at different points in the domain. The prototypical example here would be relativistic and quantum physics. They remain stubbornly ununited, so we keep both and apply them where they are most effective.

This is all well and good, but the question stands: What use is all this and how does it help us change how robot scientists behave? To answer that we have to look at what the implications of verificationism and falsificationism are for experiment design.

The way to design experiments under a verificationist regime is to try to observe as many points in the domain, X, as possible. Greater information content is yielded by points that are farther away from each other, under whatever distance metric is used in the domain space. The logical end point of this thinking is to collect samples that are evenly spaced through the domain of the function. Under a falsification approach and in the case of testing only one hypothesis the end result is very similar. The objective is to take the predictions of the hypothesis and evaluate their accuracy. Then one must eventually test all parts of the hypothesis, leading to something resembling a uniform sampling across the domain of the hypothesis.

There may be more or fewer samples spent in locations correlated with the uncertainty in the predictions at that point. Sampling based on the uncertainty in the predictions of a function is a concept developed by Kristine Smith, (Smith, 1918), the progenitor of mathematical experiment design, who preceded Popper. The underlying notion here is that predictions which are certain require fewer experiments to falsify than those which are uncertain.

Where falsification becomes more interesting is in the setting where there are multiple competing hypotheses and one wants to determine the best one. What becomes relevant in this

setting is where the hypotheses disagree. Where hypotheses agree on their predictions they are all equally right or equally wrong, sampling in those regions does help assess their overall accuracy, but it does not help assess which of the hypotheses is better at making predictions. Points in the hypotheses' domains where they disagree in their predictions are exactly the points that one wants to sample to determine their relative accuracy.

If the goal is to assess the accuracy of different hypotheses, it is important to sample broadly in their domain in order to assess their overall performance. However, there is the additional goal of simultaneously determining which of a fixed set of hypotheses is most accurate, so we bias our search towards regions where hypotheses disagree. The technical contribution of this chapter is an algorithm which selects sampling points to learn the multinomial distribution of belief across the different hypotheses while assessing the overall quality of the set of hypotheses.

Broad sampling in the domain of the hypotheses that favours points of disagreement is the idea underpinning the algorithm developed in this chapter. Sampling only points of disagreement would address which of the set of hypotheses are most accurate, but it would not necessarily inform the overall accuracy of the proposed hypotheses.

On the other hand, while broad sampling in the domain of the competing hypotheses will help assess the overall accuracy of the hypotheses, a sampling approach which ignores the predictions of the hypotheses may well miss regions of disagreement in predictions, and thus be unable to distinguish the relative value of the hypotheses.

The proposed algorithm selects sample points in the domain of a set of competing hypotheses in order to determine which one of them is the most likely to be correct. To assess the overall assessment of the set of hypotheses, we introduce an additional, fictional hypothesis, H_0, which represents the belief that none of the proposed hypotheses are correct.

We believe that our algorithm represents a principled fusion of verificationist and falsificationist approaches to experiment design. Here we view falsification as sampling for information gain not in a distribution over the domain of a hypothesis, but information gain in the belief distribution over a set of competing hypotheses.

We test our algorithm in a simulated mission with a multi-hop lander that is trying to determine the accuracy of hypotheses that make predictions about some quantity that is not observable remotely. Such an activity might be drilling to access subsurface materials, or conducting laser ablation of the objects in the environment. The point is, travel to and interaction with the environment is required. The domain of the hypotheses in this setting is a 2D map of the terrain.

This type of mission would be typical of a mission to, for example, the lunar South Pole-Aitken Basin (Board et al., 2012). Here the geological scale that must be sampled is on the order of thousands of km, and as such would not readily admit exploration by a ground vehicle.

To further support the simulation findings we tested the algorithm in the Atacama desert as part of the ARADS project. The algorithm was deployed on the KRex2 robot where it chose locations to drill in order to localize subsurface halite.

The robot is testing a set of competing hypotheses that classify regions into one of two binary classes. Here we ignore any intermediary features that might be used to classify the terrain and simply test binary maps as black-box hypotheses regarding some model presented to the robot from scientists.

Such hypotheses could be classifications developed from remote sensing data, such as the presence of frozen water in lunar soil, or subsurface halite in the Atacama desert. Depending on

the models used to make predictions the same precursor data could generate multiple predictions about the underlying variable(s) of interest. The observations that robots collect on the surface help determine which of these hypotheses accurately predict the robots observations, in turn implying the relative accuracy of the hypotheses.

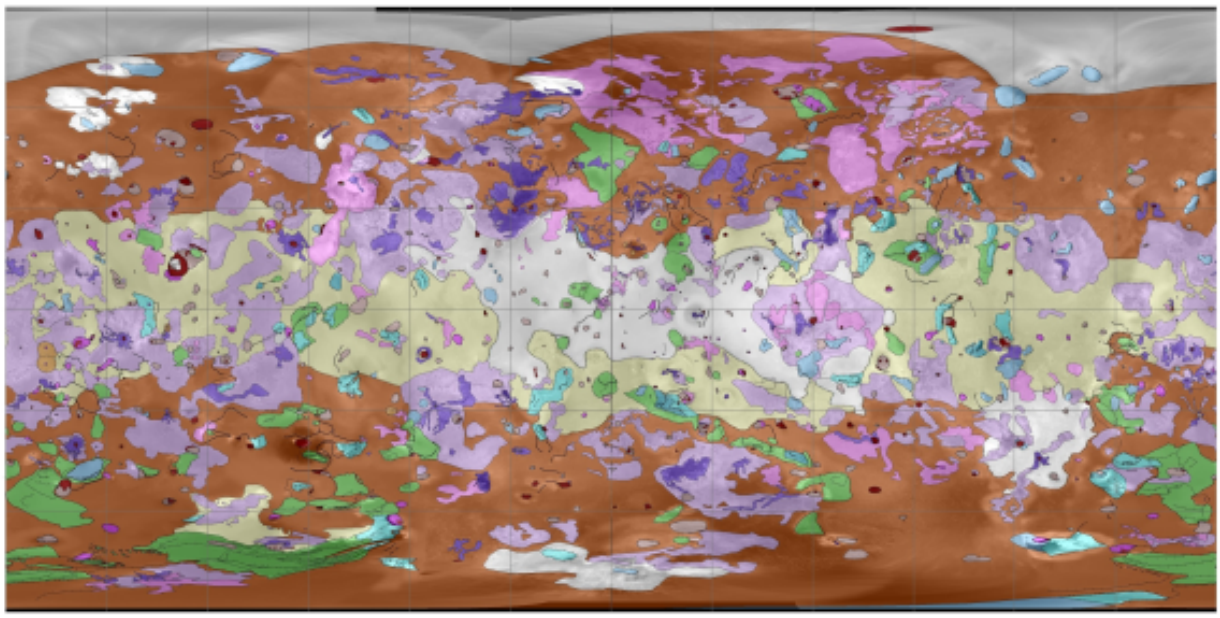


Figure 5.1: A geological map of Jupiter's Moon, Io. The remote data produced from observing Io has been interpreted as a map of terrain classifications. However, observations from the ground would be required to confirm the predictions of the map creators. Image courtesy wikipedia.org

We compare our algorithm to another algorithm which simply does mutual information sampling in the domain of the hypotheses. We find that our algorithm is statistically significantly better at identifying the best hypothesis out of the set than the competing algorithm when all hypotheses are good, and when all hypotheses are bad. We found that when the quality of the hypotheses were mixed the mutual information sampling in the domain had better performance than the proposed algorithm for some settings of sensor noise and sampling budget, but were otherwise statistically indistinguishable at a 95% confidence level.

Neither algorithm uses planners that attempt to be efficient with any resource other than samples. We use a greedy planner that simply hops to the next most informative site. The approach used in this chapter may not map to long-term ground vehicle operations. There is additional information to be gained by visiting locations between the identified goal points, which this algorithm ignores. If modified to address this short-coming, the falsification sampling algorithm could be used in ground vehicle scenarios.

Additionally, the proposed algorithm does not attempt to generate hypotheses from the observations it collects. It is an important part of future work to decide when to generate new hypotheses. To track the belief that none of the presented hypotheses are correct, we maintain a parameter, H_0. By keeping track of this quantity for both the mutual information sampling and

the proposed falsification sampling it was possible to represent the confidence that none of the proposed hypotheses are correct.

5.1 Prior Work

Science autonomy robots have not previously needed to choose between hypotheses, they have generally been focused on collecting data for remote operators. However, the design of experiments literature has been largely focused on selecting experiments that inform hypotheses. A popular example of an experiment selection technique is the multi-armed bandit.

Multi-armed bandits (MABs) (Robbins, 1952) are a formalization for sequentially selecting the most rewarding of a set of experiments. There are a number of different algorithms for approaching the MAB problem. The main families of which appear to Upper Confidence Bound algorithms (Lai and Robbins, 1985), the Gittins index (Gittins et al., 2011), and Thompson sampling (Thompson, 1933). Thompson sampling has recently gained attention for being simple to implement and for having near-optimal regret properties (Chapelle and Li, 2011; Agrawal and Goyal, 2012; Ortega and Braun, 2010).

Standard MAB settings assume the interaction with the candidate arms is simple. An arm is selected for testing, the arm is pulled, and a reward is delivered. In planetary exploration one can map candidate hypotheses to bandit arms, but executing tests is complex. Robots have to move through space and interact with the world, incurring unpredictable costs. Because targets of interest, like geologic materials, have highly spatially dependent distributions, sampling a random point is not guaranteed to be an informative action.

We look to science autonomy algorithms to determine how to select sample points. Most noteworthy of these algorithms is the OASIS/AEGIS (Castano et al., 2007; Estlin et al., 2012) system, as it has actually been deployed on Mars. However OASIS, like other prior work, either learns models from without prior data (Thompson, 2008) or uses proxy measures of importance that encode scientists' preferences (Paar et al., 2012; Chien et al., 2005), but is unaware of the hypotheses under examination. Given that access to data from precursor remote sensing is likely, starting each new scientific mission from a blank slate seems inefficient.

Girdhar et al. (2014)'s approach to exploration focusses on visiting locations that help construct a topic model for classifying satellite imagery. This approach continually improves the topic model the robot is constructing but it does not inform any hypotheses that scientists are attempting to resolve.

Thompson (2008) maximizes the diversity of collected samples with mutual information sampling. This approach ensures samples are distributed through the input space of the function it is learning. A shortcoming of this approach, and all approaches that rely on mutual information sampling using Gaussian processes (GP) with stationary kernel functions is that they do not depend on the observations collected. For a GP $P\left(f(x)|x_1, \ldots, x_t\right)$ the mutual information for a new data point, x_{t+1}, is proportional to:

$$I(x_{t+1}) \propto k(x_{t+1}, x_{t+1}) - \mathbf{k}_t(x_{t+1})^{\mathrm{T}} \mathbf{K}_t^{-1} \mathbf{k}_t(x_{t+1}) \tag{5.1}$$

where k is the kernel function. $\mathbf{k}_t$ is the vector of distances between the t collected data points and the test point x, and $\mathbf{K}_t$ is the $t \times t$ matrix of the distances between all previously collected

observation points. Mutual information sampling depends only on points in the input space that have been observed and the kernel function. While this is a principled approach, being aware of the hypothesis tested would require accounting for observations, $f(x)$.

The algorithm in (Thompson et al., 2015b) is aware of observations made on the ground. Its robot attempts to explain spectra data from satellites by collecting a library of spectra from the ground. Each pixel of satellite data is explained as a mixture of the end member spectra collected by the rover. The end members represent a basis of examples of "pure" minerals. The rover travels to locations where library of rover-observed spectra poorly explains the satellite data and collects more observations to explain the satellite data. This approach is implicitly constructing a hypothesis about the terrain composition but the algorithm itself does not test points in the satellite data that it considers well explained, nor does it consider alternate hypotheses.

Miller et al. (2016) use the expected value of the Fisher Information to determine points of interest. Like mutual information (Lindley, 1956), Fisher information is used as a score to select the most informative experiments. Their path planner produces smooth paths maximize the number of high information value observations. Fisher information and the mutual information are intimately related, but not identical, quantities. However, a rigorous comparison of the behaviour of robots maximizing mutual information and those maximizing expected Fisher information does not exist in the literature. Without such evidence it is difficult to select one reward function over the other and this should be studied further.

The above approaches to distributing sampling points throughout the input space are principled approaches to learning functions *de novo*. However, when trying to select between competing hypotheses it is conceivable that a sequence of observation points $x_1, \ldots, x_t$ will miss the points of conflict between those hypotheses, especially when sampling resources are limited. However, widely sampling an input space is important to ensure the predictions of a hypothesis are generally accurate. Most deployed science autonomy algorithms do not operate in the realm of hypotheses, they either focus on distributions of data or on hand-coded proxy measures, not necessarily the verification or falsification of hypotheses under consideration. For that reason we propose the method described in Section 5.2.

The work that probably bears the greatest similarity to the work contained in this chapter is Balcan's Agnostic Active Learning algorithm. Agnostic Active Learning, as developed in (Balcan et al., 2006), is an approach to selecting samples in order to determine which of a set of hypotheses most accurately models data. Through repeated sampling the algorithm reduces the number of hypotheses that are considered valid. The algorithm repeatedly takes two hypotheses under consideration, determines where they disagree and samples in the region of disagreement. This is a method consistent with the strategy of falsification laid out by Popper. Sampling where two hypotheses agree cannot help determine which of the two hypotheses is more correct.

Their work, as described in (Balcan et al., 2006) focused on linear functions and threshold functions, which makes determining the region of disagreement more straightforward than in the general case of arbitrary functions. In our work we use a sampling strategy to determine which regions of the domain are interesting. While this has the downside of potentially considering more unnecessary locations, it has the distinct advantage of not requiring knowledge of how to compute the region of disagreement between competing hypotheses. Further, while the A^2 algorithm iteratively compares only two hypotheses at a time, we consider the change in belief over all the hypotheses. This comes at a greater computational cost, but it helps simultaneously

re-evaluate more hypotheses.

The work contained in this chapter is an extension of the work presented in (Furlong, 2017). That work bears some resemblance to the work of Dorsa Sadigh et al. (2017). The work by Dorsa *et al.* attempts to select the optimal control strategy for a dynamic system by first simulating performance and then asking a human, playing the role of an oracle, to rate the performance of the algorithm. Both these algorithms are instantiations of the Thompson sampling algorithm.

The distinguishing factor between the work of (Dorsa Sadigh et al., 2017) and this work is where the former simply finds the best action at every iteration and then evaluates it. The work presented in this chapter identifies the sampling action, the question which would be posed to the oracle in (Dorsa Sadigh et al., 2017), that yields the greatest reduction in uncertainty in the belief over the competing hypotheses.

Similarly, Ravanbakhsh and Sankaranarayanan (2017) use Thompson sampling-like behaviour. They are attempting to find control laws for different dynamic systems. However, where the work above attempts to learn accurately the relative performance of different points in the action space Ravanbakhsh *et al.* simply need to find one control law that is successful and halts immediately.

There have been proposed some algorithms for active model selection. These algorithms start with a fixed set of hypotheses and attempt to resolve which one of them are most accurate, given an oracle they can use to label points

Madani et al. (2004) perform an empirical evaluation of active model selection algorithms. They consider a setting where the different hypotheses are modelled as coins, and the goal is to find the coin with the highest success rate. This differs from our problem in that the observations from one coin flip only informs the success rate of that coin. In our problem setting, the observation from conducting a sampling action informs all the hypotheses under investigation.

Madani et al. (2004) find that their "biased round-robin" sampling algorithm is one of the consistently highest-performing algorithms. Biased round robin cycles through the coins in turn, but does not move from one coin to the next until a tails has been observed. This bears some similarity to our algorithm which treats the hypothesis with the greatest success rate as the most likely hypothesis.

Sawade et al. (2010) present a method for active sampling that attempts to estimate the accuracy of one pre-trained classifier. They design a sampling distribution which is proportional to a function of the uncertainty of the candidate sampling point, x, given previous observations, σ_x^2, and the risk of the classifier, R_θ. R_θ is a constant for any classifier parameterization, θ.

The sampling distribution developed by Sawade *et al.* is given in Eq. (5.2). For a fixed parameterization of a classifier, this term is proportional to the uncertainty as a function of the candidate point. They assume that the estimate of uncertainty is provided by some other means, such as a Gaussian process.

$$q^*(x) \propto \sqrt{3\sigma_x^4 - 2R_\theta\sigma_x^2 + R_\theta^2} \tag{5.2}$$

Sampling candidate points based on the uncertainty function given by a Gaussian process was implemented for robotic planning in (Thompson, 2008). It is also interesting to note that for stationary Guassian kernels the estimate of uncertainty of a sampling point, x, does not depend on the observations collected at those locations.

Further, sampling locations based on uncertainty again echos the work of Smith (1918). In

broad strokes, the algorithm developed in (Sawade et al., 2010) is analogous to the default mutual information sampling algorithm which we compare against in our experiments.

Kumar and Raj (2016) present a method for estimating the accuracy of a single, trained classifier with a limited sampling budget. They use the classifier to label the data, and then use those labels to segment the data into strata. The goal was to estimate the accuracy of the classifier by requesting labels for the data in these strata.

They tested a number of different allocation techniques to distribute random samples to the different strata. They found that techniques which allocated samples proportional to the variance of the classifier accuracy within the strata performed well, but that an equal allocation of samples across the strata also had competitive performance.

Kumar and Raj (2016) use random sampling within the strata of data points. The algorithms they use are about deciding how many samples to draw from each location. It would be interesting to examine what happens when informative sampling is conducted within the strata.

Ali et al. (2014) present an active learning algorithm for simultaneously training and validating the different hypotheses, called Active Learning and Model Selection (ALMS). They consider candidate sampling points and score their relative value for either training the hypotheses or selecting the best hypothesis.

In order to determine how valuable a candidate point would be for training, the algorithm predicts the label of the point by averaging over the predicted labels from all the hypotheses. ALMS then re-trains the hypotheses with the previously collected training set, all but two elements of the validation set, and the candidate point with its estimated label. Leave-two-out validation is conducted with the remaining two data points, which have had labels determined by the oracle. The value of the candidate point is the average error in the cross-validation process, averaged by the current belief in the competing hypotheses.

The value of the candidate point for validation purposes is the average loss of the hypotheses trained with *only* the training dataset, computed using leave-two-out cross-validation on the entire validation set plus the candidate point with its estimated label. The training-value quantity is, in effect, asking which data point, if added to the training set, causes the worst performance for the validation of the hypotheses. On the other hand, the validation-value quantity is asking the question of which candidate point causes the greatest reduction in belief in the hypotheses.

If the expected reduction in performance from training with the data point is higher than the reduction in validation, then the worst performing candidate point is added to the training data set. If the expected reduction in validation is greater than the reduction in training, then an unbiased sample is drawn from the pool of candidate points, and added to the validation data set.

The algorithm recomputes the probability of the models given the dataset it is building over every time step, and uses that to determine the value of previously unobserved sampling points. It is necessary to recompute this probability before each sample is selected because the algorithm is simultaneously training the hypotheses with the data it collects while trying to validate them. In our approach, with fixed hypotheses, we don't need to re-train the hypotheses, so we can accumulate evidence with each data point selected.

The algorithm proposed by Ali et al. (2014) does not account for the probability that none of the proposed hypotheses are correct, something we do address in this chapter. This is an important quantity to keep track of for future algorithms which will be not only searching for candidate sample points, but also new candidate hypotheses.

5.2 Method

The objective of the mission is to build belief in the hypothesis that is least false, H^*. In order to do this the robots must select sampling locations, $l \in L$, in the input space of the hypotheses that most productively inform the investigation. The robots collect true observations, $M(l)$, from the true map, M, that can be used to update the belief in the different hypotheses based on the prediction $H_i(l)$ from the corresponding hypothesis H_i. The hypotheses the robots are attempting to validate were generated by corrupting true map data. This is a stand-in for hypotheses that have different interpretations of the precursor data.

5.2.1 Belief in Hypotheses

The algorithms estimate their belief that hypothesis H_i, where $i \in 1, \ldots, K$, with a multinomial distribution, $P(H_i)$, where $\sum_{i=0}^{K} P(H_i) = 1$. We have a special place-holder hypothesis H_0, which represents the probability that none of the proposed hypotheses are correct.

We place a Dirichlet prior on the distribution of belief in the hypotheses, with corresponding condensation parameters $\alpha_0, \ldots, \alpha_K$. The condensation parameters are initialized with an uninformative prior, $\alpha_i = 1$, as initially all hypotheses are considered equally likely. However, the condensation parameters can be initialized to reflect any prior belief in the competing hypotheses.

After an observation the condensation parameters are updated to reflect the agreement with the predictions from the different hypotheses. That is to say:

$$\alpha_{i,t+1} = \alpha_{i,t} + P(H_i(l_t) = M(l_t)|H_i) \tag{5.3}$$

The observations the robot makes in these experiments have sensor noise. However, we assume that the hypotheses themselves do not make probabilistic predictions. This mirrors the author's experience working with planetary scientists, but dealing with probabilistic predictions should be addressed in future work.

5.2.2 Site Selection Algorithm

Algorithm 5.1 selects sampling sites with a modified version of Thompson sampling. At each time step, t, the algorithm samples a belief state $\vec{\theta}$ from the Dirichlet prior described in Section 5.2.1, and then chooses the H_i that has the largest belief θ_i. The algorithm assumes that H_i is "true" for the duration of this step, and uses it to evaluate which location l_t is most informative. At the location l_t the robots collect the observation $M(l_t)$. Observations are used to update the belief in the hypotheses that have been assigned to them.

We assume the robot starts in the $(0, 0)$ position on the map, which corresponds to the top left-hand location of the maps shown in Figure 5.2. The robot is capable of sampling at locations l in the input space L, which in this experiment are (x, y) locations on a two-dimensional map with integer coordinates. A Robot at location l_t that chooses to travel to l_{t+1} incurs a cost of $cost(l_t, l_{t+1})$. We assume that $cost(l_1, l_2) = 0$ when $l_1 = l_2$, and $cost(l_1, l_2) > 0$ when $l_1 \neq l_2$.

Once a location has been selected, the robot travels to the specified location and collects an observation from the actual map, $M(l_t)$, and the belief state is updated. After each observa-

tion the place-holder condensation parameter α_0 is incremented with the belief that none of the hypotheses predict the observation from the map at l_t.

The $reward$ function for candidate sampling locations is how the competing algorithms control the vehicle behaviour. The reward functions for the mutual information domain sampling algorithm and our proposed falsification sampling algorithm are specified in Sections 5.2.2 and 5.2.2.

Algorithm 5.1 Site Selection Algorithm

function SAMPLE-SELECTION(M,$\langle H_1, \ldots, H_K \rangle$,$budget$)
  $\alpha_i \leftarrow 1 \, \forall \, i \in \{0, \ldots, K\}$
  $t \leftarrow 0$
  $l_0 \leftarrow \langle (0,0) \rangle$
  repeat
    $\vec{\theta} \sim \text{Dirichlet}\,(\alpha_0, \alpha_1, \ldots, \alpha_K)$
    $i \leftarrow \underset{i}{\text{argmax}} \quad \theta_i$
    $l_t \leftarrow \underset{l}{\text{argmax}}$ reward($\mathbb{E}\left[\vec{\theta}\right], H_i, \langle l_0, \ldots, l_{t-1} \rangle$)
    for $i \leftarrow 1, K$ **do**
      $\alpha_i \leftarrow \alpha_i + \mathbb{1}(M(l_t) = H_i(l_t))$
    end for
    if $\sum_i^K \mathbb{1}(M(l_t) = H_i(l_t)) = 0$ **then**
      $\alpha_0 \leftarrow 1$
    end if
    $budget \leftarrow budget - cost(l_t)$
    $t \leftarrow t + 1$
  until budget = 0
end function

Control Algorithm - Mutual Information Domain Sampling

For the control algorithm we build a density estimator using a Gaussian kernel function and we select the candidate point where the increase in information would be the greatest. We compute the entropy, $\mathbb{H}(\cdot)$ of the kernel density estimator as described by Beirlant et al. (1997). The algorithm for the reward function is given in Algorithm 5.2.

Algorithm 5.2 Mutual information domain sampling. This algorithm uses a measure of mutual information like the one used in (Thompson, 2008) or (Sawade et al., 2010). It depends only on previously observed locations to find the most informative point.

function SPATIAL-REWARD($P\left(\vec{\theta}\right)$,$H_i$,$\langle l_0, \ldots, l_{t-1} \rangle$,$l_t$)
  $P_t(l) \leftarrow KDE(\langle l_0, \ldots, l_{t-1} \rangle)$
  return $\mathbb{H}(P_t) - \mathbb{H}(P_t | l_t)$
end function

The information gain computed by this algorithm is with respect to the domain of the hypotheses, given all previous points in the domain where observations were collected. The kernel density estimator used for computing mutual information in the domain makes assumptions about the smoothness of the underlying function.

This algorithm is analogous, but not identical to, that proposed in (Sawade et al., 2010). Previously, we observed that algorithms like that of (Sawade et al., 2010) did not necessarily consider the predictions made by the different functions. Consequently, by following a sampling method which does not strictly depend on the predictions of any one hypothesis, we can, in effect, simultaneously learn the accuracy of all three hypotheses. As we will see in the results, this algorithm does perform well in the experiments we conduct.

Proposed Algorithm - Hypothesis Falsification Sampling

This algorithm seeks sampling locations that concentrate belief across the hypotheses given observation $H_i(l_t)$, as seen in Equation 5.4, where H_i is considered the "true" hypothesis as per Algorithm 5.1. These locations maximize the mutual information between the belief state $\vec{\theta}$ and the observation $H_i(l_t)$.

$$I\left(\vec{\theta}; H_i\left(l_t\right)\right) = \mathbb{H}\left[\vec{\theta}\right] - \mathbb{H}\left[\vec{\theta}|H_i(l_t)\right] \tag{5.4}$$

This will automatically seek out locations that maximize the change in the belief distribution over the hypotheses. With Equation 5.4 as the reward function at any point l where the H_i's agree will increase the credibility of all hypotheses, resulting in negative mutual information for l_t.

However, one also wants to ensure that the hypotheses are accurate. Sampling at disagreement points will help build the credibility of the best hypothesis, but it does not give confidence with the overall predictions of H_i. To mitigate this problem we add to Equation 5.4 the result computed in Algorithm 5.2 to ensure the sampled points are spatially diverse. The reward function used for this algorithm is given in Algorithm 5.3.

We can derive this algorithm as follows:

$$
\begin{aligned}
\mathbb{H}(\theta_i, L) &= -\sum_{i}^{N}\sum_{l\in L} p(\theta_i, l)\log\left(p(\theta_i, l)\right) \\
&= -\sum_{i}^{N}\sum_{l\in L} p(\theta_i|l)p(l)\left[\log\left(p(\theta_i|l)p(l)\right)\right] \\
&= -\sum_{i}^{N}\sum_{l\in L} p(\theta_i|l)p(l)\left[\log\left(p(\theta_i|l)\right) + \log\left(p(l)\right)\right] \\
&= -\sum_{i}^{N}\sum_{l\in L} \left[p(\theta_i|l)p(l)\log p(\theta_i|l)\right] - \sum_{i}^{N}\sum_{l\in L}\left[p(\theta_i|l)p(l)\log p(l)\right] \\
&= -\sum_{i}^{N}\sum_{l\in L} \left[p(\theta_i|l)p(l)\log p(\theta_i|l)\right] - \sum_{l\in L}\sum_{i}^{N}\left[p(\theta_i|l)p(l)\log p(l)\right] \\
&= -\sum_{i}^{N}\sum_{l\in L} \left[p(\theta_i|l)p(l)\log p(\theta_i|l)\right] - \sum_{l\in L}\left[\left[\sum_{i}^{N} p(\theta_i|l)\right] p(l)\log p(l)\right] \\
&= -\sum_{i}^{N}\sum_{l\in L} \left[p(\theta_i|l)p(l)\log p(\theta_i|l)\right] - \sum_{l\in L}\left[p(l)\log p(l)\right] \\
&= -\sum_{i}^{N}\sum_{l\in L} \left[p(\theta_i|l)p(l)\log p(\theta_i|l)\right] + \mathbb{H}(L) \\
&= -\sum_{l\in L}\sum_{i}^{N} \left[p(\theta_i|l)p(l)\log p(\theta_i|l)\right] + \mathbb{H}(L) \\
&= \mathbb{E}_L\left[-\sum_{i}^{N} p(\theta_i|l)\log p(\theta_i|l)\right] + \mathbb{H}(L) \\
&= \mathbb{E}_L\left[\mathbb{H}(\theta_i|L))\right] + \mathbb{H}(L)
\end{aligned}
$$

Allowing us to conclude that

$$\mathbb{H}(\theta_i, L) = \mathbb{E}_L\left[\mathbb{H}(\theta_i|L)\right] + \mathbb{H}(L) \tag{5.5}$$

The expectation term of Equation 5.5 can be difficult to compute as the space of L increases. However, since mutual information is the difference before and after the observation the computation can be simplified.

$$MI = \mathbb{H}(\theta_i, L) - \mathbb{H}(\theta_i, L|z(l_t), \xi = l_t) \tag{5.6}$$

Substituting the derived value for entropy from Equation 5.5 into Equation 5.6 we can write

$$MI = \mathbb{E}_Z\left[\mathbb{H}\left(\theta_i|L=l_t\right)\right] - \mathbb{E}_Z\left[\mathbb{H}\left(\theta_i|L=l_t, z(l_t), \xi=l_t\right)\right] + \mathbb{H}\left(L\right) - \mathbb{H}\left(L|z(l_t), \xi=l_t\right) \quad (5.7)$$

$\mathbb{H}\left(L\right) - \mathbb{H}\left(L|z(l_t), \xi=l_t\right)$ is the information gained by sampling at a location l_i in the domain, given the prior sampling locations. The term $\mathbb{E}_Z\left[\mathbb{H}\left(\theta_i|L\right)\right] - \mathbb{E}_Z\left[\mathbb{H}\left(\theta_i|L, z(l_t), \xi=l_t\right)\right]$ permits some opportunities for simplification that hinge on how much of the hypothesis an observation effects. We make the naive assumption that conditioned on the map location, observations are independent of all other locations.

Algorithm 5.3 Hypothesis falsification sampling. We use Eq. (5.4) to bias the value of sites towards locations where the hypotheses disagree.

function DISAGREE-REWARD($P(\vec{\theta})$,H_i,$\langle l_0, \ldots, l_{t-1}\rangle$,$l_t$)
  $P(\vec{\theta}|H_i(l_t)) \leftarrow$ update($P(\vec{\theta})$, $H_i(l_t)$)
  $r_h \leftarrow \mathbb{H}\left[\vec{\theta}\right] - \mathbb{H}\left[\vec{\theta}|H_i(l_t)\right]$
  $r_s \leftarrow$ spatial-reward($P(\vec{\theta})$, H_i, $\langle l_0, \ldots, l_{t-1}\rangle$, l_t)
  return $r_h + r_s$
end function

In this experiment we directly add the information gained from the spatial distribution and the hypothesis belief distribution with an equal weighting term of 1. The relative importance of the two terms could be modified in order to produce behaviour that either more aggressively targets differences in predictions (increase weight on r_h), or prioritizes determining overall accuracy (increase weight on r_s).

At this juncture it is important to recognize that the proposed algorithm is implicitly making naive assumptions about the relationships between the information gained by sampling in the domain and the information gained in the belief distribution over the different hypotheses. In effect, our algorithm may double count some of the information gained by making assumptions about independence that do not necessarily hold.

It remains for future work to construct a more formal representation which explicitly incorporates the relationship between sampling in the domain of the hypotheses and sampling in the belief over the hypotheses. With the correct representation, the spatial diversity should be achieved while attempting to maximize the information gained in the belief distribution over the hypotheses.

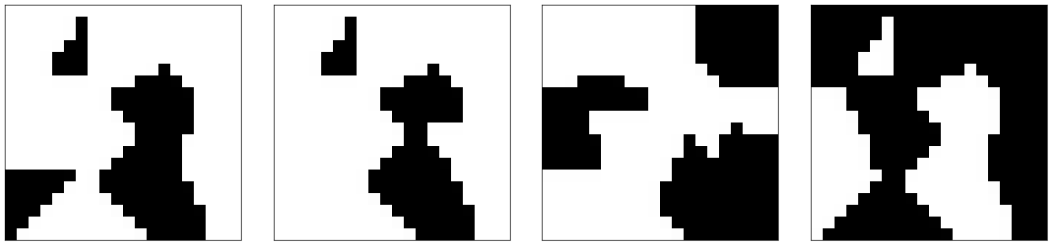
5.2.3 Map Generation

We generated 10 different maps, each 20×20 pixels, with each pixel containing a label from $z \in 1, \ldots, N$. We produce the ground truth maps by selecting 20 seed locations and randomly assigning them a label z with uniform probability (denoted by the function $U(\cdot)$), over the N labels. We then use a Voronoi map generation algorithm, given in Algorithm 5.4. Example maps are shown in Figure 5.2a.

To generate the hypotheses maps we take the same 20 seed points used to generate a true map and mislabel the seed locations with probability $P(z=i|z=j) = \epsilon/(N-1)$ for all $i \neq j$,

Algorithm 5.4 Map Generation Algorithm

function GENERATE-MAP(numSeeds,numPixels,numLabels)
  $seeds \leftarrow \langle \cdot \rangle$
  $map \leftarrow zeros(numPixels, numPixels)$
  for $i \in 1, \ldots, numSeeds$ **do**
    $x_i \sim U(numPixels)$
    $y_i \sim U(numPixels)$
    $label \sim U(numLabels)$
    $seeds \leftarrow seeds + \langle (x_i, y_i, label) \rangle$.
  end for
  for $x \in 1, \ldots, numPixels$ **do**
    for $y \in 1, \ldots, numPixels$ **do**
      $map(x, y) \leftarrow closest(seeds, x, y).label$
    end for
  end for
  return map
end function



(a) The True map for M_0.

(b) H_1,93.25% similarity with M_0, $\epsilon = 0.1$.

(c) H_4, 49% similarity with M_0, $\epsilon = 0.5$.

(d) The H_7, 7.25% similarity with M_0, $\epsilon = 0.9$.

Figure 5.2: Examples of a map and the hypotheses generated that attempt to predict the map.

$i, j \in 1, \ldots, N$. Using the corrupted seeds we then generate maps using the Algorithm 5.4. Figures 5.2b, 5.2c,5.2d shows the effect of maps as ϵ increases. We tested hypotheses generated with $\epsilon \in \{0.1, 0.2, 0.3, 0.5, 0.7, 0.8, 0.9\}$ which correspond to H_1 to H_7 in Table 5.1. Because the maps are binary, we set $N = 2$ for these experiments.

5.2.4 Experiments

We compared the performance of our proposed algorithm and of the control algorithm in three experiments. For each experiment we tested ten different maps, and each map was repeated for twenty trials. The true maps and the hypothesis maps were generated as described in Section 5.2.3. The similarity between the true map and the hypotheses is given in Table 5.1.

True Maps	H_1	H_2	H_3	H_4	H_5	H_6	H_7
M_0	0.93	0.82	0.66	0.49	0.26	0.14	0.07
M_1	0.91	0.86	0.70	0.53	0.24	0.11	0.02
M_2	0.90	0.79	0.73	0.62	0.23	0.19	0.19
M_3	0.95	0.86	0.77	0.36	0.25	0.23	0.00
M_4	0.98	0.81	0.74	0.67	0.23	0.20	0.00
M_5	1.00	0.83	0.69	0.45	0.29	0.18	0.08
M_6	0.98	0.91	0.86	0.42	0.20	0.17	0.06
M_7	1.00	0.80	0.79	0.55	0.38	0.16	0.05
M_8	0.94	0.77	0.76	0.62	0.30	0.10	0.07
M_9	0.99	0.92	0.76	0.42	0.20	0.14	0.00

Table 5.1: This table gives the similarity between the true maps M_i and the hypotheses proposed by the simulated scientists H_j. The similarity score for each map is the fraction of locations in the hypothesis map which agree with the true map.

When a robot travelled to a point on the map it was informed of the true classification of the point subject to sensor noise. Since the observations were binary, we considered bit-flip noise models. Our noise model was symmetric, with $P(\text{Error}) \in \{0, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$.

We assume in this experiment that the robot is capable of hopping long distances. As such the cost of traversal is simply limited to the number of hops. The robots were given a budget of $T \in \{25, 50, 100, 150, 200\}$ hops. Since the mission budget is in number of hops the cost function is the constant function $cost(l_t, l_{t+1}) = 1$. The budget and cost function can be modified for different mission settings.

Experiment	$\mathbf{H}^*$	**Hypotheses**		
1 - All Good Maps	H_1	H_1	H_2	H_3
2 - Mixed Quality Maps	H_1	H_1	H_4	H_7
3 - All Bad Maps	H_0	H_5	H_6	H_7

Table 5.2: The hypotheses tested in the experiments. Column $\mathbf{H}^*$ gives the best hypothesis for each experiment. Subscripts correspond to Table 5.1.

In Experiment 1 the hypotheses are all of reasonable quality. This represents a mission with good precursor data and generated hypotheses. In Experiment 2 the hypotheses are of mixed quality. In this scenario the precursor data is ambiguous, but there is one good hypothesis. In Experiment 3 all hypotheses are of poor quality. In this scenario the hypotheses do not describe the environment accurately. The parameters used are given in Table 5.3.

Table 5.3: The parameter settings tested in the experiments in this chapter.

Parameter	**Settings**
Budget	$T \in \{25, 50, 100, 150, 200\}$
Sensor Noise	$P(\text{Error}) \in \{0, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$

Analysis of Results At the end of the experiments we compute mean of $P(H^*)$. Across the ten different maps we computed the effect size of our intervention. To determine the effect size we use Cohen's d, with pooled standard deviation, σ_{pooled}.

$$d = \frac{\mu_1 - \mu_0}{\sigma_{pooled}} \tag{5.8}$$

$$\sigma_{pooled} = \sqrt{\frac{\sigma_1^2 + \sigma_0^2}{2}} \tag{5.9}$$

where μ and σ are the sample mean and standard deviation of $P(H^*)$ for the mutual information (0) and falsification sampling (1) algorithms, respectively. Cohen's d can be divided into four levels of effect size: $d < 0.2$ is a negligible effect, $0.2 < d < 0.5$ is a small effect, $0.5 < d < 0.8$ is a medium effect, and $d > 0.8$ is a large effect. Negative values of d mean that the control algorithm (domain mutual information sampling) has a higher belief in H^*. Lipsey et al. (2012) gives criticisms of using these thresholds for Cohen's d, but the dearth of effect size reporting in robotics does not provide alternative thresholds for effect size magnitudes.

5.3 Results

We present the results based on the quality of hypotheses being tested. These experiments are for "All Good", "Mixed Quality", and "All Poor" hypotheses. For each condition there is a predetermined "best" hypothesis for the algorithm to select. We report the change in probability mass assigned to the best hypothesis.

In the tables below we denote whether statistical significance has been achieved with $p \geq 95\%$. We use paired Bayesian significance testing, as per (Bååth, 2014). To report effect size we use Cohen's d (Lipsey et al., 2012).

In plots of Cohen's d (Fig. 5.3 to Fig. 5.5) regions of white indicate that there was no statistical significance between the performance of the algorithm. Where the maps of Cohen's d are blue, it means that the mutual information sampling algorithm is better than the falsification sampling. Where the maps of Cohen's d are red, the falsification algorithm has greater belief in the best hypothesis.

5.3.1 Experiment 1 Results - All Good Hypotheses

In this experiment the correct hypothesis to select, H^*, is H_1. As we can see in Fig. 5.3, for all budget and error rates, the falsification algorithm is as good as or better than the mutual information sampling algorithm, although the change in performance is quite small.

Curiously, while the greatest difference in performance occurs for sampling budgets of 25, and an error rate of approximately 0.05, the greatest effect size occurs for larger budgets and moderate noise levels ($P(error) \in [0.1, 0.3]$). What this means is that the variability in performance is reduced, relative to the change in performance.

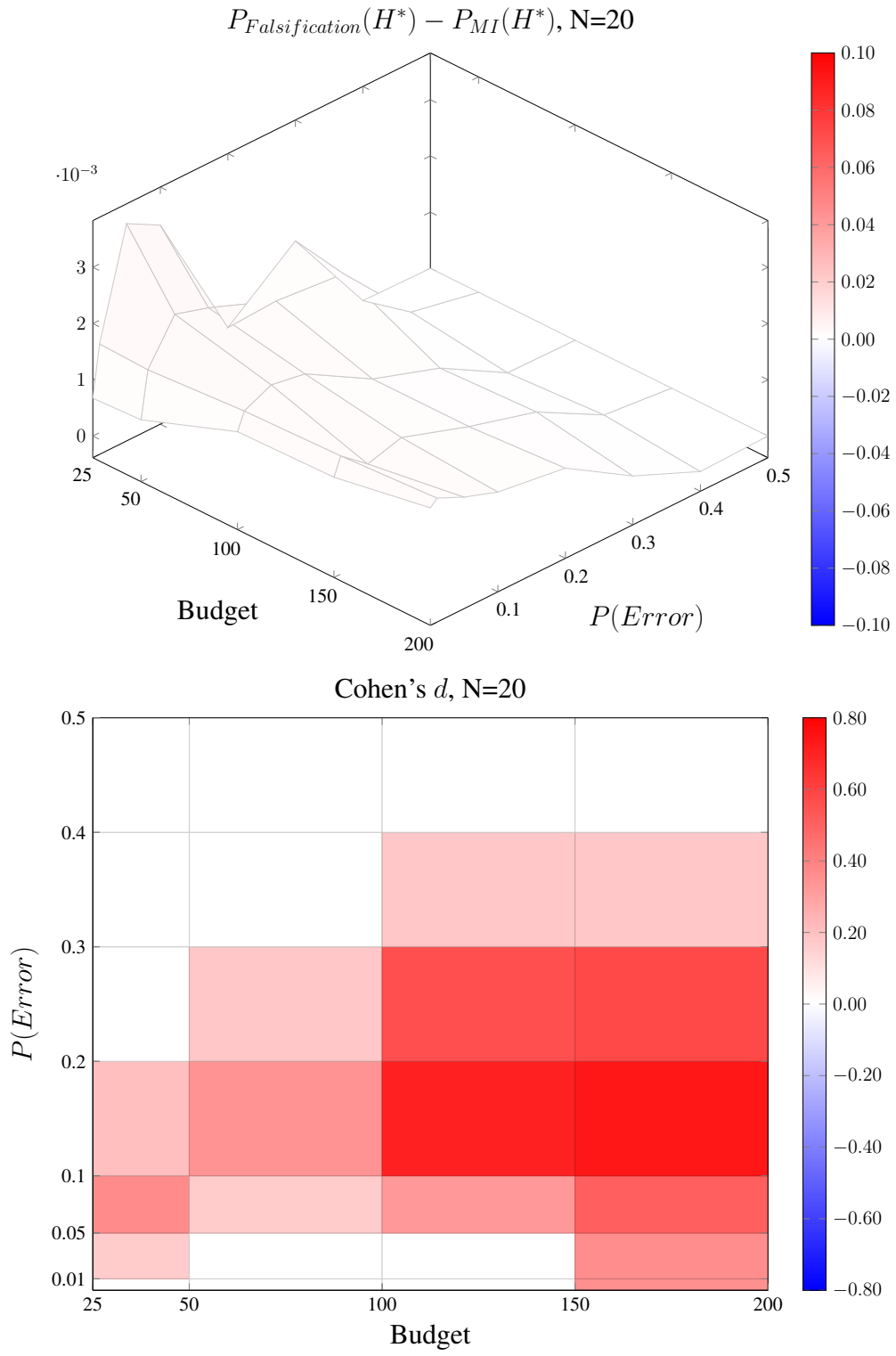


Figure 5.3: All Good Hypotheses - The difference in belief in the best hypothesis, H^*, and the effect size of that difference. Underlying data is found in Section E.1.1 (performance data) and Section E.2.1 (effect size data).

In this condition we can say with confidence that the falsification algorithm has better performance. However, we should observe that the magnitude of difference performance is relatively small.

The relative performance and the size of the effect of using the falsification algorithm can be found in Section E.2.1. The belief distributions learned by the two algorithms can be seen in Section E.1.1. Note that once the error rate exceeds 20%, both algorithms begin to lose the ability to identify the best hypothesis.

5.3.2 Experiment 2 Results - Mixed Quality Hypotheses

In this experiment the correct hypothesis to select, H^*, is H_1. As we can see in Fig. 5.4, the mutual information algorithm produces greater belief in the best hypothesis, especially for small sample sizes.

In this case only when the budget is large and the noise is moderate does the falsification algorithm perform better than the mutual information sampling algorithm.

The relative performance and the size of the effect of using the falsification algorithm can be found in Section E.2.2. The belief distributions learned by the two algorithms can be seen in Section E.1.2. Note that once the error rate exceeds 40%, both algorithms begin to lose the ability to identify the best hypothesis. This is a higher rate than when all hypotheses were of high quality.

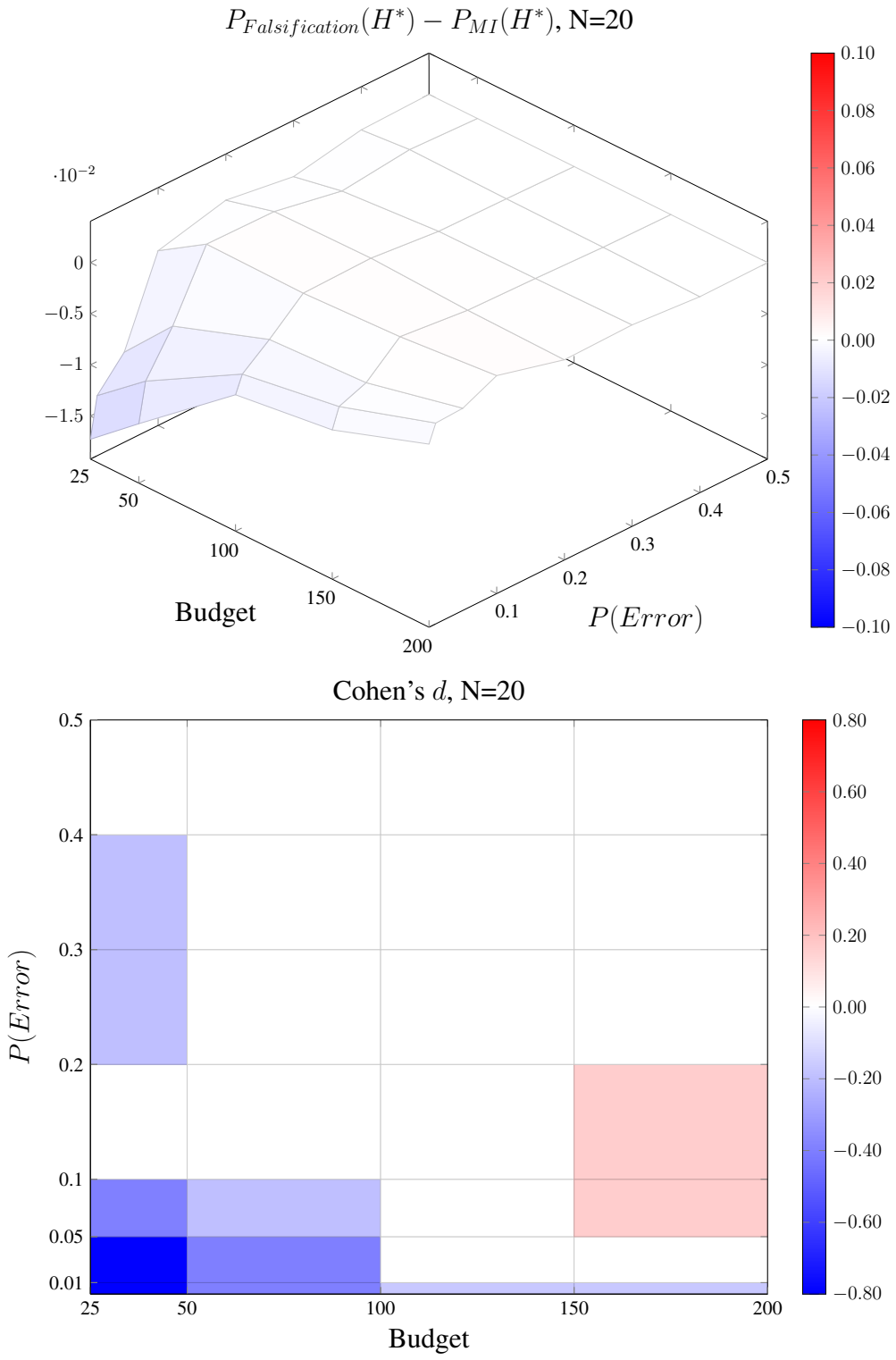


Figure 5.4: Mixed Quality Hypotheses - In this case the standard mutual information sampling algorithm outperforms the falsification algorithm for a wide range of parameter settings. Data supporting these plots are found in Section E.1.2 (performance data) and Section E.2.2 (effect size data).

5.3.3 Experiment 3 Results - All Bad Hypotheses

In this experiment the correct hypothesis to select, H^*, is H_0 - the possibility that none of the proposed hypotheses are correct. In Fig. 5.5, the falsification algorithm generally performs better than the mutual information algorithm, except when the sampling budget is small (< 50) and the noise level is moderate to large, ($P(\text{Error}) \in [0.2, 0.4]$).

The relative performance and the size of the effect of using the falsification algorithm can be found in Section E.2.3. Where the falsification algorithm has better performance, it has a very large effect size, $d \geq 1.0$.

The belief distributions learned by the two algorithms can be seen in Section E.1.3. There are two modes of degradation. First, once the error rate in the sensor reaches about 20%, the algorithms both select the best hypothesis of all three poor hypotheses, instead of selecting the hypothesis that none of the proposed hypotheses are correct. Then, once the error rate exceeds 40%, both algorithms loose the ability to distinguish between any of the proposed hypotheses.

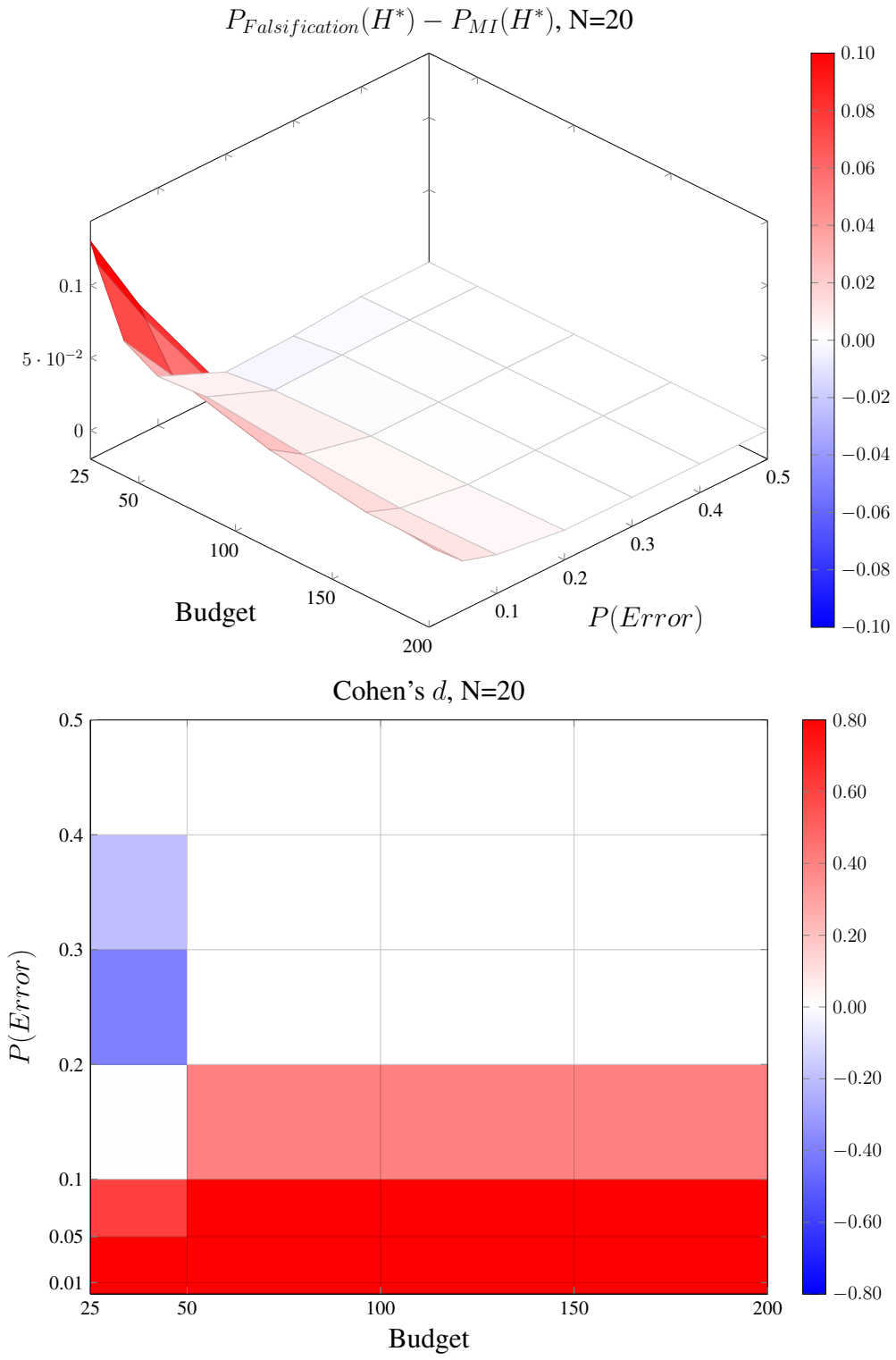


Figure 5.5: All Poor Hypotheses - The falsification algorithm performs better than the standard mutual information algorithm, with the exception of small budget sizes and moderate to high error rates. The data supporting these plots can be found in Section E.1.3 (performance data) and Section E.2.3 (effect size data).

5.4 Deployment in Chile

To demonstrate the proposed algorithm, we deployed it on the KRex2 robot in Chile (Figure 5.6) as part of the 2017 field season of NASA Ames' ARADS project. Planetary scientists who were part of the project were interested in mapping subsurface halite deposits. The scientists considered a location around a region they called "the pit", which was located, in UTM coordinates, at Zone 19J 396528.98 E 7334167.93 S.

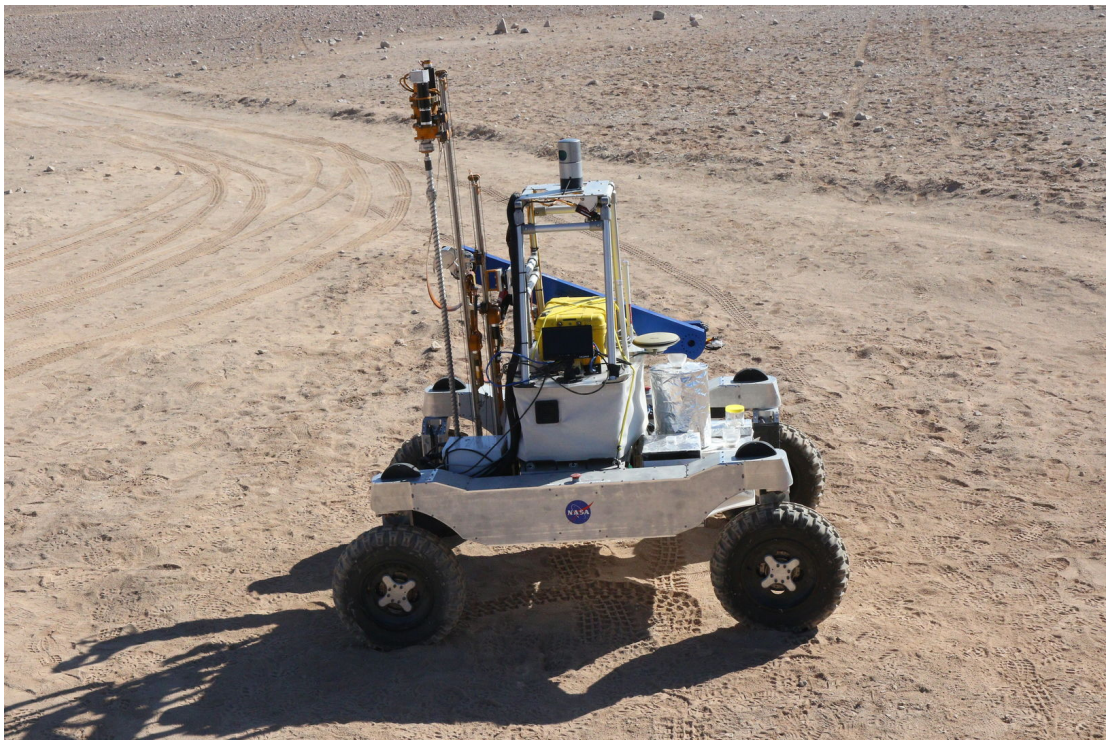
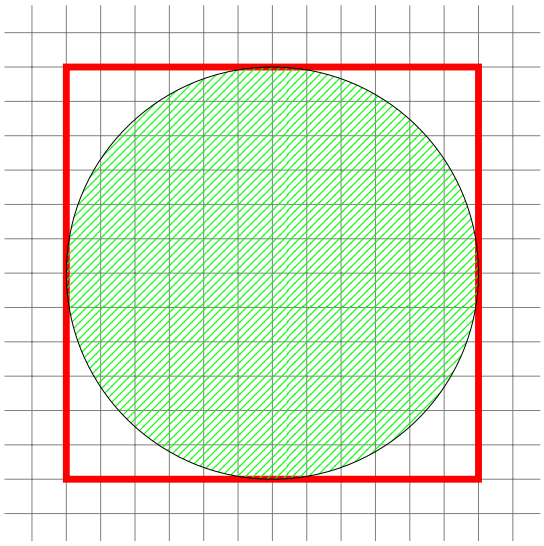


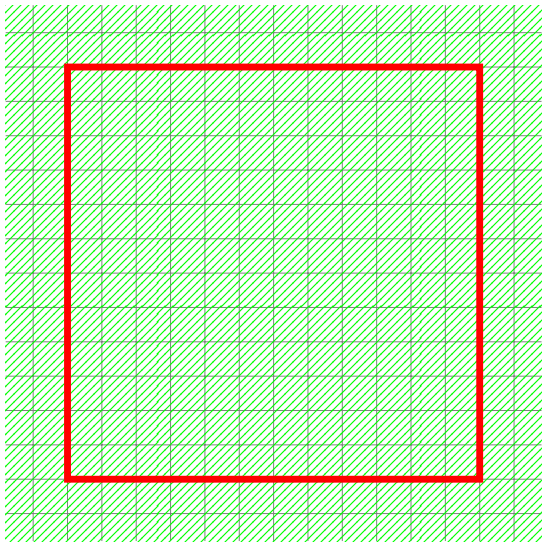
Figure 5.6: The IRG robot KRex2 with a mounted drill and sample collection arm exploring the Atacama desert in Chile. The drilling process could take up to four hours to drill a 2m deep hole, including time to operate sample collection arm.

The scientists were interested in two competing hypotheses about the distribution of subsurface halite. The first hypothesis (H_1) was that the halite was a deposit resulting from a dried body of water of approximately 30m in diameter, centred at the pit. The competing hypothesis was that there is a uniform distribution of halite in and around the pit (H_2). We maintained both these hypotheses along with the belief that neither hypothesis was correct. The domain of operation was restricted to a 30m square space centred on the pit. The two hypotheses are illustrated in Fig. 5.7.

The subsurface halite was detected using a rotary-percussive drill developed by Honeybee, Inc. (Bergman et al., 2016). The halite was detected by having a human operator observe the current draw of the primary motor in the drill. When current draw had crossed a threshold the human observer determined that halite had been encountered. Obviously this is a less than ideal sensor and not rigorously calibrated. However, laboratory analysis later determined that halite was recovered from the holes where the current draw indicated that halite was discovered.



(a) H_1 - Halite distributed under a dried pond.



(b) H_2 - Uniform halite distribution

Figure 5.7: The two hypotheses the ARADS planetary scientists were interested in investigating. The red line represents the restricted domain of operation. The green hatching pattern represents where the hypotheses would predict halite would be present.

It took approximately one hour to reach a depth of 1m. Since the halite was generally known to be at a depth of 1.5m in the area holes were drilled to a depth of 2m. This took approximately two hours to drill to this depth, with additional time to extract the drill from the hole after a maximum depth was reached. Five drilling sites were identified by the algorithm and were then drilled before the allotted time for the exercise was exhausted. Those locations and the findings from the drilling actions are given in Table 5.4.

We discretized the operations area into a 1m resolution grid. At each time step the algorithm considers which point in the grid would be most informative about the two hypotheses. When a grid point is selected the robot travels there and commences drilling. The presence of halite was determined by the human observer, and this observation was fed into the algorithm. The holes were drilled in the order of the hole numbers given in Table 5.4.

Table 5.4: The resultant observations from drilling operations in the Atacama desert. Holes were selected in the order given by the column titled "Hole".

Hole	Commanded Hole Coordinates (UTM)	In Circle?	Observation
1	Zone 19J 396515 E 7334151 N	No	Halite present
2	Zone 19J 396548 E 7334182 N	No	Halite not present
3	Zone 19J 396514 E 7334182 N	No	Halite not present
4	Zone 19J 396543 E 7334153 N	No	Halite not present
5	Zone 19J 396531 E 7334166 N	Yes	Halite present

In Fig. 5.8 we can see the belief in the different hypotheses change as a function of the holes

drilled. As the number of holes increases, the belief in H_1 is increased, supported by the evidence collected from the drill samples. The locations selected by the algorithm focused on where the competing hypotheses disagreed and focused its initial samples there. The final sampling point was located at the centre point of the domain. Presumably the component of the reward function that attempts to maximize the diversity of the algorithm forced the sample point to be the location furthest from the other samples while still staying within the domain of operation.

On the whole the behaviour of the algorithm in the field was consistent with the performance demonstrated in the preceding experiments. The amount of time it took to drill a hole and the restricted operation time for this portion of the field season drastically limited the number of holes that could be drilled. However, even though the number of samples was drastically limited compared to the simulation experiments the algorithm still behaved accordingly.

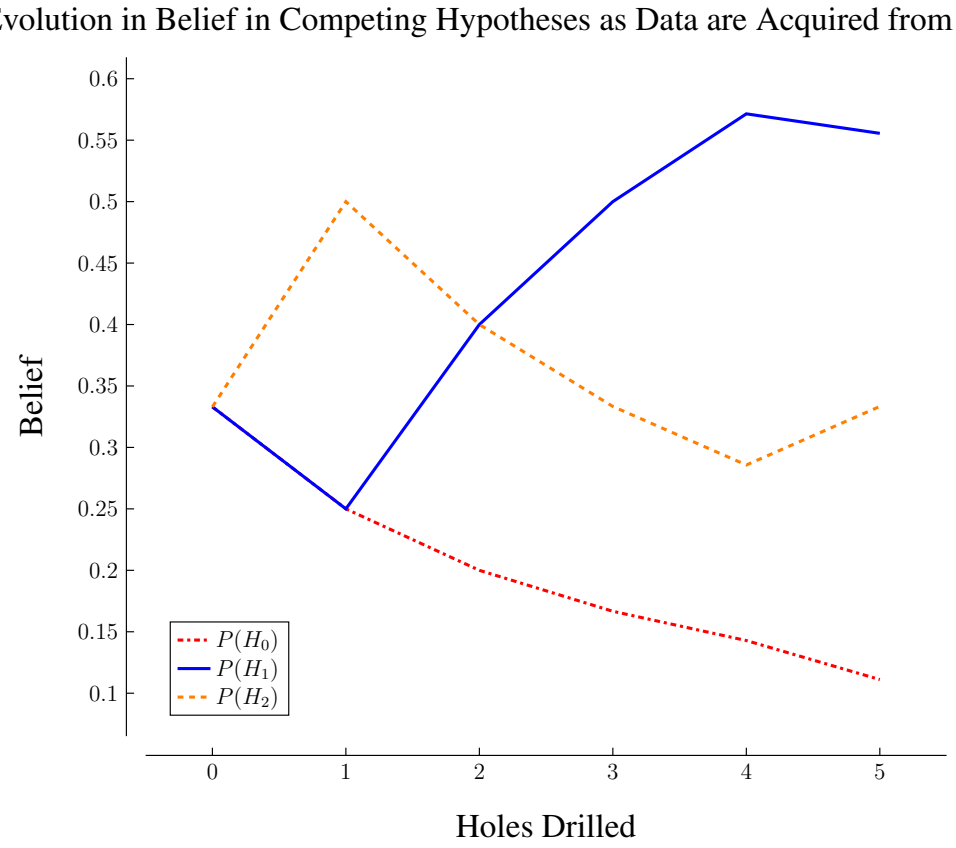


Figure 5.8: This plot shows the belief in the different hypotheses changing over time. Ultimately the non-uniform hypothesis was found to be most credible. This algorithm sent the robot first to points of disagreement between the two hypotheses.

5.5 Discussion

In this chapter we considered a symmetric noise model, with a maximum probability of an error of 0.5. Asymmetric noise models would also be worth investigating, but remain for future work.

Noisy observations warrant taking multiple readings at sites to reduce the effect of sensor noise. A principled strategy for trading off the reduction in uncertainty and the mission costs would be an important addition to a deployed system. Fig. 5.9 shows how many samples would have to be taken to reduce the sensor noise below 10%, assuming sensor errors are drawn independently. While multiple observations can reduce the noise level, they also reduce the overall mission budget.

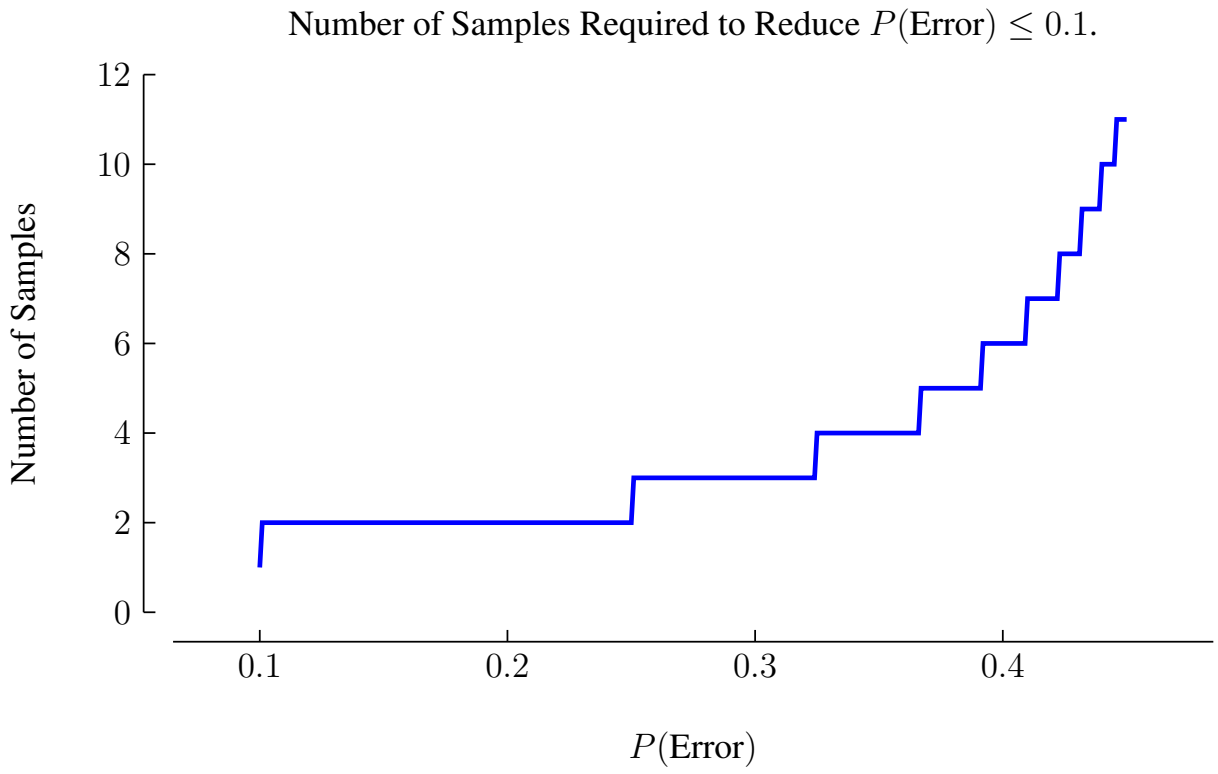


Figure 5.9: If one has a sensor that has a binary output, with an error specified along the x-axis, the y-axis gives the number of samples necessary to reduce $P(\text{Error}) \leq 0.1$. As $P(\text{Error}) \rightarrow 0.5$, the number of samples required to reduce the error level below 0.1 tends towards infinity.

As we have seen from the belief distributions learned in Appendix E, sensor error rates above 20% reduce the ability of the algorithms to successfully identify the best of the set of hypotheses. While the above graph illustrates how the effect of sensor noise can be reduced with multiple observations, it does assume that those observations are independently and identically distributed.

However, if the noise model is highly dependent on what is being observed, as say an image or spectral classifier might be, then it may prove difficult to ameliorate the effect of noise. Prior work, discussed in Section 5.1, assumes that the oracle providing labels for observations are noise-free, something that will almost certainly not be true during field operations.

Another consideration is that our robots had sampling budgets of at least 25 samples. In many planetary missions, it is not unusual to have a small number of samples. The Phoenix Mars Lander, for example, can only process four samples with the Wet Chemistry Laboratory instrument (Arvidson et al., 2009). Requiring multiple observations can drastically deplete small

sampling budgets, making it all the more important to ensure that sensor noise is small.

If one were attempting to falsify hypotheses with only a small number of samples we might propose a different strategy from the one presented above. In that case, it might be more valuable to focus exclusively on the points of disagreement and discount the spatial information gain term. Focusing on the disagreements in the hypotheses is predicated on the assumption that we would not send robots on missions without reasonable hypotheses.

We do not consider hypotheses that make probabilistic predictions. This decision is consistent with the author's experience with planetary scientists, but it is unsatisfying from a Bayesian standpoint. Incorporating such hypotheses is not challenging given the framework presented in this chapter and it should be considered in future work.

Similarly, the range of the predictions made by the hypotheses was quite narrow, it being the set $\{0, 1\}$. Hypotheses which make a broader range of predictions may yield more profound differentiation in relative belief.

The exhaustive evaluation of the domain of the hypotheses when attempting to identify the best sampling location does not scale well with the size of the domain. Scaling would also be a problem if the extent of the domain remains the same, but the level of discretization is increased. A sampling strategy such as a probabilistic roadmap, could mitigate the increased computational demand at the expense of some accuracy.

There are at least three obvious next steps that should be taken to continue the exploration of falsification-based sampling. First, do falsification planning that is more aligned to ground vehicle navigation. Second, take techniques from computational learning theory to control when hypotheses are rejected and new ones are sought. Third, integrate hypothesis generation algorithms.

The greedy algorithm used to select the waypoints is not necessarily ideal. It assumes that the vehicle has random access to any point in the hypotheses' domain. This assumption holds in the narrow context of the hypothesized mission discussed in the chapter, and indeed for experiment design where moving between experiments doesn't have a traverse cost, such as in drug design. But in the case of ground vehicles this assumption does not hold.

Paths for ground vehicles that ignore all the intervening locations between two goals are missing out on potentially valuable sampling opportunities. However, estimating the value of different trajectories can become computationally expensive. For that reason it would be natural to use an approximate planner like the one used by Arora et al. (2018), and faster information gain measures like the Cauchy-Schwartz quadratic mutual information used by Tabib et al. (2016).

The second improvement would be to use techniques from computational learning theory to evaluate the performance of the different hypotheses. Specifically, using concepts like Probably Approximately Correct bounds to estimate whether or not a hypotheses can be found to be accurate within a certain error threshold given the number of samples that are available.

What would make the PAC bounds even more useful is to develop an automatic means to estimate the number of samples required to estimate the effectiveness of a hypothesis given its symbolic representation and the anticipated noise models of the sensors. Much like automatic differentiation has been a game changer for artificial neural networks, such an automatic learning bound estimate would be instrumental in automated experiment design.

There is a problem with using PAC bounds, however. They rely on arbitrarily set thresholds for how accurate a hypothesis should be, and with what level of confidence we are prepared

to accept that accuracy. These thresholds must come from somewhere and could change how aggressively an algorithm would search the space of possible hypotheses. At this juncture the author can not think of an objective means to control these parameters.

Finally, the coupling of the falsification sampling algorithm with hypothesis generation algorithms is a vital part of liberating the robot from human oversight. The main objective of including the H_0 hypothesis in the algorithm was to identify when new hypotheses are needed. Using this belief state we could trigger the search for better hypotheses given the data collected so far, and add them to the set to be evaluated.

Having a robot simultaneously generate and falsify hypotheses while exploring the world is going to require substantially larger sampling budgets than are typically deployed in space missions. It is the necessary step to close the cycle of science in a deployed, fielded robot, a larger objective than just flight missions.

The approach developed in this chapter reduces the need to have a reliable communications link between the robot and remote human scientists. Since the algorithm is able to identify when the presented hypotheses are not credible, combining this algorithm with some of the hypotheses generation algorithms discussed in Chapter 2 would be a good step towards reducing robots' dependence on humans. A robot with the capacity to generate and evaluate hypotheses, and determine when to ask for new hypotheses, could conceivably conduct long-term scientific exploration without direct human supervision. Simultaneous hypothesis generation and falsification represents an exciting new direction for science autonomy research.

5.6 Summary

In this chapter we set out to develop an algorithm that sampled the domain of a set of hypotheses in order to determine which hypothesis was the most accurate. We achieved this by using mutual information sampling in the belief space over the hypotheses, in addition to mutual information sampling in the domain of the hypotheses. We also added a place-holder component in our algorithm which was able to identify when none of the hypotheses were of good quality.

We found that the falsification algorithm was better able to identify when none of the hypotheses were valuable, with statistical significance, than the standard mutual information sampling algorithm. Likewise, when all of the hypotheses were good, the falsification algorithm had statistically significantly more belief in the best hypothesis, although the magnitude of the increase was small. However, when the quality of the hypotheses was mixed, the standard mutual information sampling algorithm generally had superior performance.

There are three things to learn from these experiments. First, the addition of H_0 – the belief that none of the proposed hypotheses are correct – let the algorithms identify when none of the hypotheses were correct. Further, it did not distract the algorithms when hypotheses of mixed quality were present. We can conclude that H_0 provides value when attempting to explore the relative value of candidate hypotheses. Regardless of the value of the falsification relative to the mutual information sampling approaches, the H_0 term is a valuable addition to automated hypothesis testing.

Second, we learned that when all the hypotheses are of poor quality the falsification algorithm more readily learns that H_0 is the correct hypothesis to believe. While the more standard

approach to mutual information sampling does outperform the falsification algorithm when the quality of the hypotheses are mixed, both algorithms are capable of successfully identifying the correct hypotheses.

Thirdly, we learned that once the probability of getting an incorrect value from the sensor exceeds 20%, both algorithms begin to lose the ability to distinguish the quality of the hypotheses. At 50% error rate, the algorithms are incapable of distinguishing any of the hypotheses.

We also see that with sufficient sampling budget the performance of the algorithms converge. This makes sense, as the number of samples begins to cover the domain of the hypotheses, the accumulated evidence must support the same conclusion in both cases. However, if one can afford to exhaustively sample the domain of the hypotheses, then the problem addressed by these kinds of active learning vanishes.

Because we use a sampling approach to evaluating different parts of the hypotheses' domain, we do not need to understand the structure of the hypotheses in order to select informative sampling points. Since the functional form of the hypotheses is not necessary to evaluate their accuracy the presented algorithm can, and does, handle hypotheses which are more complex than simple linear classifiers.

Given that a long-term objective for science autonomy is to have robots conducting science without human supervision, ability to identify that the robot does not have a correct hypothesis is vital. Robots need to be able to identify that a new hypothesis is needed in order to decide when to ask for, or generate, a new one. The H_0 term can be used to quantify whether a new hypothesis is needed or not.

The work presented in this chapter can be a valuable tool for remote science missions. First, as a planner, it can be used to identify targets of high science value. Giving robots the freedom to plan actions that test hypotheses with reduced human interaction could increase mission science throughput. This also lets scientists communicate to robots in the language of hypothesis and mission science objectives, instead of in individual commands or actions.

Second, it can be used to reduce the need to develop a consensus on one hypothesis to be tested prior to a mission. Instead, a small pool of candidate hypotheses can be identified, and investigative actions can be planned, or suggested to a mission science team, that would resolve the relative accuracy of the proposed hypotheses. In this way the algorithm could act as an assistive tool in mission planning and resolve conflicts in the prioritization of sensing actions.

Thirdly, in the case of fully autonomous missions, where communications with a remote science team is not possible, the robot can use this algorithm to operate in a way that respects the priorities of remote science teams. Identifying that none of the proposed hypotheses are accurate lets robots determine when they should request new hypotheses from Earth, or perhaps more excitingly, generate their own.

Chapter 6

Conclusion

In this thesis we set out to improve three aspects of autonomous exploration by accounting for operational and environmental context. Each of the three algorithms – foraging, prospecting, and falsification sampling – represent components of a robot conducting science autonomously. We have shown, with statistical significance and at least moderate effect size, in all cases that improvements in performance were achieved.

We were able to improve the performance of foraging for information by recognizing that exploring agents do not have random access to all objects they would like to sample. We further demonstrated that distributions learned by foraging can be improved by detecting changes in the distributions underlying classes of objects being sampled. We improve the performance of the prospecting algorithm by building confidence in detected changes in the scalar field observed by the proxy sensor. Finally, we determined that falsification based sampling can more effectively identify which of a set of hypotheses most credibly explains the data. Below we lay out the specific contributions of this work, discuss limitations of the work, and then review possible avenues of future work.

6.1 Contributions

Our approach touched on three different aspects of autonomous field science. We demonstrated a causal link between the proposed algorithms and improvements over the credible baseline algorithms in all three approaches. The first two algorithms deal with situations where the robot is operating without global knowledge, the third algorithm plans experiments to determine which of a set of hypotheses are the most credible.

Foraging We considered opportunistic science when sampling discrete objects which are randomly distributed about an environment. Previous approaches did not consider the distribution governing which classes of objects an exploring robot would encounter. We have shown that for a variety of sampling and exploration costs our algorithm produces a statistically significant reduction in error in the estimation of the underlying distribution.

We also found that, given sufficient sampling budget, there were different settings of sampling and exploration costs which resulted in the foraging, greedy, and uniform sampling having the

best performance. When the exploration and search costs are low, the foraging algorithm is the best algorithm with statistical significance. To the best of the author's knowledge this is the first application of foraging algorithms to opportunistic sampling.

Further, we have identified that there are scenarios where the greedy and uniform sampling algorithms can have a statistically significant improvement over the other tested algorithms. By knowing the performance of the algorithms given exploration costs, sampling costs, and an operational budget, a robot scientist could modify its behaviour to produce better results.

Conversely, if one knows the exploration costs, then one could design sampling operations to control the costs of sampling, and keep robot performance within a desired regime. However, a formal understanding of these relationships remains to be determined.

We have also demonstrated that, with enough observations, a robot sampling discrete objects can successfully detect changes in the underlying distribution. This algorithm permits the robot to monitor changes while learning about the different classes of objects. This permits the robot to reset its probability distribution estimates and to segment a map based on their observations, which may demarcate changes in the environment. Regardless of the opportunistic sampling algorithm that is used by a robot, change detection would make a useful addition to any opportunistic sampling algorithm.

Prospecting The shortcoming that we identified in prospecting algorithms as they are practiced was the reliance on hard thresholds. In our work we demonstrated that we can estimate the confidence that the distribution underlying the readings has changed. The state of the art in this area is to deploy more expensive actions when readings have crossed a threshold. Our approach recognizes what could be sub-threshold behaviour, and need not be tuned to every environment.

When applied to the task of localizing maxima in a scalar field, and compared to the threshold-based algorithm we were able to better localize the maxima in the field with statistical significance and large effect size. While we did not capture as many maxima in the field as a surprise-based algorithm, change-detection algorithm made more effective use of the AIMs we deployed, with statistical significance and large effect size.

Our algorithm did not need to be adjusted for different deployments, unlike threshold based algorithms. We demonstrate on real data that our algorithm was able to detect sub-threshold changes in the distribution and act accordingly. This work represents a new, simple, algorithm that is agnostic of the trajectory the robot is following.

Falsification The third contribution of this thesis is an action planner that aims to determine which of a fixed set of hypotheses is most accurate. The algorithm selects actions that inform not only the belief distribution over the hypotheses, but also the performance of the hypotheses themselves. We compared the algorithm against mutual information sampling and found a statistically significant improvement for certain sensor error rates and sampling budgets.

Our algorithm does not need to understand the functional form of the hypotheses being investigated. It does require that the hypotheses can be evaluated at arbitrary points in their domain. Because we used a sampling approach to evaluating exploratory actions the robot only needs to know the predictions of the hypotheses and not their structure or functional form.

We found that the proposed algorithm was better able to identify that none of the hypotheses

were correct when all the hypotheses had less than 50% accuracy. This implies that the algorithm is better able to determine that a new hypothesis is needed when none of the proposed hypotheses are suitable.

We demonstrated this algorithm on the KRex2 robot in the Atacama Desert, and found the behaviour consistent with the simulation results. The samples collected by the algorithm supported the hypothesis favoured by the planetary scientists in the field.

This work demonstrates that robotic scientific explorers can plan actions that inform the belief in different hypotheses, and do so without reliance on deep knowledge of the hypotheses themselves. This, in turn, sets a new approach to science autonomy which is not focused exclusively on collecting data but on letting robots explore on behalf of scientists, collecting data relevant to the scientists' hypotheses. Further, the collected data could be used to support hypothesis generation techniques like those described in Section 2.4.

Collecting data in the context of hypotheses returns to the notion of science as a cyclic process. We have demonstrated how robots can use multiple competing hypotheses to determine what new experiments ought to be conducted, but what remains is the question of when and how to generate new hypotheses. This work is the first to direct robot actions to falsify multiple competing hypotheses. Selecting actions that simultaneously falsify multiple hypotheses and being able to identify when none of the hypotheses are suitable is a necessary component for a robot that is simultaneously generating and testing hypotheses.

6.2 Limitations

As stated previously, empirical work can naturally only speak to the circumstances under which the data were collected. While our experiments demonstrate statistically significant improvements in performance for the proposed algorithm, hypotheses about performance outside of the stated conditions must be rigorously supported with experimentation.

Foraging The foraging algorithm makes the assumption that the next encountered object is drawn from a distribution independent of previously encountered objects. Modelling the next object to arrive with a Markov chain would further improve the fidelity of the algorithm to field operations, and should gracefully handle when the classes of encountered objects are independently and identically distributed.

The foraging system as described in this document does not consider the mission objectives beyond the cost for sampling and exploring, and even then it employs a greedy approach. It would be beneficial to integrate the foraging algorithm with a higher-level scheduler like CASPER (Knight et al., 2001), in order to prioritize opportunistic sampling in the context of mission objectives.

Prospecting The prospecting algorithm that we presented in this document intentionally ignored the fact that the one dimensional signal the algorithm was monitoring was embedded in a two dimensional environment. There are almost certainly advantages to be gained by recognizing that the robot is operating in a more complex environment. Like foraging, requests to deploy

an AIM could be scheduled by a higher-level system that could account for the spacing between AIMs, and the degree of overlap from previous AIM deployments.

Falsification Our approach to falsification only considered a small number of hypotheses, testing with a larger pool of hypotheses is warranted. While adding more hypotheses should not reduce the ability to identify which of the hypotheses are most likely to be correct, it could have deleterious effects on the ability to recognize that none of the hypotheses are accurate.

Further, this algorithm does not update hypotheses in response to the data that have been collected. Being able to revise hypotheses in the light of new evidence would increase the ability of the robot to learn the correct hypothesis. However, for design purposes it might be best to keep the knowledge of how to update the hypotheses separate from the action planning algorithm.

Samples were selected such that they maximized the sum of the information gained in the distribution over the samples collected from the hypotheses' domain and the information gained in the belief distribution over the hypotheses. This representation makes naive assumptions about the relationship between the hypotheses. Further, it is possible that some samples may appear valuable by having moderate information gain in both components of the sum, but do not yield considerable information in either the domain of the hypotheses or the belief in the hypotheses alone. It is possible that other weightings of the components are required, or that the value of samples should be derived from a more comprehensive formalism.

There is lacking a measure of salience of the results. In the algorithm as it is designed no one observation can disqualify a hypothesis. In fact, if a considerable amount of favourable data has been collected, it could prove challenging to unseat a hypothesis. Partially this is a function of the codomain of the hypotheses being relatively small, but the algorithm should be modified to account for this.

One of the stated objectives of the algorithm was to identify when a new hypothesis is needed, by maintaining a belief in a hypothetical hypothesis, H_0. It might be that the variance across the belief in the hypotheses is a better indicator that a new hypothesis is needed. Regardless, it remains a problem to determine when to trigger a search for a new hypothesis and when to sample amongst the existing hypotheses.

6.3 Future Work

While the work in this thesis advances science autonomy algorithms, there still remains work to do. To that end we identify a number of extensions for the algorithms presented in this document, and to science autonomy in general, that should be undertaken to improve the state of the art.

In the presented work we used a measure of information gain based on Shannon's definition of entropy. As discussed in Chapter 2, there are advantages to using the Cauchy-Schwartz Quadratic Mutual Information (Principe, 2010) in information gathering algorithms, chiefly the increased speed of computation. The efficiency of the algorithms presented in this thesis could be improved by using the Cauchy-Schwartz Quadratic Mutual Information criterion.

Foraging The algorithm should be modified to recognize that the arrival distribution of the different classes of objects is not memoryless. This would make the algorithm more amenable

to operations in environments where the next sampling opportunity is highly dependent on the currently available option.

The authors would like to investigate different cost/benefit analyses for the exploration vs sampling decision. As noted previously, using the ratio of reward to cost is potentially numerically unstable, therefore different strategies should be investigated.

The algorithm tested in this document only considered objects where the underlying distribution was Bernoulli. Other distributions should be tested in order to be more reflective of the variety of observations – real valued, vector valued – that can be collected by robots in the field.

The algorithm should be integrated with a mission scheduler and tested on a real robot. During deployment it would also be worthwhile to integrate the algorithm with some form of classifier that identifies targets of interest in the environment. Consequently the algorithm will need to be updated to account for uncertainty in the classification of objects.

Prospecting The prospecting algorithm should be extended with a planner that makes more rigorous decisions about resource trade-offs and mission risks incurred in more involved sampling processes like drilling. The addition of confidence in change detection could let robots apply traditional decision-making rules without human supervision.

Multi-hypotheses testing would let a robot scientist deal with a broader range of scenarios while exploring. Multi-hypothesis testing would also let the algorithm consider more than one number of change points in the underlying distribution at any given time.

The AIMs that were deployed by the prospecting algorithm were simple Archimedean spirals. It would be useful to deploy AIMs that were more informed by data previously collected. This would be an excellent application of an algorithm like that deployed by Wilson and Williams (2017). A more sophisticated plan for AIM deployment could account for higher dimensional environment that the presented change detection algorithm ignored.

Falsification The falsification planning algorithm needs to incorporate uncertainty in the hypotheses' predictions. However, this is a straightforward application of Bayes' chain rule. But perhaps the most important question that needs to be addressed as the next step is when is it necessary to create a new hypothesis. If we use the quantity $P(H_0)$ to determine if a new hypothesis is required it is possible that early observations could cause an excessive amount of time spent in generating new hypotheses before a sufficiently informative dataset is collected. As discussed in Chapter 5, the automatic determination of the number of samples required, under a PAC learning framework, to be confident in the predictions of a hypothesis would be immensely valuable. This would be an important step forward in simultaneous exploration and hypothesis generation.

Science Autonomy, in general The perception algorithms that are used in science autonomy focus around fairly basic classifiers, and most of them are pre-trained before they are deployed. Having a perception system that learns *in situ* could help distinguish a wider range of scientifically interesting phenomena. DEMUD, developed by Wagstaff et al. (2013), is an important first step in this approach. Given the progress being made in artificial neural networks, they would be a natural avenue to explore. However, the training of neural networks relies on large volumes of

training data and powerful computers. The computational demands are antithetical to the computing resources available on flight missions. On the one hand, one could find a middle ground between basic, pre-trained classifiers and full deep learning system, such as extreme learning machines (Huang et al., 2011). On the other hand, one could explore alternative computing architectures.

Non-traditional computer architectures, as discussed in (Younger et al., 2014; Traversa et al., 2015), are an area that should be investigated for building robot scientists. Traversa et al. (2015) claim their computing hardware is able to solve non-polynomial problems in polynomial time. Given that solving information gathering problems can be computationally intensive, it is worth evaluating whether these new kinds of computers could improve the performance of robot scientists and what, if any, effect their use would have on scientific decision-making processes.

Another interesting advancement is the development of language for individual robots exploring both here on Earth and in the universe. This language must be grounded in sensory apparatus and also be temporally aware. A lot of information is compressed in language, and humans use metaphors and analogies to do reasoning. Imbuing a robot scientist with some of that ability could help speed up hypothesis formation. This is again something that can be addressed with non-traditional computing (Kanerva et al., 2000).

Appendices

Appendix A

When is autonomy appropriate?

The work in this document is to improve the autonomy of on-board vehicles, by increasing the adaptivity of the on-board autonomy. Autonomy can take on different roles in a mission, and include assistive tools to help humans make better decisions, faster.

How autonomy should be employed in a mission depends on how the algorithms can help achieve mission objectives. This section covers some of the considerations that go into making the decision about when and where autonomy should be used in a mission.

A.1 Factors In Selecting Autonomy Algorithms

When evaluating different autonomous systems we should consider a number of factors in designing the system. Factors that affect the ability of a system to complete tasks in a timely fashion include the mean time between interventions from remote operators, the mean time for those operators to complete interventions, and how tolerant the system is of neglect by remote operators. These factors are well analyzed by Shah et al. (2008).

The performance of the algorithms in the system can also be analyzed in terms of the time to complete tasks and the accuracy in completing those tasks. The algorithms' performance should also be assessed with respect to the mission's tolerance for commission of errors and the omission of desired actions.

There are also factors extrinsic to the algorithms being considered. These are constraints placed on the mission independent of the algorithms themselves.

The operations tempo of a mission is affected by, among other things, the time to complete the mission, how long the robot can survive in the planned environmental conditions, and the dynamics of the phenomena being observed. The mission tempo reflects the time pressures on the mission.

Operational tempo is not necessarily fixed over the lifetime of a mission. If the primary mission objectives are accomplished, and there is still time remaining for operations, then the tempo can change from high to low. Likewise, unanticipated events that shorten the mission lifetime can change the operational tempo from low to high.

The system completing the mission, including autonomous and human components, must keep pace with the mission tempo. The decision-making loop of the planned system must execute

fast enough to keep up with the operational tempo, regardless of what autonomous components are employed.

Human operators in the loop are subject to workload stresses, which can degrade their performance. Autonomy can alleviate the workload placed on human decision makers. Assistive tools in the mission operations centre is a valuable use of autonomy.

The communications throughput reflects the availability, reliability, and capacity of the communications link between the robot and the remote science team. The communications link constrains how quickly decisions can be made and, depending on where decisions are being made (on-board or remotely), can constrain the performance of decision making loops.

A major cost in the development of flight missions is the burden of proving that a given system performs the desired tasks (validation) and that they are performed correctly (verification). The cost of validation and verification for a system increases with its algorithmic complexity.

The stringency of the validation and verification process is a function of the mission risk tolerance, and mission budget. Successful validation and verification of an autonomous system depends on the ability to trust and predict the behaviour of complex systems in unknown environments.

Highly risk tolerant missions, such as CubeSats (Heidt et al., 2000), can afford much more complex or experimental, and hence risky, autonomous systems, than could a mission where human life could be endangered. Building a level of trust appropriate to mission risk posture involves formal verification of the system, empirical testing of behaviour, and building a legacy of performance and, in the case of space missions, flight heritage.

However, time and cost constraints can limit the thoroughness of the validation and verification process. In these cases risks which could be controlled through autonomy may have to be controlled through operational procedures.

A.2 One Analysis of the Design Space

Choosing what degree of autonomy to use is a decision with a continuum of factors to consider and design decisions to make – how much autonomy, how complex. Modelling this decision process is outside the scope of this thesis. However, in Table A.1 we present one possible view into how this decision might be made, considering only two factors in the mission. We consider the communications throughput and the mission operations tempo.

		Communications Throughput	
		High	**Low**
Operations Tempo	**Low**	Simple Autonomy Human Adaptivity	On-board, Complex Autonomy and Adaptivity
	High	Distributed Autonomy	On-board, Simple Autonomy and Adaptivity

Table A.1: This table gives one possible decomposition of an autonomous mission, discretizing communications throughput and operations tempo into "high" and "low". For each combination we suggest one possible approach to designing an autonomous system.

Operations tempo and communication throughput is discretized into "high" and "low" values, and we only consider the space defined by these two factors. Recognizing that these distinctions are qualitative and relative, we discuss different kinds of autonomy that may be appropriate for these scenarios.

Communications links from Earth to the Moon of Earth and to Mars have very high communications throughput compared to a surface mission to Europa or Enceladus. Surface missions to Europa would have higher communications throughput again than a mission to explore the subsurface ocean of said moon.

The early Vega missions (Surkov et al., 1986) had a very high time pressure, due to the nature of the Venusian atmosphere. They could be said to have a high operations tempo, due to the short mission duration. Likewise, if the phenomena being observed is ephemeral, such as water plumes erupting on Europa, the pressure to make decisions quickly, and execute well-timed actions is much higher.

In contrast, the Mars rovers have the benefit of a relatively benign environment, as well as a superbly engineered robotic platform, well exceeding the original nine-month planned mission lifetime. Consequently the tempo of these missions is less demanding. The ground vehicle missions to Mars could be said to have a low tempo.

High Communications Throughput, Low Operations Tempo: When the communications throughput is high it affords remote science teams the opportunity to be actively involved in the decision-making process. In this situation we would recommend using simple on-board autonomy, with the remote science team providing the adaptivity of the overall system.

When the operations tempo is low, the need to make decisions quickly is low, so humans can afford to take time to evaluate decisions and modify parameters controlling algorithms on-board the vehicle. The need for sophisticated autonomy is reduced, as humans can provide that behaviour.

A low operations tempo gives the opportunity to field simple algorithms whose behaviour can be monitored and modified by human operators. The simplicity of the on-board algorithms reduces the burden of validation and verification on the mission while still producing a responsive

system capable of conducting valuable field science.

This mode of operations is used to great success with the Mars rovers (Sojourner, Spirit, Opportunity, and Curiosity). Communications are relatively reliable and the environmental conditions do not place a strong time-pressure on vehicle operations. By using a combination of simple on-board autonomy and extensive human adaptivity, enabled by the reliable communications link, the Mars rovers and remote science team as a system produce autonomous behaviour that is responsive to the environment and is capable of learning from it.

High Communications Throughput, High Operations Tempo: When the tempo of the mission is increased, but communications throughput is still high, the role of autonomy changes. Where in the previous setting the remote science team can afford to make considered decisions, a high tempo mission adds pressure to make decisions quickly on the remote science team. Here we would recommend a distributed autonomy system.

In a distributed autonomy system some of the software components are running on-board the remote vehicle while other components are running on Earth, where computing resources are greater. The components of a distributed autonomy system which are running on Earth can either be directly in the decision-making loop or act as a decision-making tool for humans.

Increasing workload pressures on human decision makers reduces the quality of the decisions made. As such, there is a need to produce tools which can help a remote science team preserve the quality of their decisions, and ameliorate the effects of workload stress and imperfect situational awareness.

An example of a mission with high mission tempo and high communications throughput would be the proposed Resource Prospector mission (Andrews et al., 2014). The Resource Prospector rover is not designed to survive lunar night, so there are only fourteen Earth days to complete the mission. Because lunar missions can have low latency, high availability communications, a mission can afford to have autonomy algorithms running on the Earth.

As part of a distributed autonomous system, humans provide valuable oversight of the algorithms. If the components of the autonomous system are loosely coupled, as they are likely to be when distributed between the robot and the Earth, then Earth-based components of the system can be used by the remote science team to assist in the decision making process. Once trust has been sufficiently established in these tools, one might imagine reducing human oversight and integrating them more closely in future decision-making pipelines.

Low Communications Throughput, Low Operations Tempo: When the communications throughput is low, the robot is isolated from the guidance of the remote science team. Fortunately, when the operations tempo is low, the vehicle can take more time to make decisions. This is an opportunity to employ more complex autonomy, which may take longer to execute, than might be used in a mission with high communications throughput or high mission tempo.

We recommend the use of on-board autonomy and adaptivity which is as complex as the mission's validation and verification process can support. This recommendation assumes that the proposed autonomy is no more complex than what is required by the mission goals. Missions with low operations tempo can afford the time it takes to execute complex algorithms on on-board computing resources.

For example, in an orbital mission around a distant planetary body, a robotic satellite has the luxury of completing multiple passes over regions of the surface during the mission lifetime. Provided the phenomena being observed persists longer than the time to re-observe the original location, the tempo of the mission is not high. The time between observations can be used to process collected data and regenerate plans to maximize the utility of future actions.

Low Communications Throughput, High Operations Tempo: Increasing the mission tempo increases the time pressure for the algorithms on-board the vehicle. When the mission tempo is high, making imperfect decisions quickly may be better than making good decisions slowly. Hence, here we would recommend using simpler algorithms than one might use when mission tempo is low.

Especially given the constraints of space-rated hardware, the mission might not be able to afford the time to execute state-of-the-art algorithms on-board the vehicle. Repeated use of slow decision making algorithms could exhaust the mission budget for time or power, or the phenomena to be observed may no longer exist by the time sampling decisions have been made.

An example of a mission with low communications throughput and high operations tempo would be the autonomous exploration of the subsurface oceans of Europa or Enceladus. Here a robot is likely to be almost completely cut off from a remote science team during the mission. The vehicle would have to make decisions without human oversight. Without access to solar power, the robot is likely to have a finite power budget, and all science must be completed within that constraint.

Likewise, the proposed lake lander mission to Titan, the Titan Mare Explorer (Stofan et al., 2013), proposes a vehicle with limited control capability. The planned vehicle would operate at the mercy of the currents (Lorenz et al., 2012), meaning that it won't be able to keep place, or track interesting targets.

While the Titan Mare Explorer can have direct to Earth communications, at an estimated distance of approximately $1.2 \times 10^9 km$, the light-time delay in communications could be on the order of an hour. On-board autonomy can help capture events that occur below what is observable given the communications delay. However, decisions to sample or collect ephemeral data must be made quickly, requiring fast-acting autonomous algorithms.

A.3 Assessing Complexity of Autonomy

Above we made reference to "complex" and "simple" autonomy. Next we discuss what it means for an algorithm to be simple or complex.

Without reliable human communications the vehicle will require some form of autonomous behaviours, even if it is something as simple as waypoint following, or the blind drive commands sometimes used with the Mars Exploration Rovers (Biesiadecki et al., 2007). Likewise, some degree of safeguarding autonomy is likely to be necessary on-board, as eloquently argued by Pratt and Murphy (2012).

Above and beyond a baseline level of autonomy, there are more sophisticated algorithms, like path planners for information gathering. Further, because mission tempo can change during missions, it is worthwhile giving special attention to anytime algorithms.

To discuss the complexity of autonomous systems we should consider the algorithmic complexity of the algorithms, and the constraints of the computing environment in which the algorithms are running. Algorithmic complexity has a huge impact on the validation and verification burden of the mission, as well as the ease with which trust can be built in the system. This is especially important when the robot is operating without human supervision in an alien environment.

As the system's algorithmic complexity increases, it can become increasingly difficult to ensure that the space of possible behaviours has been tested. The validation and verification process can grow exponentially with the complexity of the algorithms, adding considerable burden to the mission budget.

There are also the limitations imposed by the computing resources on-board an autonomous system. When complex algorithms are run on limited hardware, which space-rated computing often is, the execution time of the algorithm can be greatly reduced. This can be a problem when considering mission tempo.

Take for example the GESTALT algorithm on MER. In comparison to, for example, a self-driving car, GESTALT is not a complex autonomy system. However, because it is running on a RAD6000 processor, which has a clock speed on the order of 200MHz, it takes approximately 70 seconds to plan an autonomous traverse (Maimone et al., 2006).

The complexity of an algorithm cannot be understood out of context of the computing system it is being executed on. There are new computing architectures being developed for operations in radiation environments, but at the time of writing this thesis, readily available options are limited.

A.4 Summary

Deciding when and how to use autonomy must be tailored to individual missions, taking into account the costs and the benefits of any proposed system. This section briefly addresses many topics which warrant considerable discussion and analysis, in and of themselves.

Key to understanding the role of autonomy in flight missions is modelling the effect the selected algorithms have on the concept of operations, workload on operators, and the mission budget. Reducing the costs of the validation and verification process, and increasing the trust in autonomy are vital to advancing autonomy in science missions.

Appendix B

List of Terms

AIM Area of Interest Manoeuvre.

Entropy Assumed to be Shannon's Entropy, $H(p)$ of a distribution $p(x)$, defined to be $H(p) = -\sum_{x \in X} p(x) \log_2 p(x)$.

Hypothesis A function that predicts from an input space X to a probability distribution over an observation space, Z. Formally: $h : X \to (Z \to [0, 1])$

I.I.D. Identically and independently distributed.

NSS Neutron Spectrometer.

Object An entity in the environment the robot can reason about.

Opportunistic Sampling Sampling conducted in response to sampling opportunities that were not or could not be anticipated.

PAC Learning Probably Approximately Correct Learning, invented by Leslie Valiant. A branch of computational learning theory that specialises in putting confidence bounds on learned functions.

Perplexity A measure of how well a probability model describes an observation.

Primary or Proxy Sensor This is the instrument that is used to identify sampling opportunities. For example, a camera which can be used to determine the type of rock visible

Prospecting The navigation of a scalar or vector field in order to seek locations that maximize an objective function.

Prospecting Sensor The primary sensor used during prospecting.

Sampling Resources Material that is consumed during the sampling process. For example, chemical reagents or sample containers.

Sampling Opportunity An object that the robot can choose to sample with one of its instruments, for some cost.

Secondary Sensor This is the instrument that is used to extract observations from sampling opportunities identified by the primary sensor. Secondary sensors can be more expensive in time, energy, or sampling resources than the primary sensor.

Speed Made Good The speed of a vehicle towards the goal. Less than the speed of the vehicle

unless the vehicle is headed directly towards the goal.

Surprise As defined by Claude Shannon, surprise of an event x given a probability distribution p is $-\log p(x)$

Transect A path along which one collects data.

Appendix C

Opportunistic Sampling of Discrete Objects (Foraging) Supplemental Material

In this appendix we present the performance of the competing algorithms (Foraging, Uniform, and Greedy) against a sampling algorithm which always engages with the environment, called "Always Engage".

C.1 Experiment 1 - Uniform Arrival Distribution, Different Underlying Distributions

In the case where all objects are equally likely, the Foraging algorithm has the least bad performance relative to the always engage algorithm.

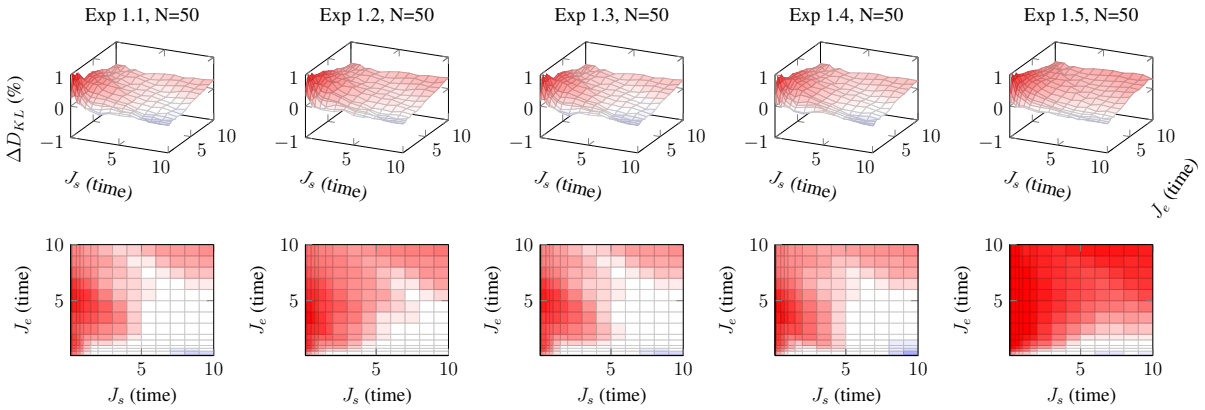


Figure C.1: Uniform sampling algorithm vs Always Engage sampling algorithm. Arrival distribution is uniform, $K = 3$ with different underlying distributions.

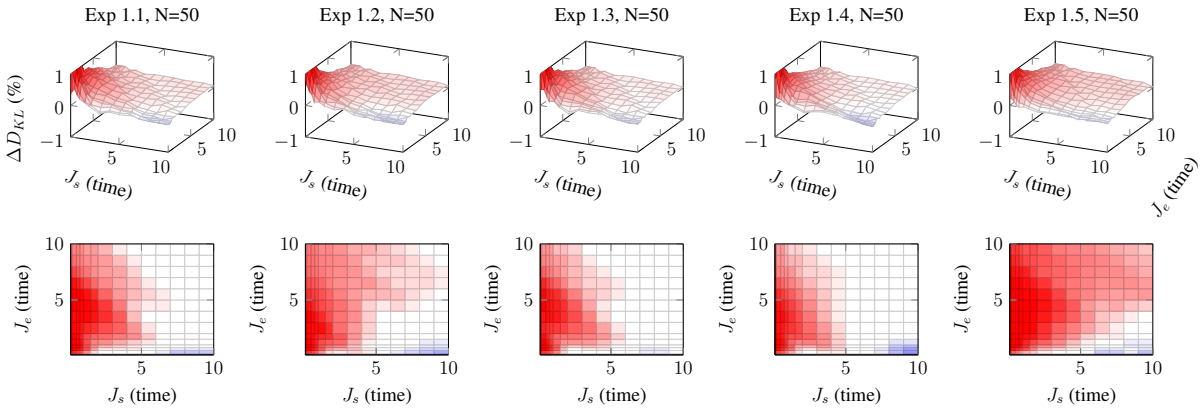


Figure C.2: Greedy sampling algorithm vs Always Engage sampling algorithm. Arrival distribution is uniform, $K = 3$ with different underlying distributions.

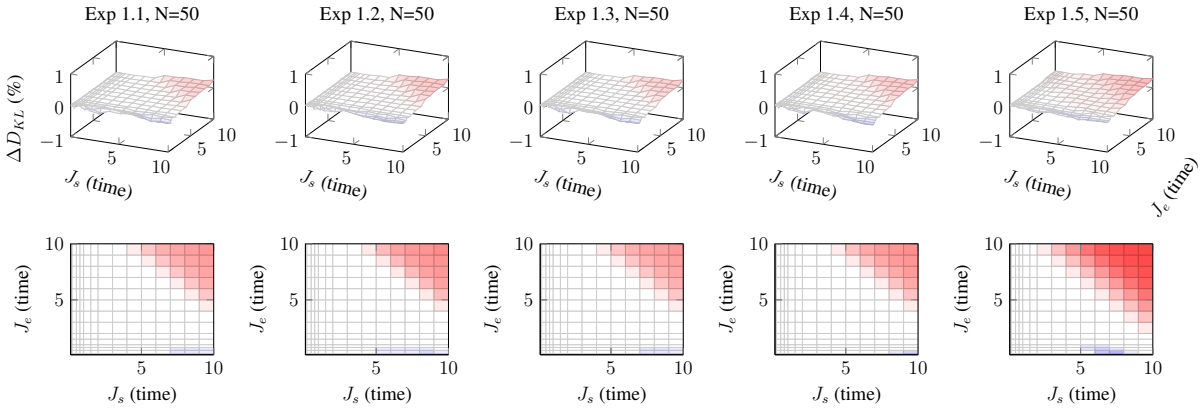


Figure C.3: Foraging algorithm vs Always Engage sampling algorithm. Arrival distribution is uniform, $K = 3$ with different underlying distributions.

Notice that the Foraging algorithm has the least amount of underperformance relative to Always Engage over the space of sampling and exploration costs.

C.2 Experiment 2 - Skewed Arrival Distribution with Identical Underlying Distributions

Here we give the performance of the competing algorithms with respect to the Always Engage algorithm with the unbalanced arrival distribution. These distributions test performance of the algorithms when $K = 6$ objects.

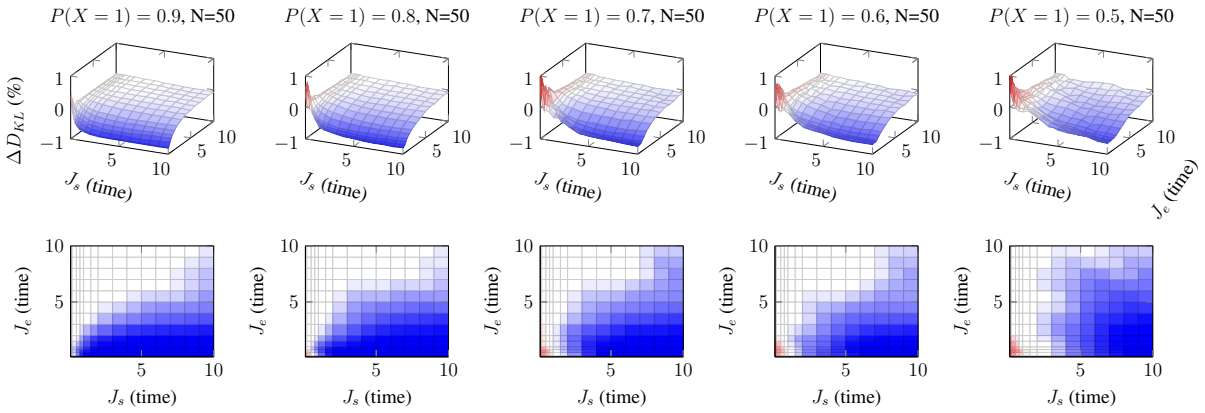


Figure C.4: Uniform sampling algorithm vs Always Engage sampling algorithm. Arrival distribution is unbalanced, with $P(X)$ being much larger than for other classes of objects. $P(Z|X = x) = 0.3 \, \forall \, x \in X$.

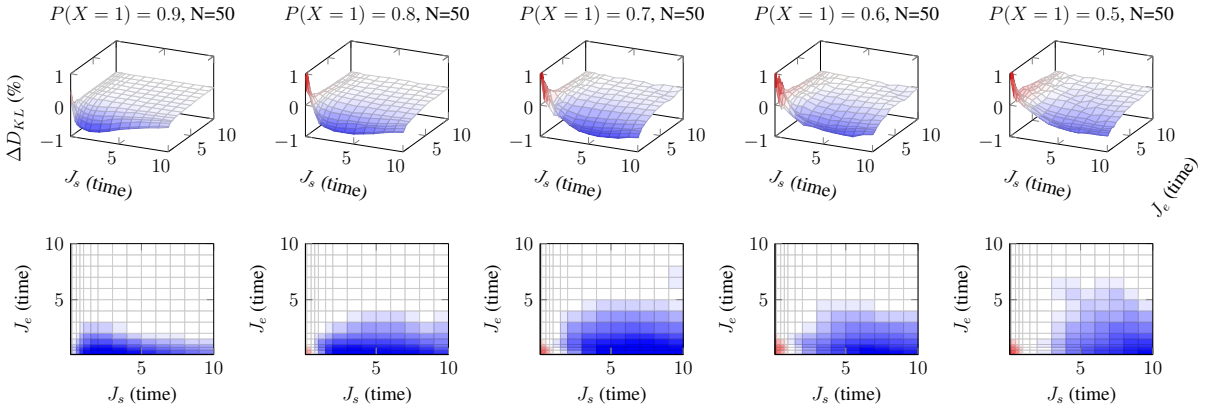


Figure C.5: Greedy sampling algorithm vs Always Engage sampling algorithm. Arrival distribution is unbalanced, with $P(X)$ being much larger than for other classes of objects. $P(Z|X = x) = 0.3 \, \forall \, x \in X$.

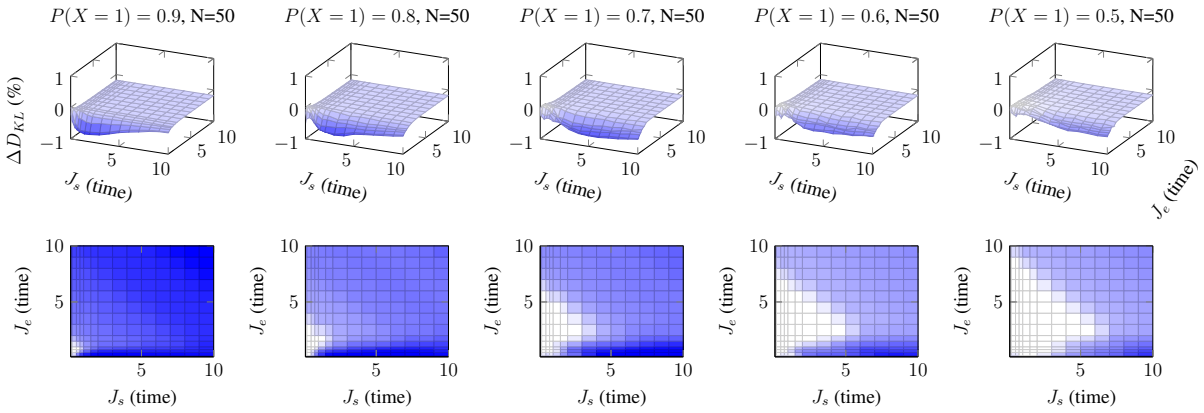


Figure C.6: Foraging algorithm vs Always Engage sampling algorithm. Arrival distribution is unbalanced, with $P(X)$ being much larger than for other classes of objects. $P(Z|X = x) = 0.3 \, \forall \, x \in X$.

Uniform (Figure C.4) and Greedy (Figure C.5) perform as well as or better than Always Engage for most settings of exploration and sampling costs. The Foraging algorithm never underperforms the Always Engage algorithm (Figure C.6).

Next we show the performance of the competing algorithms with respect to the Always Engage algorithm with a Zipfian arrival distribution, where $s = 1$ and $K \in \{5, 6, 7, 8\}$. Again, the Foraging algorithm never underperforms the Always Engage algorithm (Figure C.9).

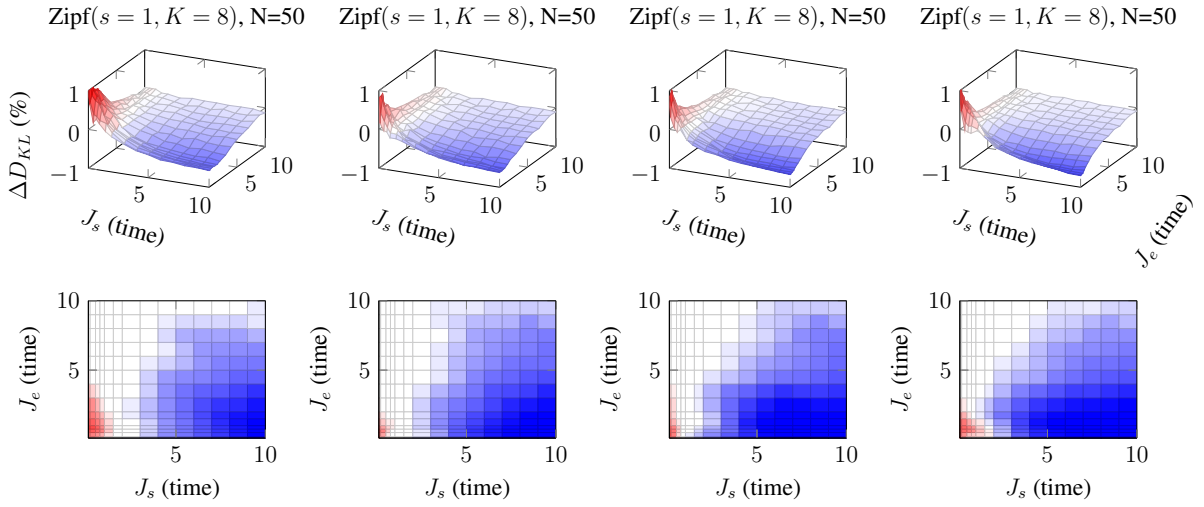


Figure C.7: Uniform sampling algorithm vs Always Engage sampling algorithm. Arrival distribution is Zipfian with $s = 1$ and $K \in \{5, 6, 7, 8\}$.

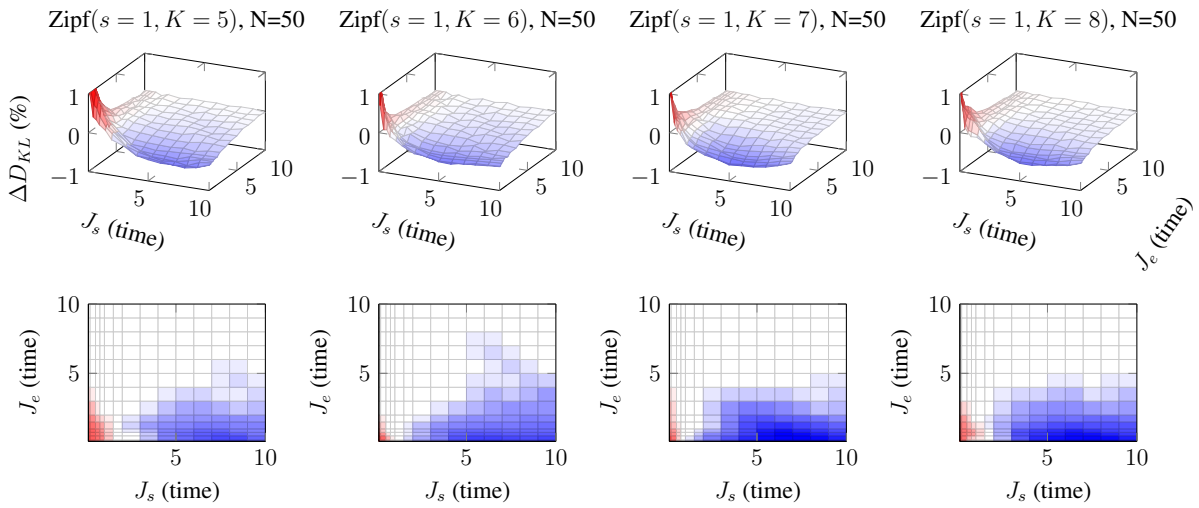


Figure C.8: Greedy sampling algorithm vs Always Engage sampling algorithm. Arrival distribution is Zipfian with $s = 1$ and $K \in \{5, 6, 7, 8\}$.

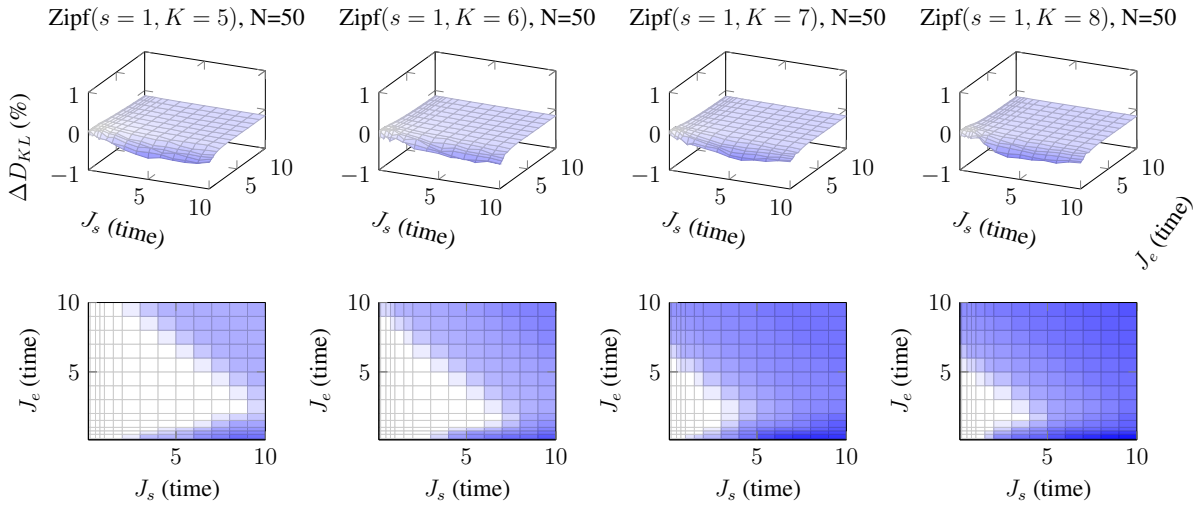


Figure C.9: Foraging algorithm vs Always Engage sampling algorithm. Arrival distribution is Zipfian with $s = 1$ and $K \in \{5, 6, 7, 8\}$.

C.3 Experiment 3 - Skewed Arrival Distribution with Distractor Object

These plots show the performance of the competing algorithms with respect to the Always Engage algorithm first with the unbalanced arrival distribution with $P(X) = 0.8$ and $K = 8$, and next with a Zipfian distribution with $s = 1, K = 8$.

Uniform's underperforming of the Always Engage algorithm is more pronounced when a Zipfian distribution is followed (Figure C.10). There appears to be no difference in performance

as a function of the underlying distribution of the modified object class. This also holds for the Greedy algorithm (Figure C.11). Again, the Foraging algorithm never underperforms the Always Engage algorithm (Figure C.12).

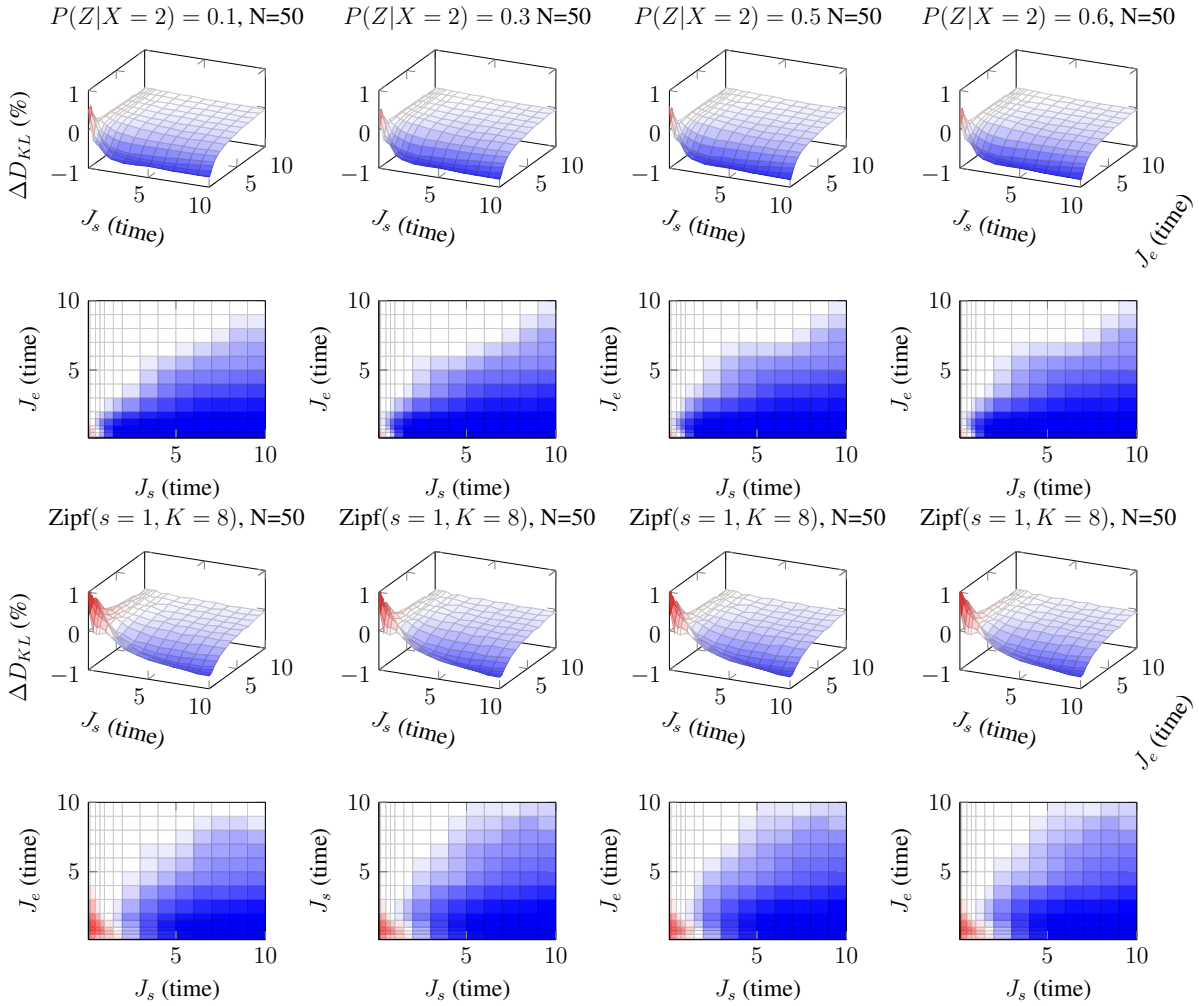


Figure C.10: Uniform sampling algorithm vs Always Engage sampling algorithm. Second most common object has a distractor underlying distribution. Both unbalanced (top) and Zipfian (bottom) arrival distributions are used.

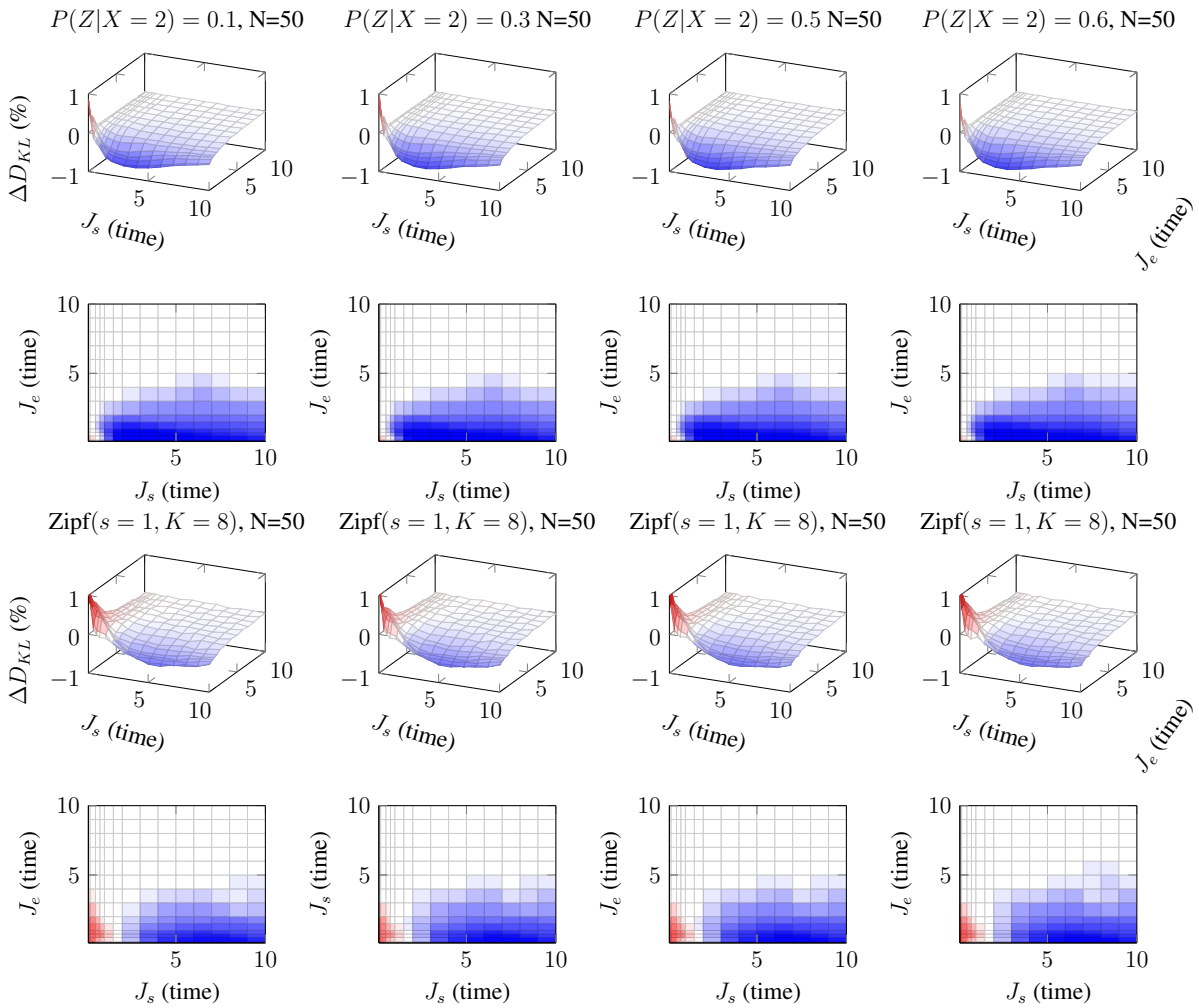


Figure C.11: Greedy sampling algorithm vs Always Engage sampling algorithm. Second most common object has a distractor underlying distribution. Both unbalanced (top) and Zipfian (bottom) arrival distributions are used.

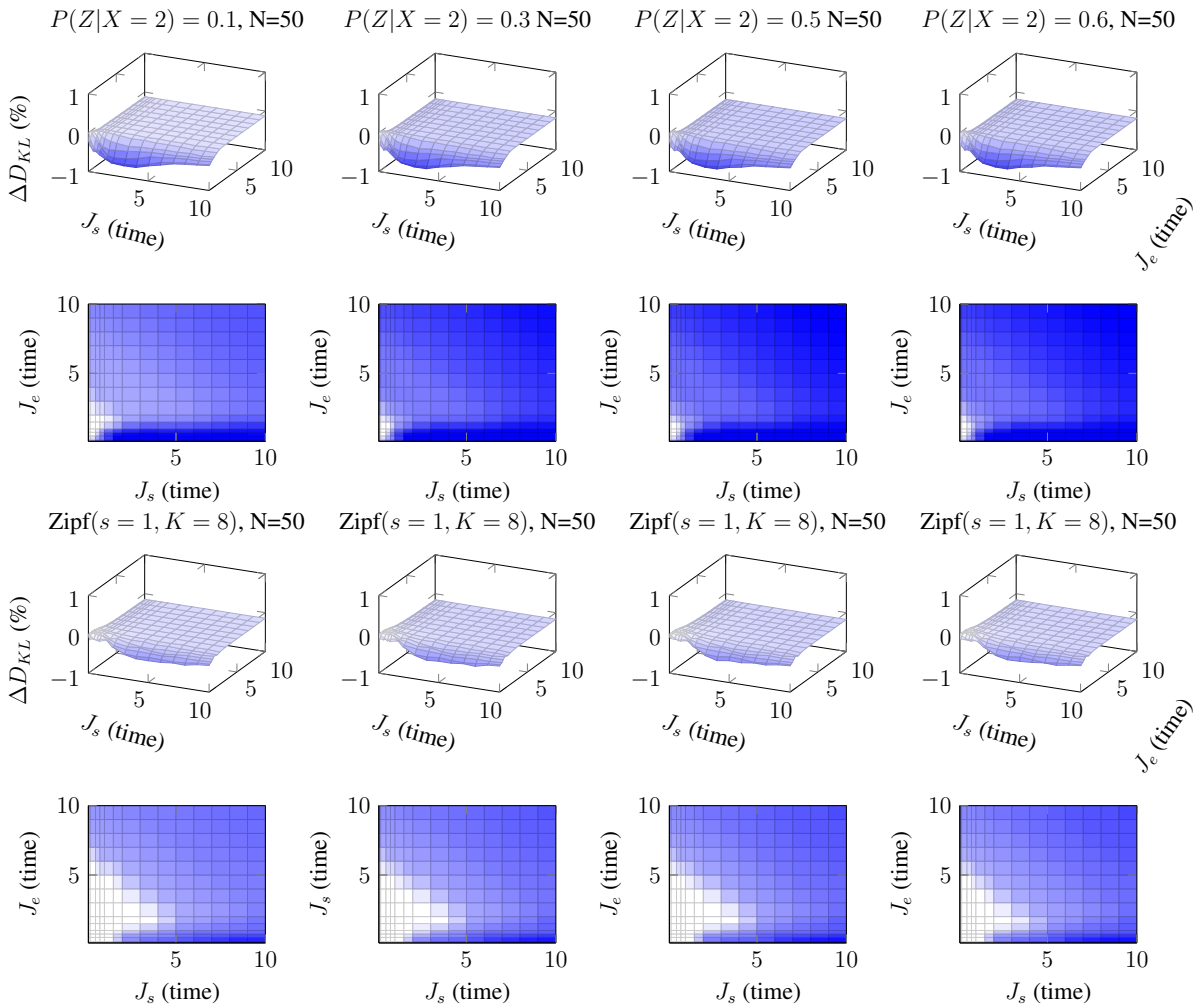


Figure C.12: Foraging sampling algorithm vs Always Engage sampling algorithm. Second most common object has a distractor underlying distribution. Both unbalanced (top) and Zipfian (bottom) arrival distributions are used.

C.4 Experiment 4 - Skewed Arrival Distribution with Random Underlying Distributions

These plots show the performance of the competing algorithms with respect to the Always Engage algorithm first with the unbalanced arrival distribution with $P(X) = 0.8$ and $K = 8$, and next with a Zipfian distribution with $s = 1, K = 8$.

Uniform sampling performs better than Always Engage for most sampling costs, with a degradation that is much more pronounced when the arrival distribution follows Zipf's law (Figure C.13. Notice that the Uniform sampling performs worse for RAND3 than for the other settings of the underlying distributions.

The Greedy algorithm (Figure C.14) appears fairly robust to the changes in the underlying distribution. Generally good performance for small exploration costs, excepting small sampling

cost, where the Always Engage algorithm performs better.

The Foraging algorithm also has performance that is sensitive to the settings of the underlying distributions, as can be seen in Figure C.15. Notice only in one setting, the RAND3 settings for the underlying distribution and the unbalanced arrival distribution, does the Foraging algorithm perform worse than the Always Engage algorithm. The worse performance is localized to large settings of exploration cost and sampling cost.

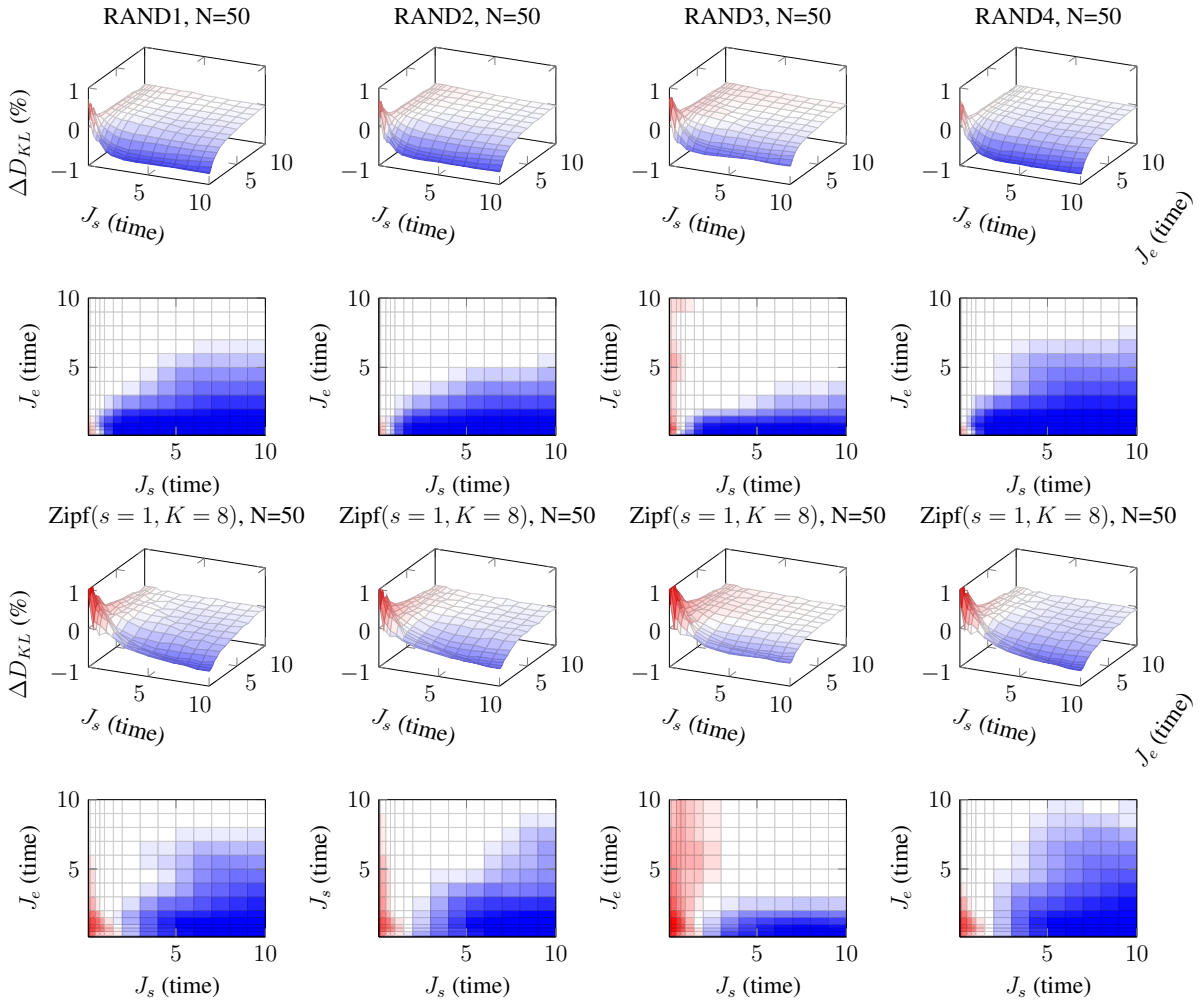


Figure C.13: Uniform sampling algorithm vs Always Engage sampling algorithm. Randomly assigned underlying distributions. Both unbalanced (top) and Zipfian (bottom) arrival distributions are used.

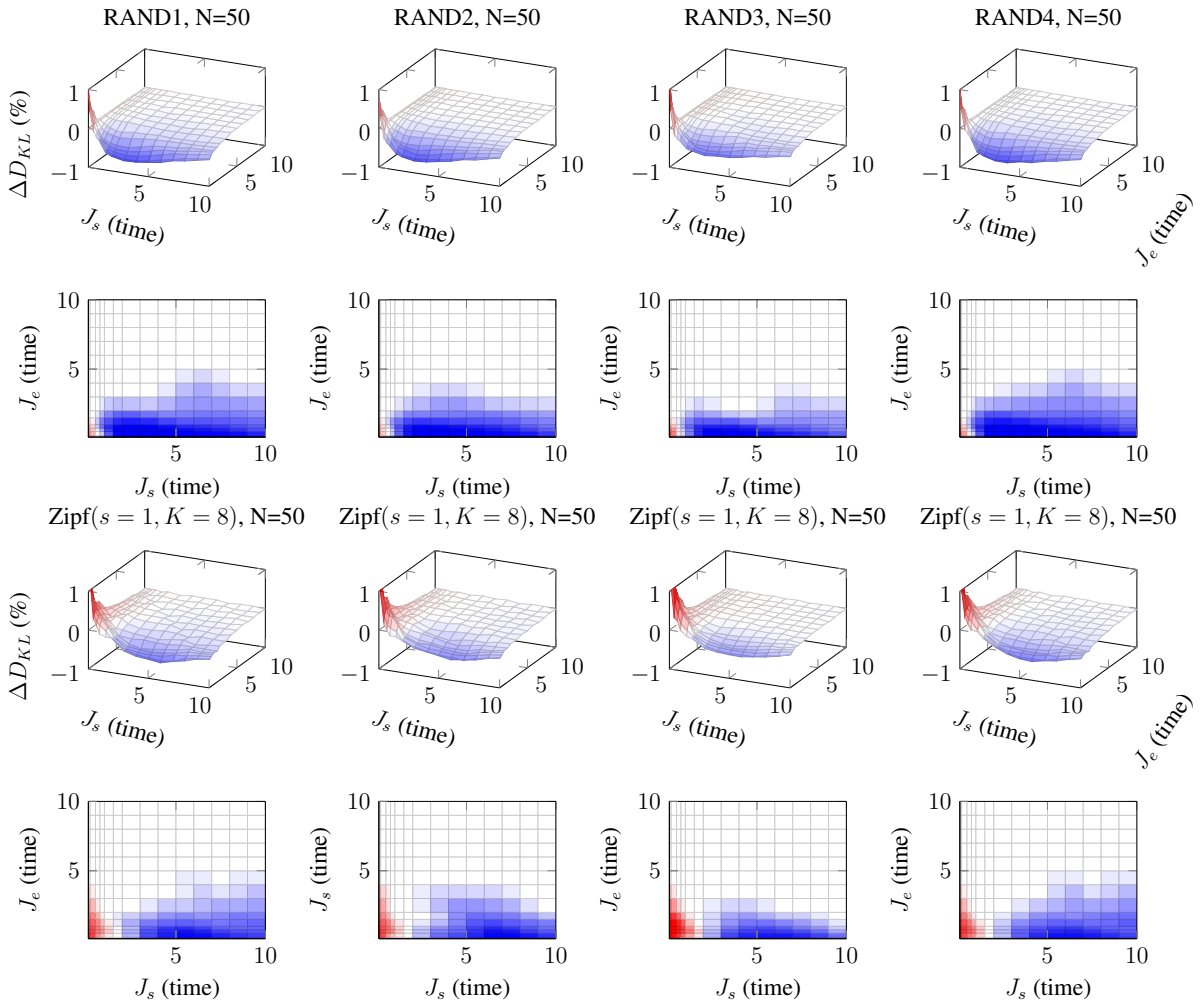


Figure C.14: Greedy sampling algorithm vs Always Engage sampling algorithm. Randomly assigned underlying distributions. Both unbalanced (top) and Zipfian (bottom) arrival distributions are used.

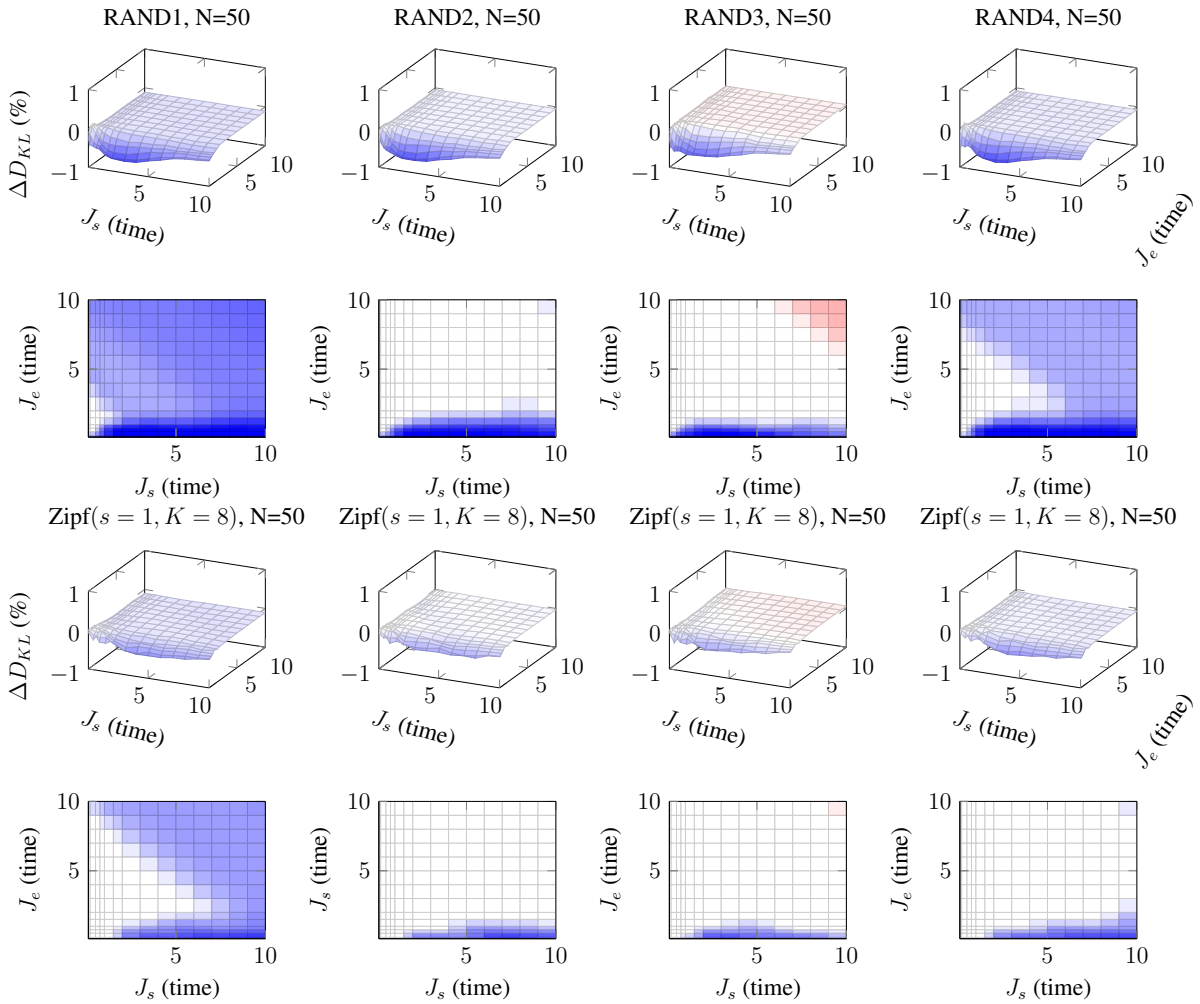


Figure C.15: Foraging sampling algorithm vs Always Engage sampling algorithm. Randomly assigned underlying distributions. Both unbalanced (top) and Zipfian (bottom) arrival distributions are used.

C.5 Experiment Parameters

	Parameter	**Purpose**
Simulation	Arrival Distribution - $P(X)$	Governs which class of the next object encountered by the algorithm.
	Underlying Distribution - $P(Z\|X = x)$	Governs the observations, z, collected by querying an object of class $x \in X$ with the secondary sensor.
	Number of object classes - K	The number of different types of objects existing in the environment the robot is exploring.
	Cost of collecting samples - J_{sample}	The cost of extracting samples. In the experiments we consider the cost to be time, but it could be energy, or some other quantity of interest to experimenters.
	Cost of exploration - $J_{explore}$	This is the cost incurred by the robot as it searches for the next object. As with sampling cost, we pose the cost in terms of time.
	Budget - T	The total quantity of resources the robot has to spend. The sampling and exploration costs draw down on this budget.

Appendix D

Opportunistic Sampling in a Scalar Field (Prospecting) Supplemental Material

D.1 Experiment 1 - Change in Underlying Distribution Rate

The following tables show the effect size comparing the SPRT algorithm to the Memory Threshold algorithm.

Table D.1: The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 2.

$\lambda_2 - \lambda_1$	$\mu[FN]_{MemThresh} - \mu[FN]_{SPRT(s=2)}$	**95% HDI**	**Effect Size (Cohen's d)**
-30	1.00	$[1, 1]$	1.98×10^3
-20	1.00	$[1, 1]$	1.98×10^3
-10	0.99	$[0.979, 0.993]$	68.8
0	0.49	$[0.380, 0.589]$	1.86
10	0.98	$[0.971, 0.988]$	49.6
20	1.00	$[1, 1]$	1.98×10^3
30	0.88	$[0.867, 0.897]$	29.6
40	0.48	$[0.460, 0.500]$	9.79
50	0.12	$[0.110, 0.139]$	3.51
60	0.02	$[0.016, 0.027]$	1.51

The following tables show the effect size on the mean error rate in predicting the time of change as a function of the change in the underlying rate.

Table D.2: The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 4.

$\lambda_2 - \lambda_1$	$\mu[FN]_{MemThresh} - \mu[FN]_{SPRT(s=4)}$	**95% HDI**	**Effect Size (Cohen's d)**
-30	1.00	$[1, 1]$	1.98×10^3
-20	1.00	$[0.995, 1]$	1.96×10^3
-10	0.95	$[0.932, 0.958]$	30.20
0	0.00	$[0.000, 0.000]$	0.02
10	0.95	$[0.934, 0.953]$	32.050
20	0.99	$[0.986, 1.000]$	133.00
30	0.88	$[0.868, 0.892]$	36.70
40	0.48	$[0.462, 0.500]$	9.81
50	0.12	$[0.110, 0.139]$	3.53
60	0.02	$[0.015, 0.027]$	1.54

Table D.3: The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 8.

$\lambda_2 - \lambda_1$	$\mu[FN]_{MemThresh} - \mu[FN]_{SPRT(s=8)}$	**95% HDI**	**Effect Size (Cohen's d)**
-30	1.00	$[1.000, 1.000]$	1.98×10^3
-20	0.98	$[0.975, 0.983]$	99.30
-10	0.87	$[0.862, 0.884]$	16.10
0	0.00	$[0.000, 0.000]$	0.01
10	0.87	$[0.844, 0.903]$	13.50
20	0.98	$[0.968, 0.981]$	96.80
30	0.87	$[0.858, 0.882]$	28.60
40	0.48	$[0.459, 0.504]$	11.00
50	0.12	$[0.110, 0.138]$	3.55
60	0.02	$[0.015, 0.027]$	1.54

The follow plots show the size of the effect on the false positive rate as a function of the change in the underlying rate.

Table D.4: The size of the effect on the false positive rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 2. Here the SPRT 2 algorithm is demonstrably worse in terms of false positive rate than the memory threshold algorithm.

Rate Change $\lambda_2 - \lambda_1$	$\mu[FP]_{MemThresh} - \mu[FP]_{SPRT(s=2)}$	**95% HDI**	**Effect Size (Cohen's d)**
-30	-0.25	$[-0.316, -0.179]$	-1.61
-20	-0.25	$[-0.313, -0.176]$	-1.65
-10	-0.25	$[-0.314, -0.176]$	-1.66
0	-0.25	$[-0.316, -0.179]$	-1.60
10	-0.25	$[-0.313, -0.177]$	-1.67
20	-0.25	$[-0.317, -0.177]$	-1.64
30	-0.25	$[-0.314, -0.177]$	-1.60
40	-0.25	$[-0.317, -0.180]$	-1.67
50	-0.25	$[-0.317, -0.178]$	-1.67
60	-0.25	$[-0.316, -0.179]$	-1.62

Table D.5: The size of the effect on the false positive rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 4. The performance of the two algorithms is indistinguishable at a 95% confidence level.

Rate Change $\lambda_2 - \lambda_1$	$\mu[FP]_{MemThresh} - \mu[FP]_{SPRT(s=4)}$	**95% HDI**	**Effect Size (Cohen's d)**
-30	0.00	$[-0.014, 0.0001]$	N/A
-20	0.00	$[-0.016, 0.0001]$	N/A
-10	0.00	$[0.000, 0.000]$	N/A
0	0.00	$[0.000, 0.000]$	N/A
10	0.00	$[-0.010, 0.0001]$	N/A
20	-0.01	$[-0.018, 0.0001]$	N/A
30	-0.01	$[-0.018, 0.00003]$	N/A
40	-0.002	$[-0.015, 0.0002]$	N/A
50	-0.001	$[-0.009, 0.0002]$	N/A
60	0.00	$[0.000, 0.000]$	N/A

Table D.6: The size of the effect on the false positive rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 8. The two algorithms are indistinguishable at a 95% confidence level.

Rate Change $\lambda_2 - \lambda_1$	$\mu[FP]_{MemThresh} - \mu[FP]_{SPRT(s=8)}$	**95% HDI**	**Effect Size (Cohen's d)**
-30	0.00	$[0.0, 0.0]$	N/A
-20	0.00	$[0.0, 0.0]$	N/A
-10	0.00	$[0.0, 0.0]$	N/A
0	0.00	$[0.0, 0.0]$	N/A
10	0.00	$[0.0, 0.0]$	N/A
20	0.00	$[0.0, 0.0]$	N/A
30	0.00	$[0.0, 0.0]$	N/A
40	0.00	$[0.0, 0.0]$	N/A
50	0.00	$[0.0, 0.0]$	N/A
60	0.00	$[0.0, 0.0]$	N/A

Table D.7: The size of the effect on the mean error using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 2. The two algorithms are indistinguishable at a 95% confidence level.

Rate Change $\lambda_2 - \lambda_1$	$\mu[FP]_{MemThresh} - \mu[FP]_{SPRT(s=2)}$	**95% HDI**	**Effect Size (Cohen's d)**
-30	44.15	$[45.683, 42.617]$	56.44
-20	44.04	$[45.569, 42.508]$	56.41
-10	42.77	$[44.343, 41.195]$	53.26
0	24.08	$[27.045, 21.106]$	15.89
10	42.95	$[44.501, 41.407]$	54.42
20	35.48	$[40.028, 30.927]$	15.28
30	18.80	$[21.248, 16.357]$	15.07
40	18.74	$[20.391, 17.086]$	22.22
50	13.12	$[14.945, 11.305]$	14.14
60	2.11	$[-0.150, 4.365,]$	N/A

Table D.8: The size of the effect on the mean error using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 4. The two algorithms are indistinguishable at a 95% confidence level.

Rate Change $\lambda_2 - \lambda_1$	$\mu[FP]_{MemThresh} - \mu[FP]_{SPRT(s=4)}$	**95% HDI**	**Effect Size (Cohen's d)**
-30	49.62	$[49.834, 49.409]$	457.27
-20	49.50	$[49.723, 49.285]$	443.31
-10	48.20	$[48.759, 47.648]$	170.17
0	18.06	$[22.923, 13.205]$	7.29
10	48.41	$[48.850, 47.962]$	213.69
20	40.96	$[45.243, 36.671]$	18.73
30	24.26	$[26.179, 22.346]$	24.81
40	24.21	$[24.850, 23.562]$	73.69
50	18.60	$[19.598, 17.594]$	36.36
60	7.58	$[9.249, 5.908]$	8.89

Table D.9: The size of the effect on the mean error using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 8. The two algorithms are indistinguishable at a 95% confidence level.

Rate Change $\lambda_2 - \lambda_1$	$\mu[FP]_{MemThresh} - \mu[FP]_{SPRT(s=8)}$	**95% HDI**	**Effect Size (Cohen's d)**
-30	50.00	$[50.0, 50.0]$	inf
-20	49.89	$[49.975, 49.807]$	1167.42
-10	48.62	$[49.145, 48.088]$	180.34
0	0.82	$[-0.771, 2.411]$	N/A
10	48.89	$[49.266, 48.508]$	252.90
20	41.34	$[45.617, 37.055]$	18.93
30	24.64	$[26.544, 22.735]$	25.35
40	24.58	$[25.191, 23.977]$	79.33
50	18.97	$[19.953, 17.995]$	37.97
60	7.96	$[9.614, 6.300]$	9.41

D.2 Experiment 2 - Effect of Delay of Change Onset

Table D.10: The average reduction in false positive rate due to using the SPRT algorithm vs the Memory Threshold algorithm.

Change Point (sec)	$\mu[FP]_{MemThresh} - \mu[FP]_{sprt}$	**95% HDI**	**Effect Size (Cohen's d)**
40	0.016	$[0.009, 0.022]$	0.96
80	0.016	$[0.011, 0.020]$	1.96
120	0.017	$[0.014, 0.020]$	2.84
160	0.016	$[0.013, 0.018]$	3.13
200	0.016	$[0.014, 0.018]$	5.08
240	0.017	$[0.015, 0.020]$	4.72
260	0.017	$[0.015, 0.019]$	4.61
320	0.017	$[0.015, 0.019]$	4.94
360	0.017	$[0.015, 0.019]$	5.72

Table D.11: The reduction in FN rate from the Memory Threshold to the SPRT algorithm.

Change Point (sec)	$\mu[FN]_{MemThresh} - \mu[FN]_{sprt}$	**95% HDI**	**Effect Size (Cohen's d)**
40	0.029	$[0.027, 0.031]$	5.14
80	0.029	$[0.026, 0.031]$	4.78
120	0.028	$[0.026, 0.031]$	4.41
160	0.028	$[0.025, 0.031]$	4.18
200	0.027	$[0.025, 0.030]$	6.19
240	0.026	$[0.022, 0.030]$	2.65
260	0.025	$[0.021, 0.030]$	2.52
320	0.025	$[0.019, 0.031]$	1.57
360	0.018	$[0.008, 0.027]$	0.73

Table D.12: The effect size of the reduction in mean error as the change point is varied.

Change Point (sec)	$\mu[MeanError]_{MemThresh} - \mu[MeanError]_{sprt}$	**95% HDI**	**Effect Size (Cohen's d)**
40	155	$[155, 157]$	39.3
80	136	$[134, 138]$	44.4
120	115	$[112, 117]$	25.1
160	94.2	$[91.6, 96.9]$	19.1
200	73.9	$[70.7, 77]$	13.8
240	51.9	$[49.2, 54.5]$	10.4
260	42.6	$[40.2, 45]$	9.47
320	11.8	$[9.35, 14.2]$	2.81
360	16.1	$[-18.9, -13.4]$	-3.42

D.3 Experiment 3 - Real MVP Data

D.3.1 Data Supporting ROC Curves on MVP Data

Table D.13: True positive rate and false positive rate for the SPRT algorithm on all days of the MVP data.

s	**TPR**	**FPR**
1.00	0.9789	0.0201
2.00	0.9263	0.0072
3.00	0.9053	0.0042
4.00	0.8737	0.0026
5.00	0.8105	0.0018
6.00	0.7474	0.0011
7.00	0.7053	0.0008
8.00	0.7053	0.0003
9.00	0.6632	0.0002
10.00	0.6105	0.0002

Table D.14: True positive rate and false positive rate for the Memory Threshold algorithm as the threshold is varied.

γ	**TPR**	**FPR**
10.00	0.0000	0.0000
20.00	0.0000	0.0000
30.00	0.0737	0.0007
33.00	0.1474	0.0117
35.00	0.3579	0.0297
37.00	0.4737	0.0436
40.00	0.6842	0.0447
43.00	0.8421	0.0465
45.00	0.8421	0.0744
47.00	0.8737	0.1030
50.00	0.6316	0.0686
52.00	0.3474	0.0308
55.00	0.2000	0.0084
57.00	0.1474	0.0067
60.00	0.0947	0.0019
70.00	0.0316	0.0000
80.00	0.0211	0.0000
90.00	0.0211	0.0000
100.00	0.0000	0.0014
110.00	0.0000	0.0000

Table D.15: True positive rate and false positive rate for the Relative Change algorithm as the threshold for the High Density Interval was changed.

δ	**TPR**	**FPR**
0.85	0.6421	0.0078
0.90	0.4842	0.0074
0.93	0.4105	0.0065
0.95	0.3368	0.0024
0.97	0.2316	0.0015
0.99	0.2211	0.0009
0.99	0.1368	0.0005
1.00	0.1158	0.0004
1.00	0.0947	0.0001

Table D.16: True positive rate and false positive rate for the Adaptive Threshold algorithm as the suggested number of AIMs and the minimum threshold is varied. Results here for $N_s = 5$. We selected an initial threshold of $\gamma_B = 43$ because that was the best performing threshold for the Memory Threshold algorithm.

γ_B	N_s	γ_{Min}	**TPR**	**FPR**
43.00	5.00	20.00	0.8421	0.0465
43.00	5.00	25.00	0.8105	0.0438
43.00	5.00	30.00	0.7789	0.0424
43.00	5.00	33.00	0.7474	0.0421
43.00	5.00	35.00	0.6737	0.0361
43.00	5.00	37.00	0.4316	0.0263
43.00	5.00	40.00	0.2211	0.0173
43.00	5.00	43.00	0.2316	0.0122
43.00	5.00	45.00	0.2000	0.0159
43.00	5.00	47.00	0.2211	0.0185
43.00	5.00	50.00	0.2105	0.0197
43.00	5.00	53.00	0.2842	0.0232
43.00	5.00	55.00	0.3474	0.0239
43.00	5.00	60.00	0.6000	0.0360
43.00	5.00	65.00	0.7789	0.0410

Table D.17: True positive rate and false positive rate for the Adaptive Threshold algorithm as the suggested number of AIMs and the minimum threshold is varied. Results here for $N_s = 10$. We selected an initial threshold of $\gamma_B = 43$ because that was the best performing threshold for the Memory Threshold algorithm.

γ_B	N_s	γ_{Min}	**TPR**	**FPR**
43.00	10.00	20.00	0.8421	0.0465
43.00	10.00	25.00	0.8105	0.0438
43.00	10.00	30.00	0.7789	0.0424
43.00	10.00	33.00	0.7474	0.0421
43.00	10.00	35.00	0.6737	0.0361
43.00	10.00	37.00	0.4316	0.0263
43.00	10.00	40.00	0.2211	0.0173
43.00	10.00	43.00	0.2632	0.0155
43.00	10.00	45.00	0.2000	0.0164
43.00	10.00	47.00	0.2211	0.0185
43.00	10.00	50.00	0.2105	0.0197
43.00	10.00	53.00	0.2842	0.0232
43.00	10.00	55.00	0.3474	0.0239
43.00	10.00	60.00	0.6000	0.0360
43.00	10.00	65.00	0.7789	0.0410

Table D.18: True positive rate and false positive rate for the Adaptive Threshold algorithm as the suggested number of AIMs and the minimum threshold is varied. Results here for $N_s = 15$. We selected an initial threshold of $\gamma_B = 43$ because that was the best performing threshold for the Memory Threshold algorithm.

γ_B	N_s	γ_{Min}	**TPR**	**FPR**
43.00	15.00	20.00	0.8421	0.0465
43.00	15.00	25.00	0.8105	0.0438
43.00	15.00	30.00	0.7789	0.0424
43.00	15.00	33.00	0.7474	0.0421
43.00	15.00	35.00	0.6737	0.0362
43.00	15.00	37.00	0.4421	0.0265
43.00	15.00	40.00	0.2316	0.0173
43.00	15.00	43.00	0.2737	0.0164
43.00	15.00	45.00	0.2105	0.0190
43.00	15.00	47.00	0.2211	0.0198
43.00	15.00	50.00	0.2105	0.0200
43.00	15.00	53.00	0.2842	0.0232
43.00	15.00	55.00	0.3474	0.0239
43.00	15.00	60.00	0.6000	0.0360
43.00	15.00	65.00	0.7789	0.0410

Table D.19: True positive rate and false positive rate for the Adaptive Threshold algorithm as the suggested number of AIMs and the minimum threshold is varied. Results here for $N_s = 20$. We selected an initial threshold of $\gamma_B = 43$ because that was the best performing threshold for the Memory Threshold algorithm.

γ_B	N_s	γ_{Min}	**TPR**	**FPR**
43.00	20.00	20.00	0.8421	0.0465
43.00	20.00	25.00	0.8105	0.0438
43.00	20.00	30.00	0.7789	0.0424
43.00	20.00	33.00	0.7474	0.0421
43.00	20.00	35.00	0.6947	0.0395
43.00	20.00	37.00	0.4526	0.0290
43.00	20.00	40.00	0.2316	0.0178
43.00	20.00	43.00	0.2737	0.0161
43.00	20.00	45.00	0.2526	0.0267
43.00	20.00	47.00	0.2211	0.0200
43.00	20.00	50.00	0.2105	0.0207
43.00	20.00	53.00	0.2842	0.0232
43.00	20.00	55.00	0.3474	0.0241
43.00	20.00	60.00	0.6000	0.0343
43.00	20.00	65.00	0.7789	0.0393

Table D.20: True positive rate and false positive rate for the Adaptive Threshold algorithm as the suggested number of AIMs and the minimum threshold is varied. Results here for $N_s = 25$. We selected an initial threshold of $\gamma_B = 43$ because that was the best performing threshold for the Memory Threshold algorithm.

γ_B	N_s	γ_{Min}	**TPR**	**FPR**
43.00	25.00	20.00	0.8421	0.0465
43.00	25.00	25.00	0.8105	0.0438
43.00	25.00	30.00	0.7789	0.0424
43.00	25.00	33.00	0.7474	0.0421
43.00	25.00	35.00	0.7053	0.0395
43.00	25.00	37.00	0.4632	0.0291
43.00	25.00	40.00	0.2421	0.0179
43.00	25.00	43.00	0.2842	0.0180
43.00	25.00	45.00	0.2211	0.0207
43.00	25.00	47.00	0.2316	0.0217
43.00	25.00	50.00	0.2105	0.0216
43.00	25.00	53.00	0.2842	0.0232
43.00	25.00	55.00	0.3474	0.0244
43.00	25.00	60.00	0.6000	0.0343
43.00	25.00	65.00	0.7789	0.0393

Table D.21: True positive rate and false positive rate for the Adaptive Threshold algorithm as the suggested number of AIMs and the minimum threshold is varied. Results here for $N_s = 30$. We selected an initial threshold of $\gamma_B = 43$ because that was the best performing threshold for the Memory Threshold algorithm.

γ_B	N_s	γ_{Min}	**TPR**	**FPR**
43.00	30.00	20.00	0.8421	0.0465
43.00	30.00	25.00	0.8105	0.0438
43.00	30.00	30.00	0.7789	0.0424
43.00	30.00	33.00	0.7474	0.0421
43.00	30.00	35.00	0.7053	0.0395
43.00	30.00	37.00	0.4842	0.0308
43.00	30.00	40.00	0.2842	0.0203
43.00	30.00	43.00	0.3684	0.0242
43.00	30.00	45.00	0.2526	0.0263
43.00	30.00	47.00	0.2316	0.0235
43.00	30.00	50.00	0.2211	0.0222
43.00	30.00	53.00	0.2842	0.0235
43.00	30.00	55.00	0.3474	0.0245
43.00	30.00	60.00	0.6000	0.0343
43.00	30.00	65.00	0.7789	0.0393

Table D.22: True positive rate and false positive rate for the Adaptive Threshold algorithm as the suggested number of AIMs and the minimum threshold is varied. Results here for $N_s = 35$. We selected an initial threshold of $\gamma_B = 43$ because that was the best performing threshold for the Memory Threshold algorithm.

γ_B	N_s	γ_{Min}	**TPR**	**FPR**
43.00	35.00	20.00	0.8421	0.0465
43.00	35.00	25.00	0.8105	0.0438
43.00	35.00	30.00	0.7789	0.0424
43.00	35.00	33.00	0.7474	0.0421
43.00	35.00	35.00	0.7053	0.0395
43.00	35.00	37.00	0.4842	0.0309
43.00	35.00	40.00	0.3895	0.0268
43.00	35.00	43.00	0.3579	0.0198
43.00	35.00	45.00	0.2632	0.0315
43.00	35.00	47.00	0.2632	0.0260
43.00	35.00	50.00	0.2211	0.0235
43.00	35.00	53.00	0.2842	0.0238
43.00	35.00	55.00	0.3579	0.0246
43.00	35.00	60.00	0.6000	0.0343
43.00	35.00	65.00	0.7789	0.0393

Table D.23: True positive rate and false positive rate for the Adaptive Threshold algorithm as the suggested number of AIMs and the minimum threshold is varied. Results here for $N_s = 40$. We selected an initial threshold of $\gamma_B = 43$ because that was the best performing threshold for the Memory Threshold algorithm.

γ_B	N_s	γ_{Min}	**TPR**	**FPR**
43.00	40.00	20.00	0.8421	0.0465
43.00	40.00	25.00	0.8105	0.0438
43.00	40.00	30.00	0.7789	0.0424
43.00	40.00	33.00	0.7474	0.0421
43.00	40.00	35.00	0.7053	0.0395
43.00	40.00	37.00	0.5158	0.0351
43.00	40.00	40.00	0.3579	0.0248
43.00	40.00	43.00	0.3579	0.0235
43.00	40.00	45.00	0.3053	0.0326
43.00	40.00	47.00	0.2526	0.0279
43.00	40.00	50.00	0.2421	0.0252
43.00	40.00	53.00	0.2842	0.0241
43.00	40.00	55.00	0.3579	0.0246
43.00	40.00	60.00	0.6000	0.0343
43.00	40.00	65.00	0.7789	0.0393

Table D.24: True positive rate and false positive rate for the Adaptive Threshold algorithm as the suggested number of AIMs and the minimum threshold is varied. Results here for $N_s = 45$. We selected an initial threshold of $\gamma_B = 43$ because that was the best performing threshold for the Memory Threshold algorithm.

γ_B	N_s	γ_{Min}	**TPR**	**FPR**
43.00	45.00	20.00	0.8421	0.0465
43.00	45.00	25.00	0.8105	0.0438
43.00	45.00	30.00	0.7789	0.0424
43.00	45.00	33.00	0.7474	0.0421
43.00	45.00	35.00	0.7053	0.0395
43.00	45.00	37.00	0.5158	0.0359
43.00	45.00	40.00	0.3789	0.0268
43.00	45.00	43.00	0.3789	0.0253
43.00	45.00	45.00	0.3263	0.0331
43.00	45.00	47.00	0.3053	0.0298
43.00	45.00	50.00	0.2316	0.0262
43.00	45.00	53.00	0.2842	0.0248
43.00	45.00	55.00	0.3368	0.0232
43.00	45.00	60.00	0.6000	0.0343
43.00	45.00	65.00	0.7789	0.0393

Table D.25: True positive rate and false positive rate for the Adaptive Threshold algorithm as the suggested number of AIMs and the minimum threshold is varied. Results here for $N_s = 50$. We selected an initial threshold of $\gamma_B = 43$ because that was the best performing threshold for the Memory Threshold algorithm.

γ_B	N_s	γ_{Min}	**TPR**	**FPR**
43.00	50.00	20.00	0.8421	0.0465
43.00	50.00	25.00	0.8105	0.0438
43.00	50.00	30.00	0.7789	0.0424
43.00	50.00	33.00	0.7474	0.0421
43.00	50.00	35.00	0.7053	0.0395
43.00	50.00	37.00	0.5263	0.0381
43.00	50.00	40.00	0.4947	0.0376
43.00	50.00	43.00	0.4105	0.0251
43.00	50.00	45.00	0.3895	0.0366
43.00	50.00	47.00	0.3158	0.0297
43.00	50.00	50.00	0.2421	0.0283
43.00	50.00	53.00	0.3053	0.0256
43.00	50.00	55.00	0.3368	0.0232
43.00	50.00	60.00	0.6000	0.0343
43.00	50.00	65.00	0.7789	0.0393

Table D.26: True positive rate and false positive rate for the Adaptive Threshold algorithm as the suggested number of AIMs and the minimum threshold is varied. Results here for $N_s = 55$. We selected an initial threshold of $\gamma_B = 43$ because that was the best performing threshold for the Memory Threshold algorithm.

γ_B	N_s	γ_{Min}	**TPR**	**FPR**
43.00	55.00	20.00	0.8421	0.0465
43.00	55.00	25.00	0.8105	0.0438
43.00	55.00	30.00	0.7789	0.0424
43.00	55.00	33.00	0.7474	0.0421
43.00	55.00	35.00	0.7474	0.0429
43.00	55.00	37.00	0.5053	0.0320
43.00	55.00	40.00	0.4632	0.0384
43.00	55.00	43.00	0.4421	0.0259
43.00	55.00	45.00	0.3684	0.0283
43.00	55.00	47.00	0.3263	0.0323
43.00	55.00	50.00	0.2526	0.0297
43.00	55.00	53.00	0.3053	0.0267
43.00	55.00	55.00	0.3368	0.0233
43.00	55.00	60.00	0.6000	0.0343
43.00	55.00	65.00	0.7789	0.0393

Table D.27: True positive rate and false positive rate for the Adaptive Threshold algorithm as the suggested number of AIMs and the minimum threshold is varied. Results here for $N_s = 60$. We selected an initial threshold of $\gamma_B = 43$ because that was the best performing threshold for the Memory Threshold algorithm.

γ_B	N_s	γ_{Min}	**TPR**	**FPR**
43.00	60.00	20.00	0.8421	0.0465
43.00	60.00	25.00	0.8105	0.0438
43.00	60.00	30.00	0.7789	0.0424
43.00	60.00	33.00	0.7474	0.0421
43.00	60.00	35.00	0.7579	0.0438
43.00	60.00	37.00	0.5474	0.0399
43.00	60.00	40.00	0.4316	0.0394
43.00	60.00	43.00	0.5053	0.0276
43.00	60.00	45.00	0.4211	0.0417
43.00	60.00	47.00	0.3263	0.0331
43.00	60.00	50.00	0.2421	0.0310
43.00	60.00	53.00	0.3053	0.0271
43.00	60.00	55.00	0.3368	0.0215
43.00	60.00	60.00	0.6000	0.0343
43.00	60.00	65.00	0.7789	0.0393

D.3.2 ROC for Adaptive Threshold Algorithm as a Function of γ_B

To give a better exploration of the Adaptive Threshold algorithm, Fig. D.1 shows an ROC curve with just the performance of the Adaptive Threshold algorithm and the SPRT algorithm. The parameters for the SPRT algorithm maked by the blue circles and for the Adaptive algorithm marked yellow squares are given in Table 4.7. For the other settings of the Adaptive Threshold algorithm the parameters are given in Table D.28.

Table D.28: Adaptive Threshold algorithm parameters explored in Fig. D.1.

Parameter	**Settings**
γ_B	$\{40, 50, 60, 80\}$
N_s	$\{5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60\}$
γ_{min}	$\{20, 25, 30, 35, 40, 45, 50, 55, 60, 65\}$

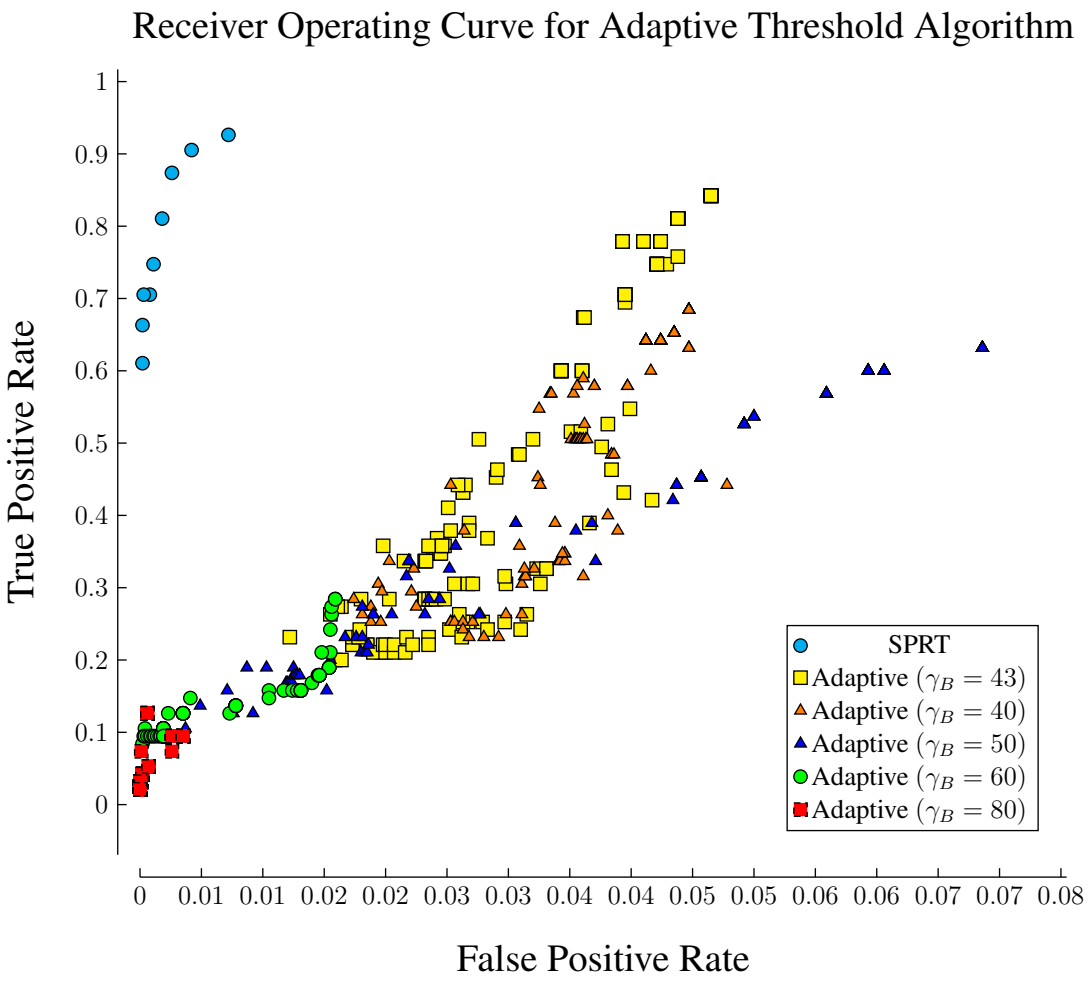


Figure D.1: The Adaptive Threshold algorithm is sensitive to the initial threshold parameter, γ_B, in addition to the other parameters of the algorithm. The settings of the Adaptive algorithm which give comparable false positive rates to the SPRT algorithm have worse true positive rates, and those settings which give comparable true positive rates have worse false positive rates.

D.4 Experiment 4 - Effect on Performance in 2D Operations

Here we report the number of maxima in the underlying maps which were observed by the robot using the different algorithms, as well as the number of AIMs deployed. For each map 10 trials were run, and we report the average number and standard deviation for the different quantities observed.

D.4.1 Results Using Fixed Parameters

Here we provide the results for the algorithms using the parameters that were either specified as part of the MVP project. The parameters are given in Table D.29.

Table D.29: Parameters used to produce results in Table D.30 and Table D.31.

Algorithm	**Parameters**	**Value**
Memory Threshold	γ	80
SPRT	s	8
Relative Change	δ	0.95
Adaptive	γ_B	80
	N_s	55
	γ_{min}	53

Table D.30: The average number of peaks captured by the three different algorithms on each map. We report average $\pm$ one standard deviation, $n = 10$.

Map Number	No AIMs	Memory Threshold	Adaptive Threshold	Relative Change	SPRT
00	10.0 ± 0.0	16.6 ± 0.3	47.4 ± 4.1	88.2 ± 4.0	42.9 ± 3.1
01	11.0 ± 0.0	11.0 ± 0.0	44.8 ± 3.4	87.4 ± 3.8	44.1 ± 2.5
02	20.0 ± 0.0	21.0 ± 0.0	48.9 ± 2.9	95.5 ± 2.9	53.5 ± 4.8
03	13.0 ± 0.0	16.6 ± 0.3	49.0 ± 2.5	95.1 ± 1.9	46.3 ± 3.3
04	15.0 ± 0.0	16.1 ± 0.7	50.5 ± 3.7	96.9 ± 2.9	54.1 ± 2.5
05	12.0 ± 0.0	13.2 ± 1.0	50.2 ± 7.1	90.4 ± 4.0	53.1 ± 3.0
06	19.0 ± 0.0	19.2 ± 0.4	49.0 ± 5.8	88.6 ± 3.3	56.9 ± 1.9
07	16.0 ± 0.0	17.4 ± 0.4	43.8 ± 5.2	88.9 ± 3.1	49.9 ± 3.0
08	12.0 ± 0.0	15.9 ± 0.4	44.7 ± 5.3	89.7 ± 3.2	48.3 ± 2.5
09	17.0 ± 0.0	18.2 ± 0.4	47.0 ± 4.4	92.3 ± 3.5	53.0 ± 2.9
10	12.0 ± 0.0	12.4 ± 0.4	43.1 ± 5.8	93.9 ± 2.6	51.0 ± 3.6
11	15.0 ± 0.0	15.1 ± 0.2	47.3 ± 3.2	93.7 ± 4.8	48.7 ± 2.0
12	18.0 ± 0.0	19.4 ± 0.4	51.1 ± 6.7	100.8 ± 2.6	53.0 ± 2.7
13	19.0 ± 0.0	19.0 ± 0.0	52.1 ± 3.1	93.1 ± 2.3	52.0 ± 1.9
14	24.0 ± 0.0	24.1 ± 0.2	51.0 ± 4.2	96.9 ± 2.5	56.9 ± 2.7
15	17.0 ± 0.0	19.1 ± 0.7	52.8 ± 3.3	97.5 ± 3.2	53.4 ± 4.5
16	19.0 ± 0.0	24.2 ± 0.4	50.1 ± 5.9	100.2 ± 4.5	59.9 ± 3.4
17	16.0 ± 0.0	16.0 ± 0.0	53.1 ± 5.1	97.2 ± 3.8	54.4 ± 2.7
18	18.0 ± 0.0	21.1 ± 1.2	48.5 ± 3.3	88.1 ± 3.2	50.3 ± 3.1
19	10.0 ± 0.0	12.3 ± 0.9	38.2 ± 3.6	92.3 ± 2.9	46.9 ± 3.2
20	22.0 ± 0.0	23.4 ± 0.4	52.5 ± 4.3	93.9 ± 2.1	56.4 ± 2.5
21	20.0 ± 0.0	21.6 ± 0.6	57.3 ± 5.4	95.1 ± 2.1	51.6 ± 2.3
22	19.0 ± 0.0	19.5 ± 0.3	55.0 ± 5.6	108.9 ± 3.2	63.5 ± 2.5
23	16.0 ± 0.0	16.4 ± 0.4	45.3 ± 3.4	89.4 ± 3.7	51.5 ± 2.4
24	18.0 ± 0.0	20.5 ± 0.7	47.8 ± 3.0	91.1 ± 3.5	56.3 ± 3.8
25	13.0 ± 0.0	15.8 ± 0.4	43.7 ± 3.9	80.2 ± 3.8	42.0 ± 1.4
26	14.0 ± 0.0	16.2 ± 0.4	49.9 ± 5.7	89.8 ± 2.1	49.4 ± 3.5
27	19.0 ± 0.0	20.6 ± 0.6	49.0 ± 2.5	94.3 ± 5.2	53.6 ± 3.2
28	22.0 ± 0.0	24.0 ± 0.5	52.5 ± 5.5	97.3 ± 4.1	55.8 ± 3.5
29	12.0 ± 0.0	12.9 ± 0.7	45.0 ± 3.5	86.0 ± 2.4	52.8 ± 2.7

Table D.31: The average number of AIMs deployed by the different algorithms on each map. We report average $\pm$ one standard deviation, $n = 10$

Map Number	No AIMs	Memory Threshold	Adaptive Threshold	Relative Change	SPRT
00	0.0 ± 0.0	3.3 ± 0.3	60.9 ± 3.1	109.0 ± 1.7	39.9 ± 1.9
01	0.0 ± 0.0	0.0 ± 0.0	53.2 ± 1.9	107.5 ± 1.7	39.8 ± 0.7
02	0.0 ± 0.0	1.1 ± 0.2	56.5 ± 2.2	108.1 ± 0.9	43.8 ± 1.3
03	0.0 ± 0.0	5.4 ± 0.6	56.0 ± 1.8	109.8 ± 2.2	37.3 ± 1.4
04	0.0 ± 0.0	2.6 ± 0.4	56.2 ± 2.3	109.6 ± 1.1	41.9 ± 1.0
05	0.0 ± 0.0	1.0 ± 0.4	53.1 ± 2.7	108.8 ± 0.7	43.2 ± 1.5
06	0.0 ± 0.0	1.2 ± 0.4	52.2 ± 1.8	108.4 ± 1.5	38.2 ± 0.9
07	0.0 ± 0.0	2.0 ± 0.0	53.9 ± 1.9	107.6 ± 1.6	43.6 ± 2.3
08	0.0 ± 0.0	5.1 ± 0.2	54.1 ± 2.4	108.8 ± 2.1	42.0 ± 1.4
09	0.0 ± 0.0	2.2 ± 0.2	54.2 ± 2.6	109.6 ± 1.9	39.8 ± 1.2
10	0.0 ± 0.0	3.3 ± 0.3	55.4 ± 3.2	108.6 ± 1.0	42.3 ± 1.3
11	0.0 ± 0.0	1.1 ± 0.2	63.5 ± 9.5	108.7 ± 1.1	41.3 ± 1.7
12	0.0 ± 0.0	1.1 ± 0.2	54.2 ± 1.9	107.6 ± 2.2	39.5 ± 1.8
13	0.0 ± 0.0	0.0 ± 0.0	55.0 ± 1.8	107.4 ± 1.4	39.0 ± 0.8
14	0.0 ± 0.0	0.2 ± 0.2	55.0 ± 2.0	108.7 ± 2.0	40.5 ± 1.6
15	0.0 ± 0.0	2.2 ± 0.2	57.3 ± 2.4	108.7 ± 1.9	36.9 ± 1.7
16	0.0 ± 0.0	4.8 ± 0.2	57.6 ± 3.3	109.1 ± 0.9	46.1 ± 1.3
17	0.0 ± 0.0	1.0 ± 0.0	55.2 ± 2.9	107.2 ± 1.5	40.7 ± 1.6
18	0.0 ± 0.0	2.1 ± 0.6	56.2 ± 3.2	108.2 ± 1.5	41.1 ± 1.5
19	0.0 ± 0.0	3.3 ± 0.6	53.2 ± 1.2	108.3 ± 1.8	41.9 ± 1.6
20	0.0 ± 0.0	3.0 ± 0.0	57.1 ± 2.4	108.7 ± 1.5	42.0 ± 0.6
21	0.0 ± 0.0	2.0 ± 0.3	56.2 ± 2.6	110.0 ± 2.0	41.2 ± 1.2
22	0.0 ± 0.0	1.0 ± 0.0	86.3 ± 29.9	108.8 ± 1.7	46.3 ± 1.4
23	0.0 ± 0.0	0.3 ± 0.3	54.3 ± 1.7	107.9 ± 1.6	40.8 ± 0.9
24	0.0 ± 0.0	3.7 ± 0.4	52.2 ± 1.8	108.9 ± 1.3	43.9 ± 0.8
25	0.0 ± 0.0	4.8 ± 0.4	59.1 ± 5.3	108.7 ± 1.8	38.6 ± 0.7
26	0.0 ± 0.0	2.2 ± 0.2	82.4 ± 29.4	109.0 ± 1.5	43.5 ± 1.4
27	0.0 ± 0.0	2.1 ± 0.3	56.3 ± 1.8	107.1 ± 2.5	43.1 ± 0.8
28	0.0 ± 0.0	3.6 ± 0.3	59.5 ± 5.8	108.3 ± 1.9	42.4 ± 1.3
29	0.0 ± 0.0	0.7 ± 0.5	55.3 ± 2.8	108.0 ± 1.6	42.3 ± 1.6

D.4.2 Additional Settings of Adaptive Threshold Algorithm

Here we present the results for different parameter settings for the Adaptive Threshold algorithm. These settings did not perform as well as those presented in the main body of the document. We broke the results down into two plots with different settings of the γ_B parameter of the Adaptive Threshold Algorithm. These results are shown in Fig. D.2 and Fig. D.3.

In either plot we can see that the algorithms produce similar trend lines, both below the dashed line that indicates a 1:1 ratio of observations of maxima to AIMs deployed. We also see

that the performance of the algorithm can be varied substantially by the other parameters.

For the higher values of γ_B (Fig. D.3) the range of the number of AIMs deployed is larger than for the lower values of γ_B (Fig. D.2). Specifically, the higher values of γ_B have a lower lower bound on how many AIMs they deploy.

Additionally the number of AIMs deployed by the Adaptive algorithm employing lowever values of γ_B is more densely concentrated in the range $[20, 100]$. The Adaptive algorithm using higher values of γ_B have the number of deployed AIMs distributed across the range $[10, 130]$, approximately.

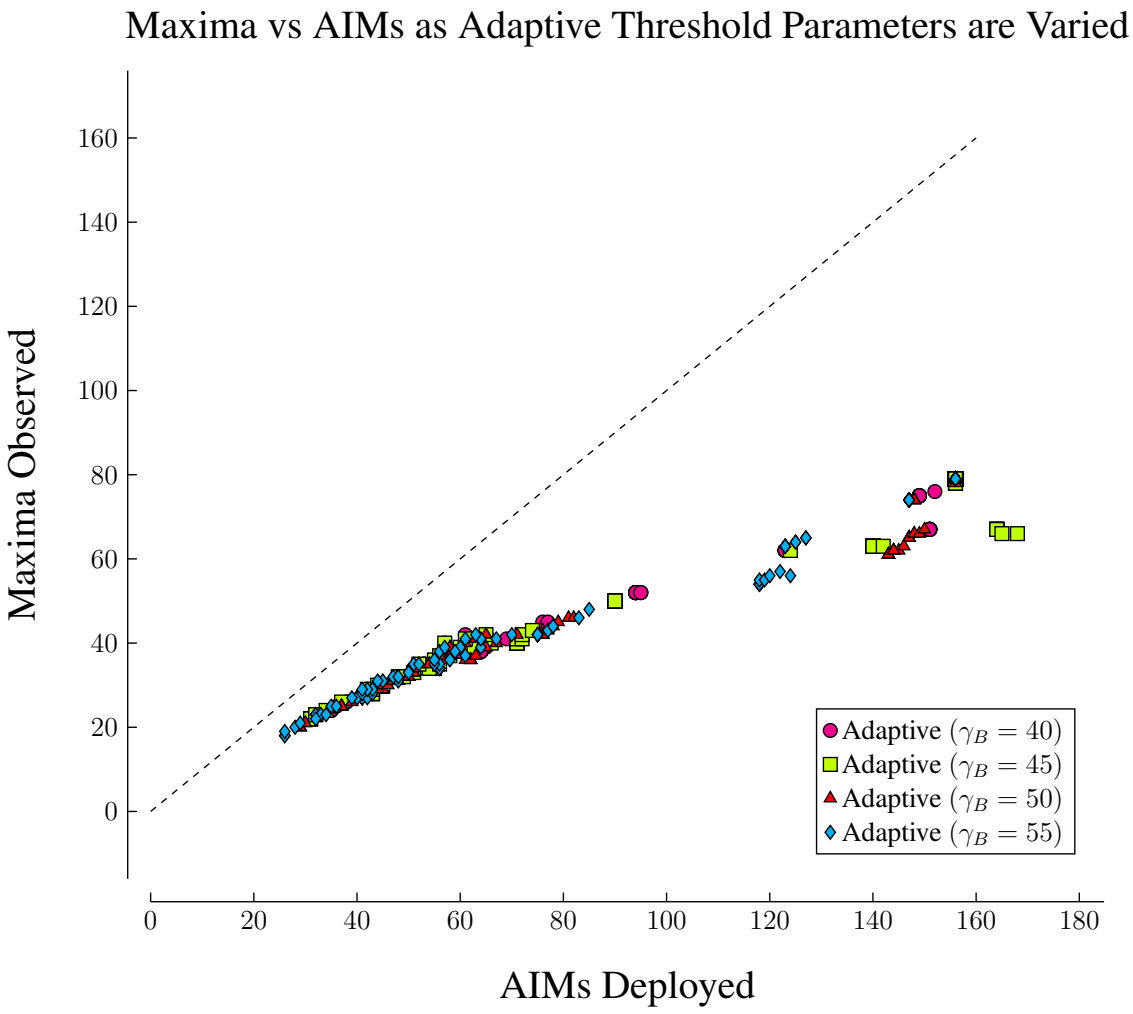


Figure D.2: The individual points in the plot represent the performance of the algorithms averaged over all 30 water maps. As in Fig. 4.22, the y-axis is the number of maxima observed by the algorithm less the maxima that would be observed by following the lawnmower path without deploying any AIMs. The dashed line has a slope of 1.

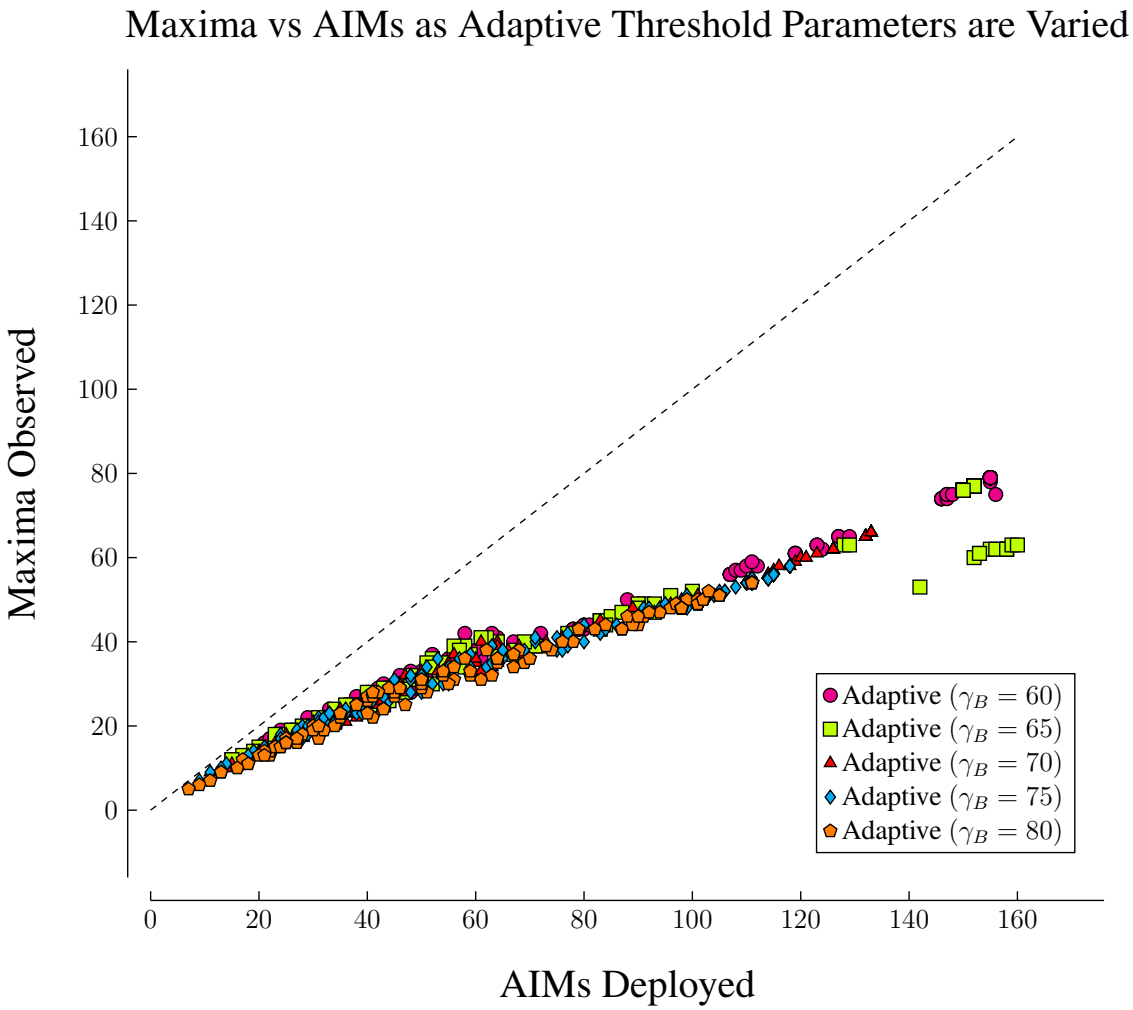


Figure D.3: The individual points in the plot represent the performance of the algorithms averaged over all 30 water maps. As in Fig. 4.22, the y-axis is the number of maxima observed by the algorithm less the maxima that would be observed by following the lawnmower path without deploying any AIMs. The dashed line has a slope of 1.

D.5 Experiment Parameters

Experiment	Parameter	Purpose
2D Simulation	Number of Kernels	Used in generating the water maps. The kernels are location, value pairs that control the generation of the water map. More kernels would mean more variable water maps. Fewer kernels would result in maps with less spatial variance.
	Underlying Distribution	Controls the values used to seed the Kernels in the Voronoi water map generation.
	Instrument Field of View	The radius of the circular field of view of the NSS. This smooths out the hard edges of the Voronoi water map.
	Speed	The speed of the vehicle while driving. We used a speed of $10cm/s$, but changing this speed determines how aliased the water map will be by the
	Sampling Rate	The rate at which the sensor receives readings. We used a value of 1 reading per second, as this was the sampling rate of the NSS. Combined with the speed of the vehicle, this parameter controls how accurately the subsurface is sampled.
	Lawnmower Path Spacing	Controls how far apart the rows of the lawnmower path are spaced. Increasing this number reduces the spacing between switch-backs along the path, and hence lengthens the overall path.
	Outer AIM Radius	To execute our AIMs we followed an Archimedean spiral with a limited outer radius. Changing this radius changes how much of the map is covered when an AIM is deployed.

Algorithm	**Parameter**	**Purpose**
Memory Threshold	Threshold - γ	Sensor readings above γ are considered interesting. AIMs are deployed if readings crosses γ.
Adaptive Threshold	Initial Threshold - γ_B	Readings above this threshold will trigger an AIM.
	Minimum Threshold - γ_{min}	Sets a level to determine whether readings are interesting or not. If γ_{min} is too high, the threshold γ will never be modified. If γ_{min} is too low, then patches will never close, and the threshold will never be modified.
	Number of suggested AIMs - N_s	Controls how readily the robot chooses to deploy AIMs. The higher N_s the more AIMs will be deployed, but it does not put a limit on how many AIMs can be deployed.
	Number of observations below γ_{min} - N_b	The adaptive algorithm requires the identification of patches. Patches are sequences of sensor readings that are above γ_{min}. The N_b is used to determine how many contiguous readings must be below γ_{min} before the patch is considered to be lowered.
Relative Change	High Density Interval - δ	Sensor readings outside the high density interval γ are considered to indicate a change has occurred. Lower values of δ means changes are more likely to be detected. As $\delta \rightarrow 1$ the likelihood of detecting a change goes to zero.
SPRT	Confidence level - s	s sets the confidence level for the SPRT algorithm. As s approaches 1 the likelihood of reporting false positives increases. There is no upper bound on s, but higher values reduce the likelihood of reporting false positives.

D.6 Ergodic Planning vs AIM Deployment

In Chapter 4 we presented a prospecting algorithm which followed a pre-determined path. The simulated robot was operating in a field of water density, like those that would drive the readings of the neutron spectrometer onboard the Resource Prospector robot. The objective of the work was to determine if using change detection could improve the number of maxima in that field that were observed by the robot as the result of deploying discrete Area of Interest Manoeuvres, called AIMs.

The change detection algorithm was agnostic of the underlying planning algorithm, and made the conservative assumption that precursor data were not available. We used a lawnmower path as an exemplar planner, as lawnmower paths have the advantage of being simple and thorough, but they can be inefficient.

Determining which type of planner should be used for mission operations is outside the scope of this thesis. However, the question remains about whether or not a different planner might make for more efficient operations. It has been suggested to the author that a planner using an ergodic cost function may produce the desired behaviour for this robot, and can incorporate precursor map data.

Here we conduct a brief exploration of an approximately optimal ergodic planner using the water density maps employed in Chapter 4. We also discuss some implications of using this planner in a Resource Prospector-like mission scenario.

D.6.1 Method

Miller and Murphey (2013b) use an optimal planner with an ergodic cost function for producing trajectories with high coverage, but which spend proportionally more time in high-value regions of a map. Fundamental to the work is the recognition that trajectories which have the same (or similar) spatial frequency components as the map being explored will produce the desired properties of coverage while accruing high value observations.(Miller and Murphey, 2013b).

The planner is designed to minimize the ergodic error metric, defined in Eq. (D.1). The summation in the error computation is over all possible combinations of the K frequency components along the n different axes of the map. In our experiment $n = 2$, as we are operating in a two-dimensional, rectangular map.

$$\mathcal{E} = \sum_{k \in [0,\ldots,K]^n} \Lambda_k \left| c_k - \mu_k \right|^2 \tag{D.1}$$

Where $\Lambda_k = \frac{1}{(1+\|k\|^2)^s}$ is a weighting term that discounts different frequency components, with $s = (n+1)/2$. The higher frequency components are discounted proportional to the square of the magnitude of the component vector, $\|k\|^2$.

The term c_k (Eq. (D.2)) is the k^{th} time-averaged component of the robot trajectory, and μ_k (Eq. (D.3)) is the k^{th} spatial frequency component of the map. $\mu(x)$ is the value of the density map at location x in the domain, X. X is a rectangular space with sides of length L_i, $i \in \{1, \ldots, n\}$.

$$c_k = \frac{1}{T}\int_0^T f_k(x(t))dt \tag{D.2}$$

$$\mu_k = \int_X \mu(x) f_k(x) dx \tag{D.3}$$

The expression for the Fourier basis functions, f_k used to compute c_k and μ_k is given in Eq. (D.4). h_k is a normalizing constant, given in equation 9 of (Mathew and Mezić, 2011).

$$f_k(x) = \frac{1}{h_k}\prod_{k_i \in k}^{n} cos(\frac{k_i \pi}{L_i}x) \text{ where } k_1, \ldots, k_n \in \{0, \ldots, K\} \tag{D.4}$$

The quantities and methods for computing the cost function are discussed in detail in (Mathew and Mezić, 2011), and the reader is referred there for greater technical detail on the ergodic metric. For our density maps, μ, we use normalized versions of the water maps from Chapter 4.

Algorithms

The robot plans to visit hot spots in the water density maps that were used in Chapter 4. The robot is given the full water map and so does not need to gain information about the environment.

Algorithm D.2 gives the algorithm we used to implement an ergodic planning algorithm. We used a best-first search to explore to find the path for the robot. The best-first search is an approximation of the optimal trajectory planner developed by Miller and Murphy in (Miller and Murphey, 2013b,a; Miller et al., 2016).

Algorithm D.1 Here we note some of the global parameters and the tree node data structure used in representing paths in the approximately optimal ergodic planner.

global K ▷ Number of frequency components
global u ▷ Vehicle speed
global Δt ▷ Simulation time step
global $\mu_1, \ldots, \mu_K$ ▷ Frequency components of map being explored

structure Node(K, x, y)
    parent $\leftarrow$ NULL ▷ Parent node pointer
    $c_k \leftarrow 0 \; \forall \; k \in \{1, \ldots, K\}$ ▷ Per-component integrated path frequency score
    score $\leftarrow 0$
    pos $\leftarrow (x, y)$
    length $\leftarrow 0$ ▷ Length of path from root to this node.
    t $\leftarrow 0$ ▷ Time to follow path from root to this node.
end structure

We use the strategy Miller et al. (2016) use, sequentially planning optimal paths with a limited planning horizon, L_h. At the end of each planning round the best leaf node in the tree is selected

as the new root for the next round of planning. Planning loops are continued until the total path length exceeds the overall path budget L_T.

Algorithm D.2 This is a best-first search algorithm with a random expansion of leaf nodes in the tree of possible trajectories. $\mu_1, \ldots, \mu_K$ are the frequency components of the map being explored. Δt is the timestep for the planner. L_h is the maximum length of one planning horizon and L_T is the total path length budget. The Node data structure is defined in Algorithm D.1.

function PATH_PLANNER($L_h, L_T, N_{children}, N_{goals}$)
  root := Node(K)
  path_length $\leftarrow 0$
  while path_length $\leq L_T$ **do**
    goal $\leftarrow$ ergodic_path_search(root,, $L_h, N_{children}, N_{goals}$)
    root $\leftarrow$ goal ▷ Start the next round of planning from the best goal with the history needed to compute the ergodic objective function.
  end while
  return path(goal) ▷ Follow parent node pointers back to root to find path
end function
function ERGODIC_PATH_SEARCH(root,$L_h, N_{children}, N_{goals}$)
  to_expand := PriorityQueue(root)
  goals := PriorityQueue($\emptyset$)
  while |goals| $< N_{goals}$ **do**
    n $\leftarrow$ to_expand.pop()
    children $\leftarrow$ expand(n,$N_{children}$) ▷ See Algorithm D.3
    for all $c \in$ children **do**
      if c.length $\geq L_h$ **then** ▷ If new point is beyond planning horizon, add to goals
        goals.add((c.score, c))
      else
        to_extend(c)
      end if
    end for
  end while
  return goals.pop()
end function

At each step of the algorithm the approximately optimal planner selects the best node to explore, and then expands the search tree by taking $N_{children} = 5$ random actions, as seen in Algorithm D.3. The random actions were motions for the vehicle defined in Eq. (D.5) for a fixed vehicle speed of $u = 0.1m/s$ and an integration time $\Delta t = 10sec$.

$$a = u\left[cos(\theta), sin(\theta)\right]^T \Delta t \tag{D.5}$$

The direction of motion, θ, is randomly selected from a uniform distribution over $[0, 2\pi]$. Each new node in the tree is assigned a position equal to the position of its parent plus a.

We expand the search tree out to the planning horizon for $N_{goals} = 100$ goals, at which point

the trajectory planner is restarted at the best goal node, with the prior history for computing c_k taken from the goal node.

Algorithm D.3 We randomly expand nodes in the tree and integrate the ergodic score for the path up to the current node, c. $f_k(\cdot)$ is the k^{th} frequency basis function, and Λ_k is the frequency discounting factor, as defined in equations 9 and 12 of (Mathew and Mezić, 2011).

function EXPAND($p, N_{children}$)
  children $\leftarrow$ List($\emptyset$)
  while len(children) $< N_{children}$ **do**
    $\theta \sim$ Uniform$[0, 2\pi]$
    $a = u\left[cos(\theta), sin(\theta)\right]^T \Delta t$
    pos = p.pos $+ a$
    if in_map(pos) **then**
      c := Node(K)
      c.parent $\leftarrow$ p.parent
      c.pos $\leftarrow$ pos
      c.length $\leftarrow$ p.length $+ \|a\|_2$
      $c.t \leftarrow p.t + \Delta t$
      $c.c_k \leftarrow p.c_k + f_k(c.\text{pos})\Delta t \; \forall k \in \{0, \ldots, K\}$
      c.score $\leftarrow \sum_k^K \Lambda_k |(c.c_k/c.t) - \mu_k|^2$
      children.append(c)
    end if
  end while
  return children
end function

Randomly sampling the heading for the vehicle does make the assumption that the robot can change heading instantaneously. This is not necessarily possible for many vehicles, but Resource Prospector is an omni-directional vehicle, so it should be able to follow any path generated.

Randomly changing direction does mean that the algorithm will likely produce paths that are not energy-efficient. Likewise, the generated trajectories may not be feasible for some robots, even if the generated paths are feasible. We ignore these concerns here, but do note them for future implementation.

For comparison, we also test the greedy trajectory planner described by Mathew and Mezić (2011). The control law governing the vehicle is given in equation 26 of (Mathew and Mezić, 2011). This algorithm is included to act as a sanity check of the approximately optimal ergodic planner, and not meant to be interpreted as a recommendation for, or discouragement against, using the greedy planner.

Experiment

To test the performance of the ergodic planner we provided the algorithm with the water maps from Chapter 4, which were normalized to make them density functions. The algorithms were

given a planning horizon of 50m and a total trajectory budget of 1000m. The total length for the lawnmower paths used in Chapter 4 was 550m, these parameters give the algorithm additional margin to explore the map.

We report the number of maxima that were observed by the robot following this trajectory. We compare this performance to a robot that simply follows a lawnmower pattern (No AIMS) and the SPRT algorithm of Chapter 4 with $s = 8$.

The parameters for the ergodic planning algorithms are given in Table D.32. The time step for approximately optimal ergodic planner is much larger than that used for the greedy ergodic planner. The larger timestep was used for the optimal planner to decrease the planning time.

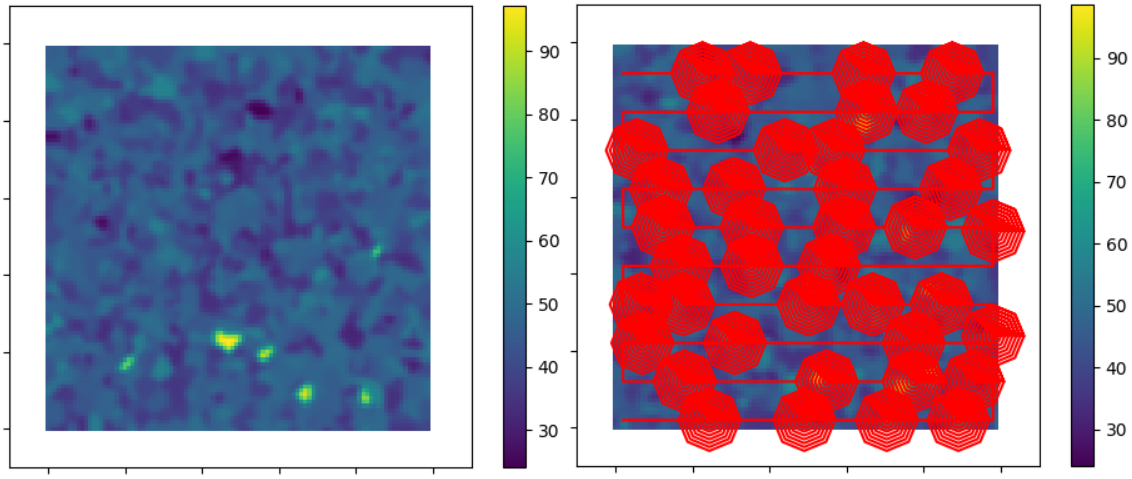
Table D.32: The parameters used in the approximately optimal and greedy ergodic planners.

Parameter	**Optimal Ergodic**	**Greedy Ergodic**
Vehicle Speed - u	0.1m/s	0.1m/s
Number of Frequency Components - K	20	20
Time step - Δ	10sec	1.0sec
Planning Horizon - L_h	50m	50m
Path Budget - L_T	1000m	1000m
Number of Actions - $N_{children}$	5	N/A
Number of Goals - N_{goals}	100	N/A

The number of frequency components used, K, was selected arbitrarily. In both cases the vehicle was given a speed of 10cm/s. This parameter was chosen because it is the top speed of the Resource Prospector robot.

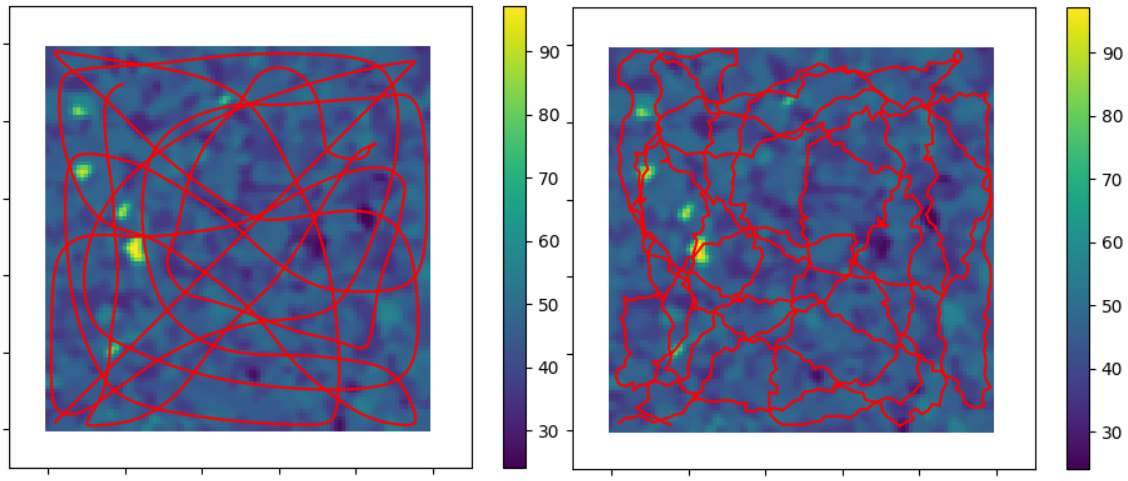
D.6.2 Results

We can see in Fig. D.4 and Fig. D.5 example trajectories produced by the ergodic planners. The greedy ergodic planner produces much smoother paths than the approximately optimal planner. This is partially due to the smaller time step for the greedy planner, and because the approximate planner randomly samples headings for each node, causing discontinuities in the path. The roughness of the approximately optimal planner's paths is a reflection of the fact that the algorithm is a coarse approximation of Miller's optimal path planner.



(a) Water density map 09.

(b) A trajectory produced by the SPRT $(s = 8)$ algorithm. On average ≈ 53 maxima were observed.

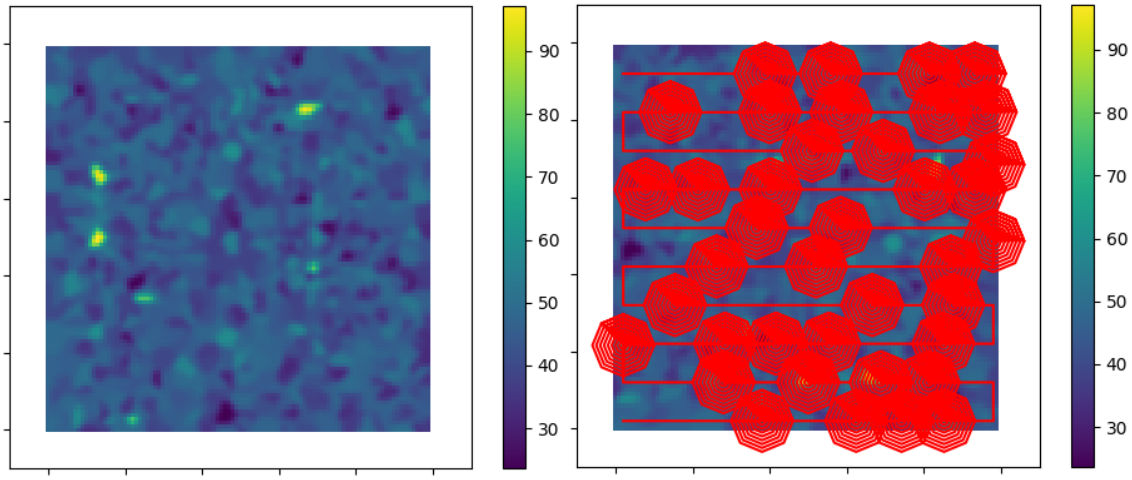


(c) The trajectory produced by the greedy ergodic planner.

(d) The trajectory produced by the approximately optimal ergodic planner. The jagged path is due to the approximations of the search algorithm described in Algorithm D.2

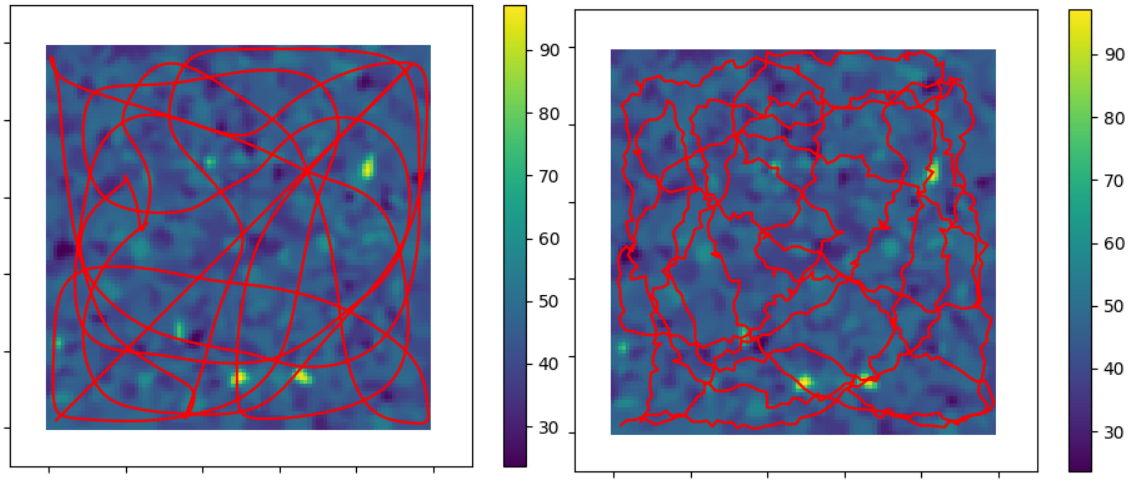
Figure D.4: Trajectories produced by the different planning algorithms for the water map where the optimal ergodic planning algorithm achieved its highest score, which was map 9. We can see that the ergodic planners either get near to or cross all of the major hot spots in the map.

Fig. D.4 shows the generated trajectories for the water map, map 09, where the approximately optimal ergodic planner had its best score. Fig. D.5 shows the trajectories generated by the algorithms for the water map, map 5, where the approximately optimal ergodic planner had its lowest score.



(a) Water density map 05.

(b) Trajectory produced by SPRT algorithm on water map 5. On average ≈ 53 maxima were observed.



(c) Greedy ergodic planner on water map 5.

(d) Approximately optimal ergodic planner on water map 5. The generated trajectory crosses all the major hotspots in the map.

Figure D.5: The trajectories of the different planning algorithms for the water map where the optimal ergodic planning algorithm had its worst performance, which was map 5. Even though this is the worst performing map, the approximately optimal ergodic planner crosses all of the bright yellow regions of the map, which are the major hotspots.

The number of maxima observed by the different algorithms is given in Table D.33. These data show that the ergodic planners don't observe as many maxima as the SPRT algorithm, but they do observe more maxima than the lawnmower path alone. This is the point of intelligent planners, that they make more efficient use of resources than uninformed planners, so this result is expected.

Table D.33: The number of peaks in the water maps that were observed by the different algorithms.

Map	No AIM	Optimal Ergodic	Greedy Ergodic	SPRT ($s = 8$)
00	10	30	34	42.9 ± 3.1
01	11	35	40	44.1 ± 2.5
02	20	31	31	53.5 ± 4.8
03	13	37	42	46.3 ± 3.3
04	15	35	39	54.1 ± 2.5
05	12	25	34	53.1 ± 3.0
06	19	35	30	56.9 ± 1.9
07	16	29	36	49.9 ± 3.0
08	12	37	42	48.3 ± 2.5
09	17	30	41	53.0 ± 2.9
10	12	28	37	51.0 ± 3.6
11	15	29	28	48.7 ± 2.0
12	18	37	46	53.0 ± 2.7
13	19	32	34	52.0 ± 1.9
14	24	25	37	56.9 ± 2.7
15	17	33	33	53.4 ± 4.5
16	19	35	55	59.9 ± 3.4
17	16	31	33	54.4 ± 2.7
18	18	30	36	50.3 ± 3.1
19	10	41	36	46.9 ± 3.2
20	22	32	40	56.4 ± 2.5
21	20	34	38	51.6 ± 2.3
22	19	35	39	63.5 ± 2.5
23	16	26	35	51.5 ± 2.4
24	18	32	44	56.3 ± 3.8
25	13	30	30	42.0 ± 1.4
26	14	34	43	49.4 ± 3.5
27	19	37	43	53.6 ± 3.2
28	22	38	36	55.8 ± 3.5
29	12	26	42	52.8 ± 2.7

In Fig. D.6 we report the number of maxima obtained by the different algorithms, averaged over all maps. We do see that the SPRT algorithm observes statistically significantly more maxima in the maps than the ergodic planners at a 95% confidence level.

The size of the effect of using the SPRT algorithm is $d = 3.21$ and $d = 2.33$, using Cohen's d, relative to the approximately optimal and greedy planners, respectively. Both these values are very large. The performance difference between the approximately optimal and greedy planners was not statistically significant at a confidence level of 95%.

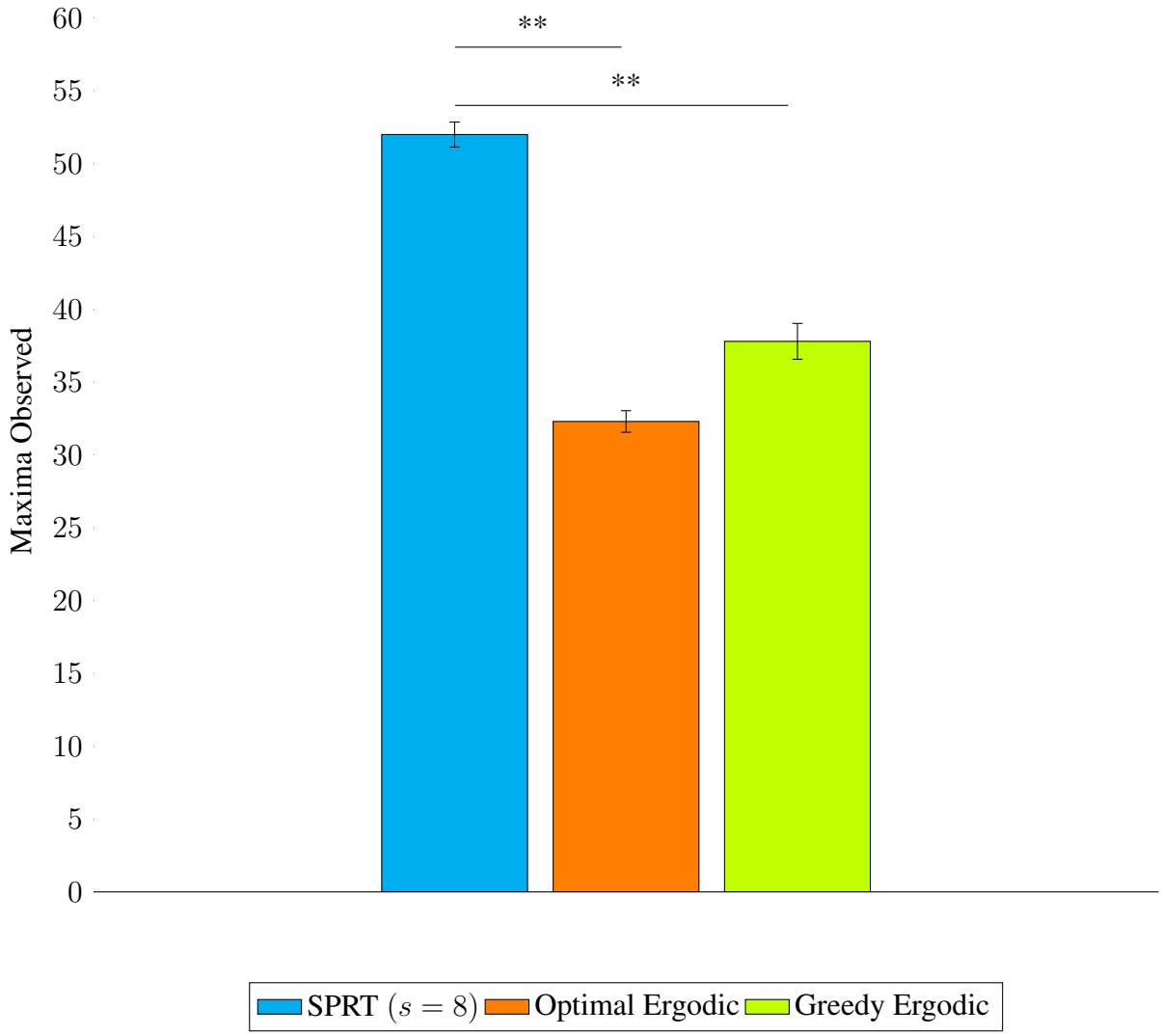


Figure D.6: The SPRT ($s = 8$) algorithm captures more maxima in the water maps than either ergodic planner with statistical significance. The performance of the greedy and approximately optimal ergodic planners are not statistically significant at a confidence level of 95%. Error bars represent one standard error. The effect size of the difference between SPRT algorithm and the approximately optimal ergodic algorithm was $d = 3.21$. The effect size of the difference between the SPRT algorithm and the greedy ergodic planner was $d = 2.33$.

D.6.3 Discussion

Saying whether or not ergodic planning, or indeed any other path planner, is the right planner to use for a given mission is beyond the scope of this thesis. However, having conducted this experiment, we do make some observations.

First, it is important to stress that our approximately optimal ergodic planner is not guaranteed to be optimal. It is a coarse approximation of the algorithm used by Miller and Murphy, used to explore the utility of ergodic planning. The performance of the optimal ergodic planner should be at least maintained, and almost certainly improved, by using a direct implementation of the

planner of Miller and Murphy.

Second, the AIMs that the SPRT algorithm conducted to collect more information represent additional path length that the ergodic planner was not assigned. From the results above we can conclude that the ergodic planner makes more *efficient* use of its path budget than the SPRT and No AIMs algorithms. Again, this is to be expected with a planner that is aware of and makes use of information about the environment. That being said, there is no reason that something like the ergodic planner couldn't be complimented by the SPRT-based change detection algorithm presented in Chapter 4. Indeed, that is one of the benefits of using a planner agnostic algorithm.

Third, the ergodic planner is good for coverage planning. In this experiment we used density maps that had more complex structure than those used in either (Miller et al., 2016) or (Mathew and Mezić, 2011). The results are in-keeping with the claims made in those works, showing that ergodic planning is useful in complex environments.

However, when there is a burden of validating the system, the benefit of a planner must warrant the costs of the required validation and verification process. There are scenarios where complex planners are warranted, but there may also be operational conditions where simply following simple, pre-determined coverage paths is the overall better option.

A fourth consideration is that in the absence of prior knowledge the planner must take steps to learn what the map looks like. At this point all unmapped regions are equally valuable, so there is a question of whether or not an information gathering planner might be more appropriate, or even simple approaches like lawnmower paths or frontier exploration. Once that initial knowledge has been acquired, however, the ergodic planner can again become useful.

In the particular case of the scenario presented in Chapter 4, the ergodic planner is especially well-suited because it doesn't need to choose to deploy discrete actions like AIMs, which the SPRT algorithm does. Because the ergodic planner attempts to spend more of its time in regions where the density map has higher values it accomplishes the purpose of the AIM deployment without needing to reason about the deployment of AIMs.

Removing the notion of AIMs from the robot's suite of actions simplifies the planning process. This is largely possible because, even though the AIMs are discrete actions the robot can take, they are executed by driving a path, and the benefits of a particular AIM can be folded into a path cost function.

A reduction in complexity is always welcome in flight missions. But the ergodic trajectory cost function doesn't address situations where robots must take distinct actions that cannot be executed by following a path, such as drilling for soil samples. In this case robots will need some representation of discrete actions and will need to reason about the value and risks of taking those actions. Discrete actions can be encoded in cost functions, and ergodicity may be a part of that cost function, but ergodicity by itself does not address this problem.

Finally, there is the matter of the shape of the generated paths. The trajectories produced by the unconstrained ergodic planner involve a considerable amount of doubling back, which, in the author's experience, seems unlikely to be palatable to mission operators. Backtracking is especially concerning when planned operations that cover long distances.

The concept of operations in the Resource Prospector mission involves travelling long distances, on the order of kilometers, only stopping to gather more localized, dense mapping data when sensor readings draw the attention of the remote science team. In this case, something like the ergodic planner is not a viable tool for mission-level trajectory generation, as the backtracking

is antithetical to the mission operations.

It is possible that the planner could be further constrained to reduce the amount of backtracking generated by the planner to increase the generated paths' alignment with mission operations. However, at this point, one would have to ask whether or not the utility of an ergodic planner is being overpowered by the mission constraints.

That is not to say that an ergodic planner, or any other planner, has no place in such a mission. In Chapter 4 we designed AIMs as blindly driven spiral paths. It seems that, for example, an ergodic planner would be an excellent tool to use to design AIMs, especially as they can incorporate precursor data. As we have seen, the ergodic planner produces better results than the lawnmower path, and lawnmower and spiral paths are typically, in the author's experience, used to conduct Area of Interest Manoeuvres.

To summarize, when one has prior knowledge of the world, informed planners can make much better use of resources than uninformed planners. Further, the ergodic planner is particularly suited to the problem setting discussed in Chapter 4. Even then, the ergodic planner could be augmented by the change detection algorithms we had discussed there.

However, the ergodic cost function alone is not panacea in space exploration missions. A planner which seeks to minimize the ergodic cost of a trajectory may need to be augmented to deal with more complex decision spaces and mission constraints. While a promising tool for planning trajectories, it will still need to be worked into mission operations on a case-by-case basis.

Appendix E

Falsification Sampling Supplemental Material

E.1 Belief Distributions

This section gives the belief distributions learned by the algorithms for the different settings of error and sampling budget. Most important to note across all settings of hypotheses quality and sampling budget is that once the probability of having a sensor error is greater than 0.2 then neither algorithm is capable of reliably identifying the best hypothesis.

E.1.1 Experiment 1 - All Good Hypotheses

Fig. E.1to Fig. E.10 give the final belief distributions of the two algorithms. We can see that as the noise level increases, the ability to distinguish the best hypotheses decreases.

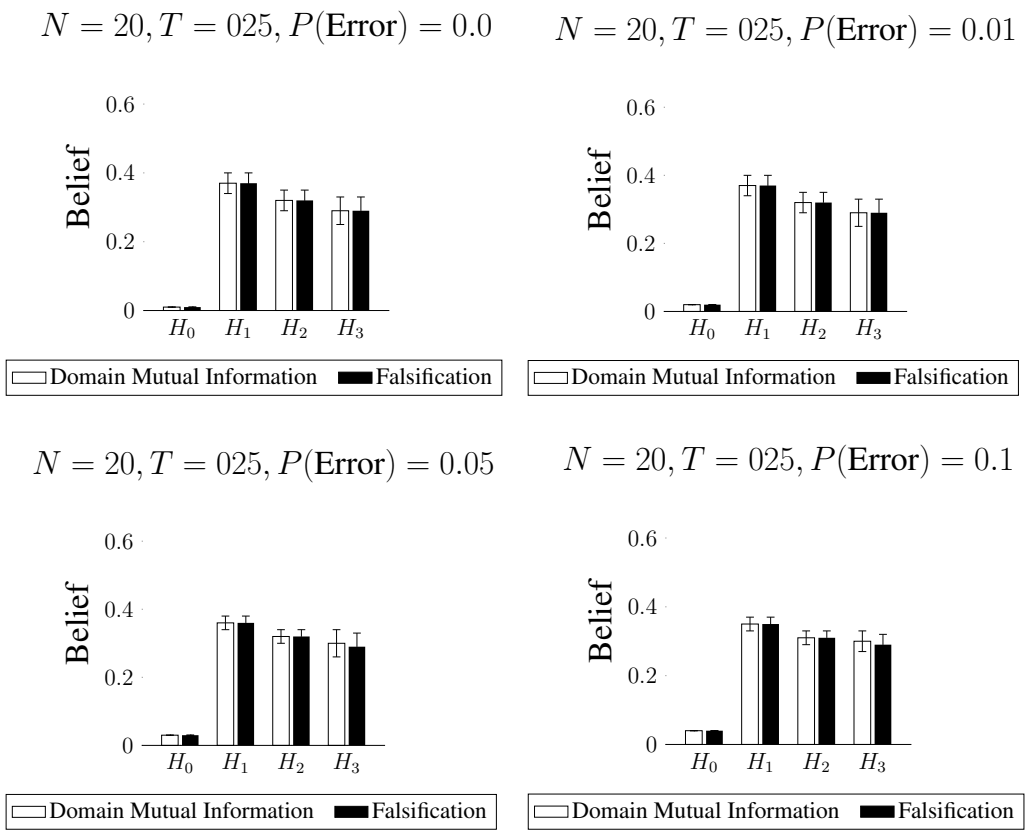


Figure E.1: All good hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 25, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

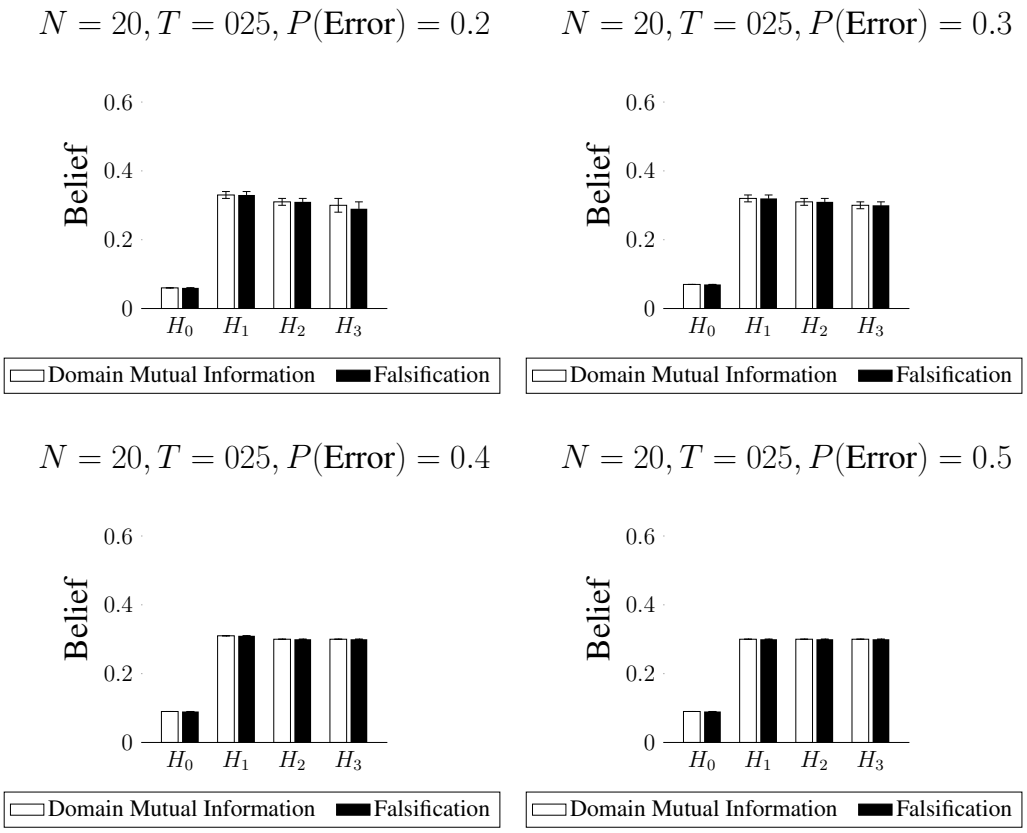


Figure E.2: All good hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 25, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

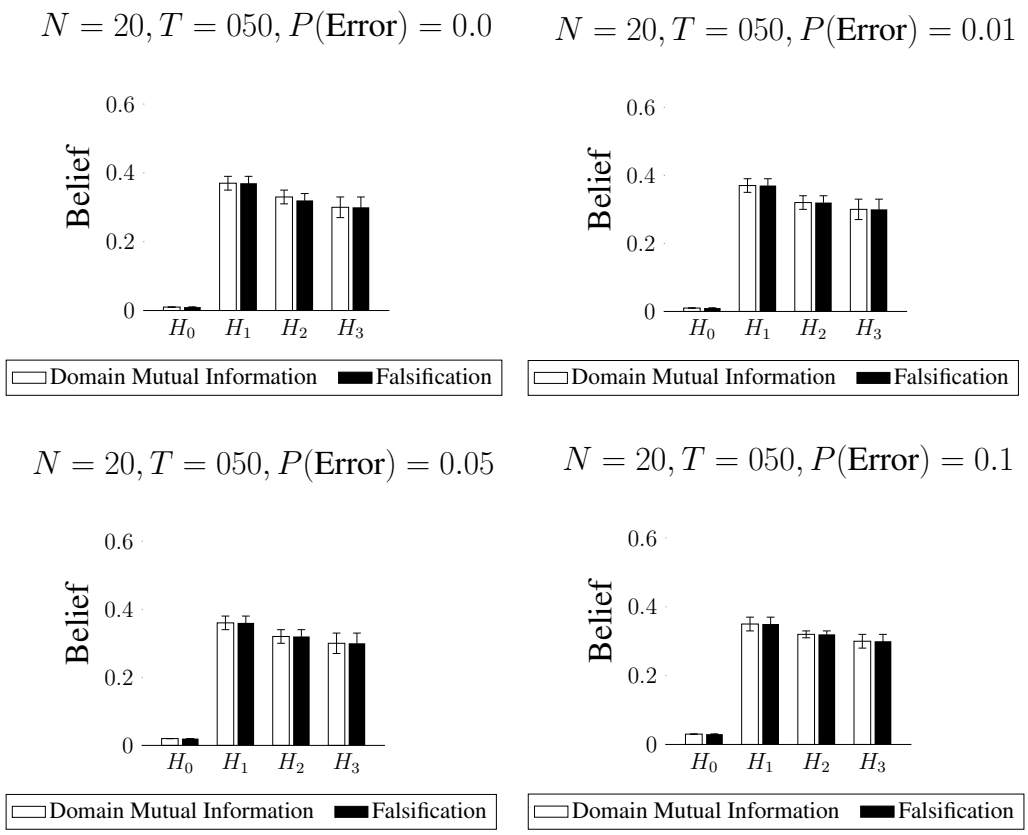


Figure E.3: All good hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 50, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

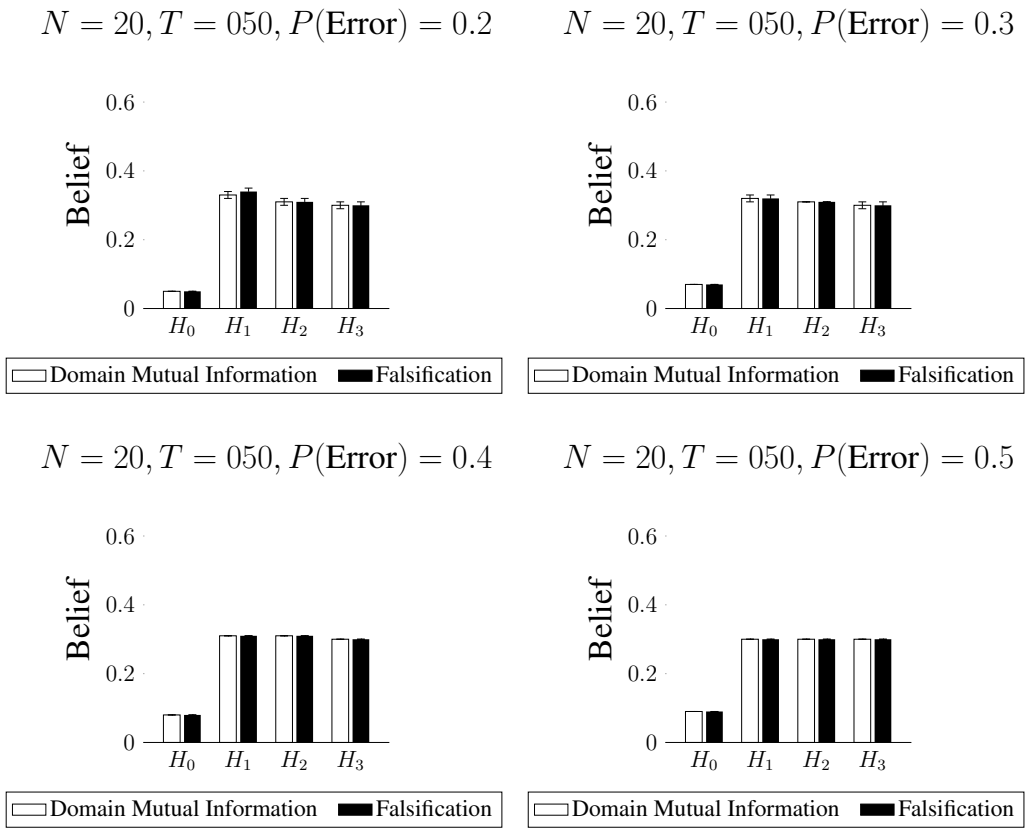


Figure E.4: All good hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 50, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

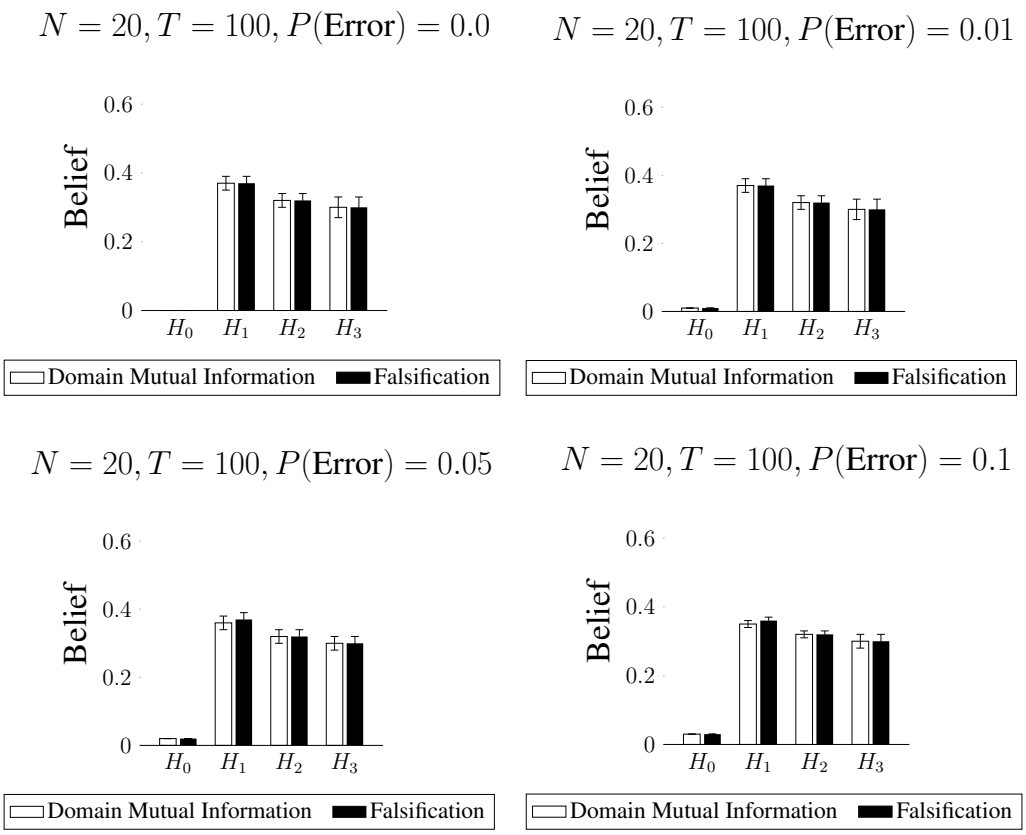


Figure E.5: All good hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 100, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

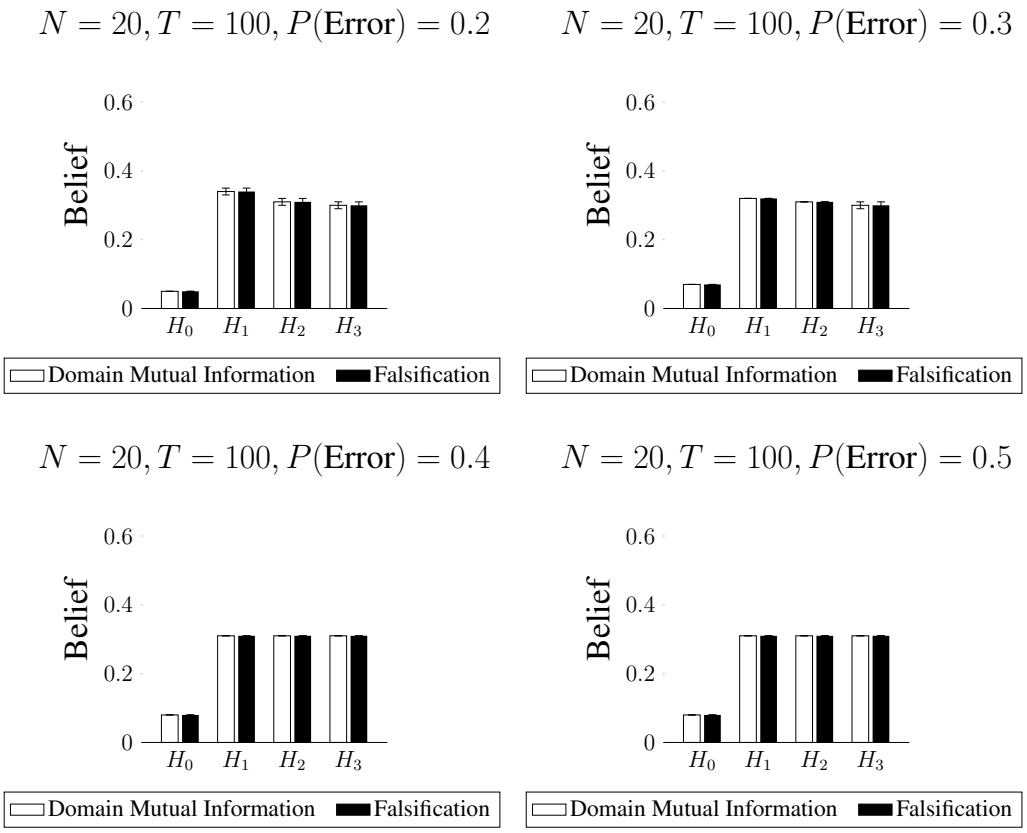


Figure E.6: All good hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 100, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

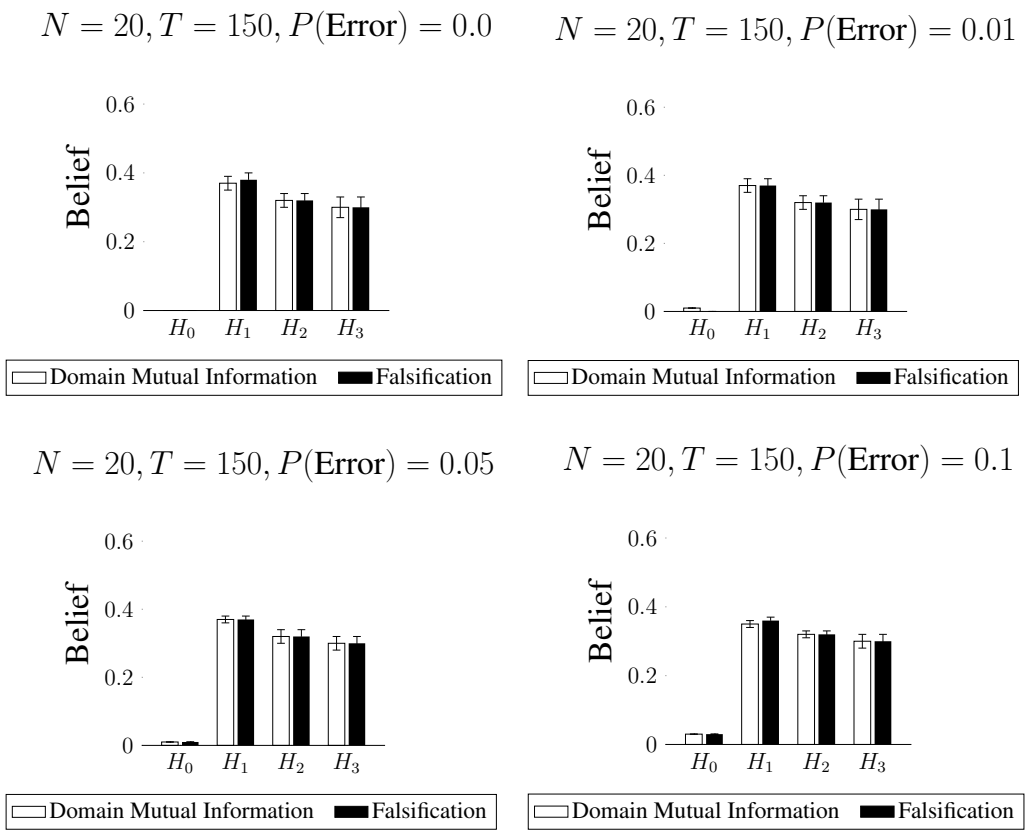


Figure E.7: All good hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 150, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

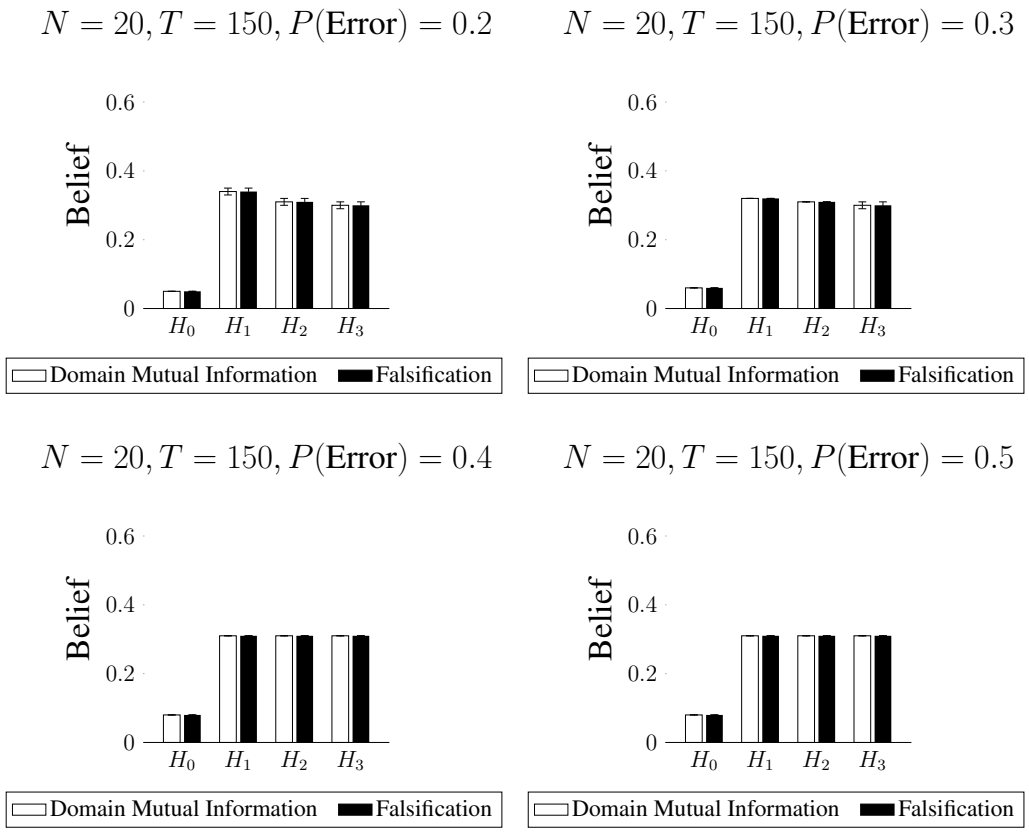


Figure E.8: All good hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 150, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

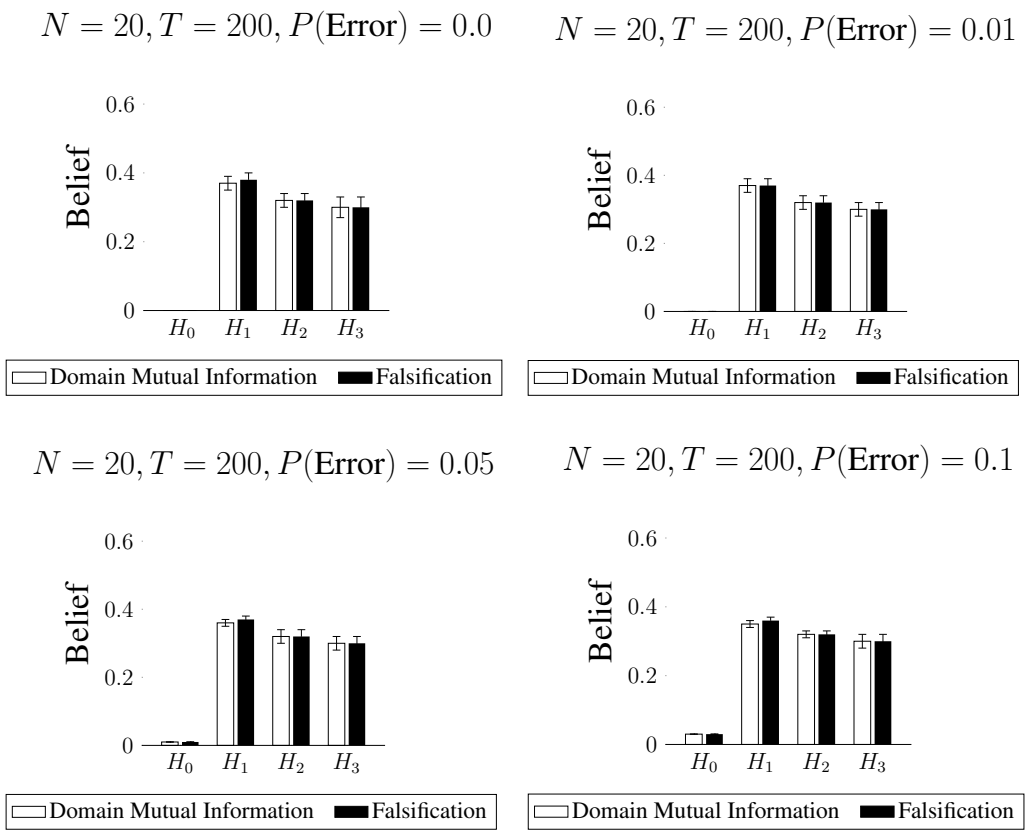


Figure E.9: All good hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 200, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

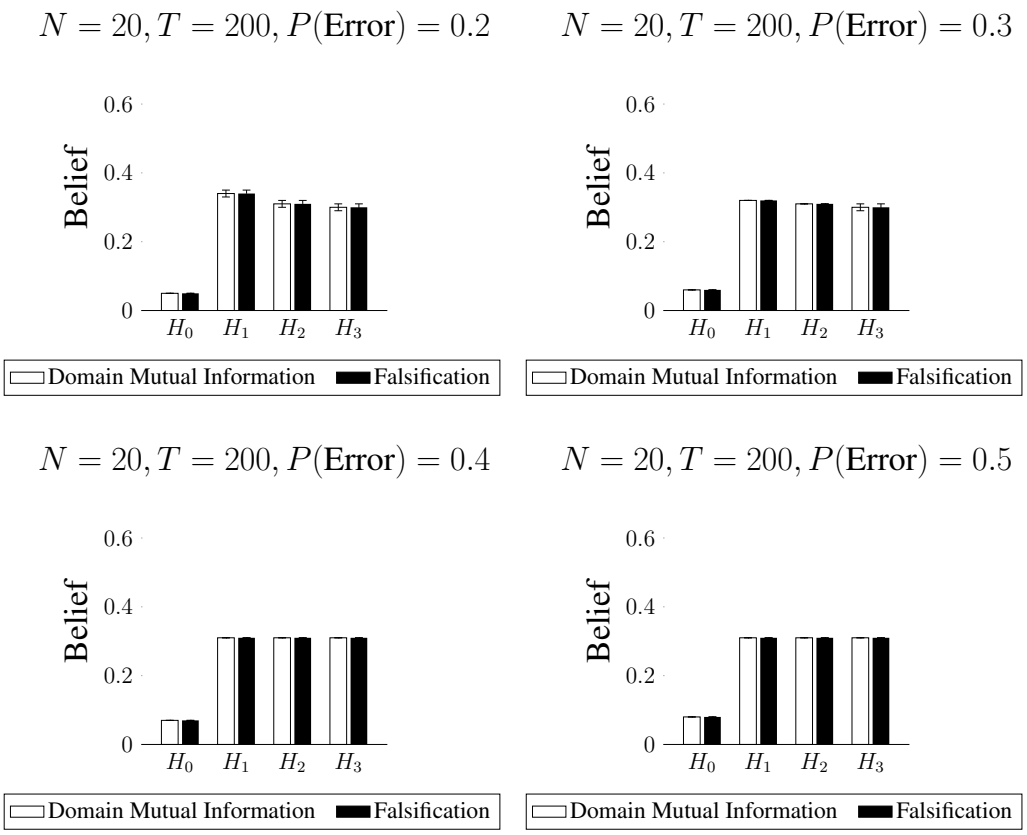


Figure E.10: All good hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 200, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

E.1.2 Experiment 2 - Mixed Quality Hypotheses

Fig. E.11to Fig. E.20 give the final belief distributions of the two algorithms. We can see that as the noise level increases, the ability to distinguish the best hypotheses decreases.

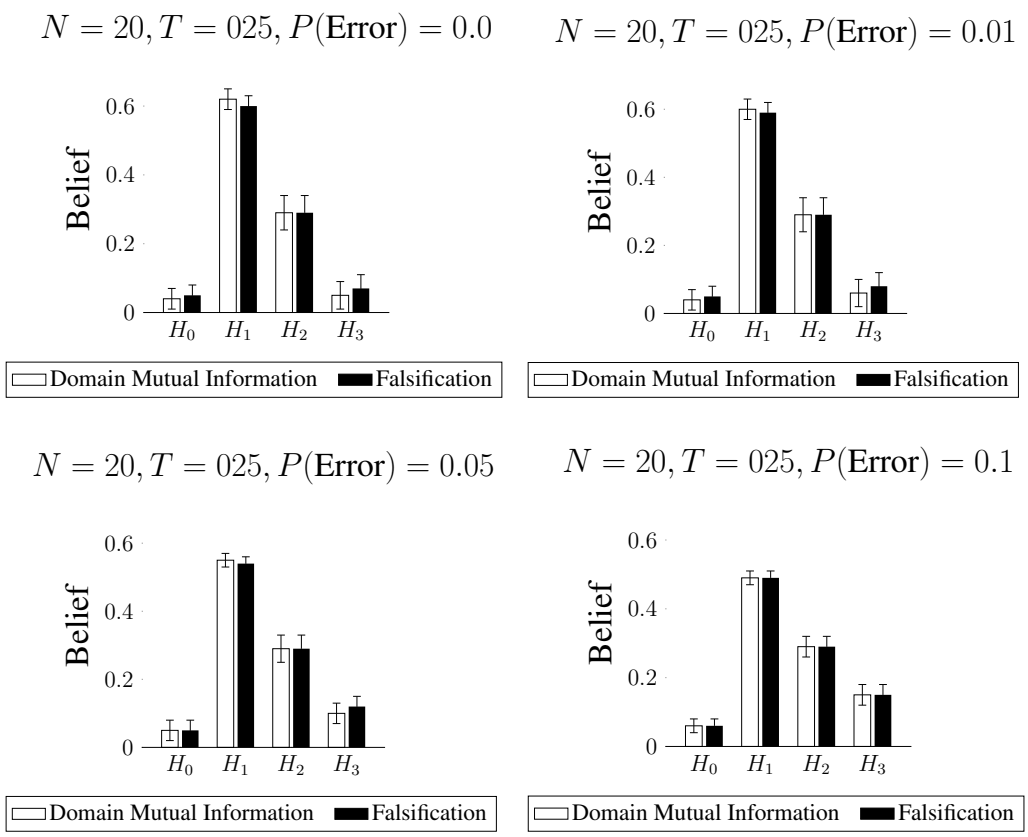


Figure E.11: Mixed quality hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 25, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

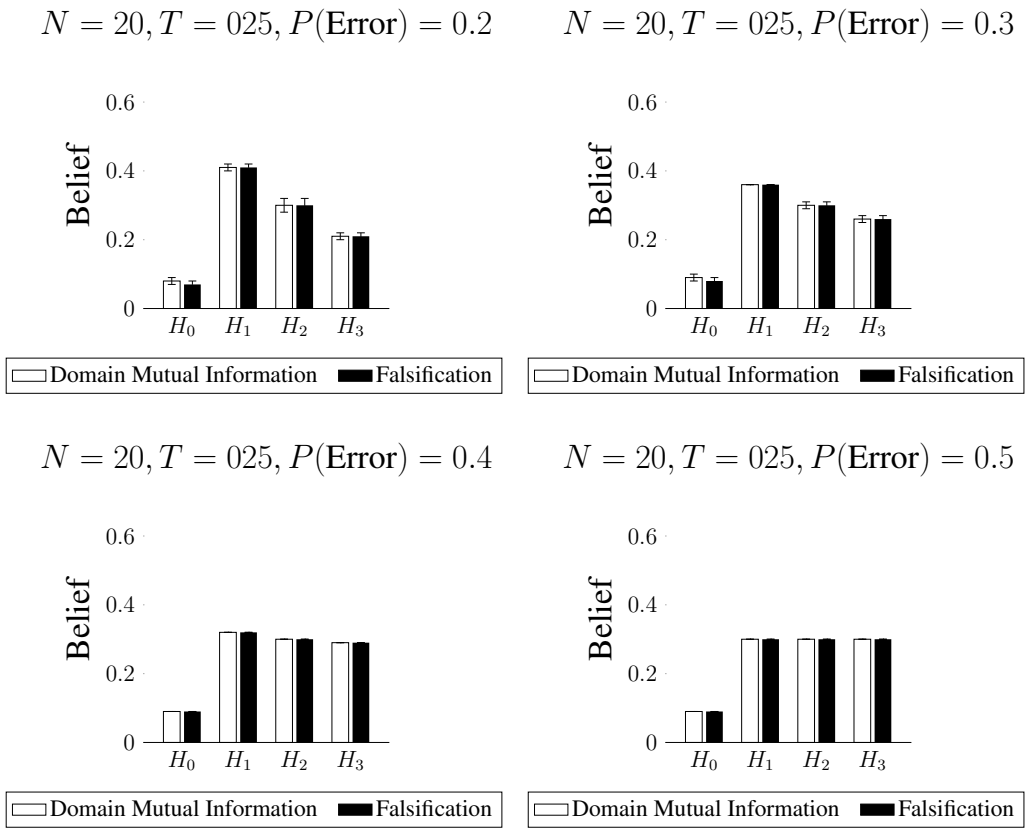


Figure E.12: Mixed quality hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 25, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

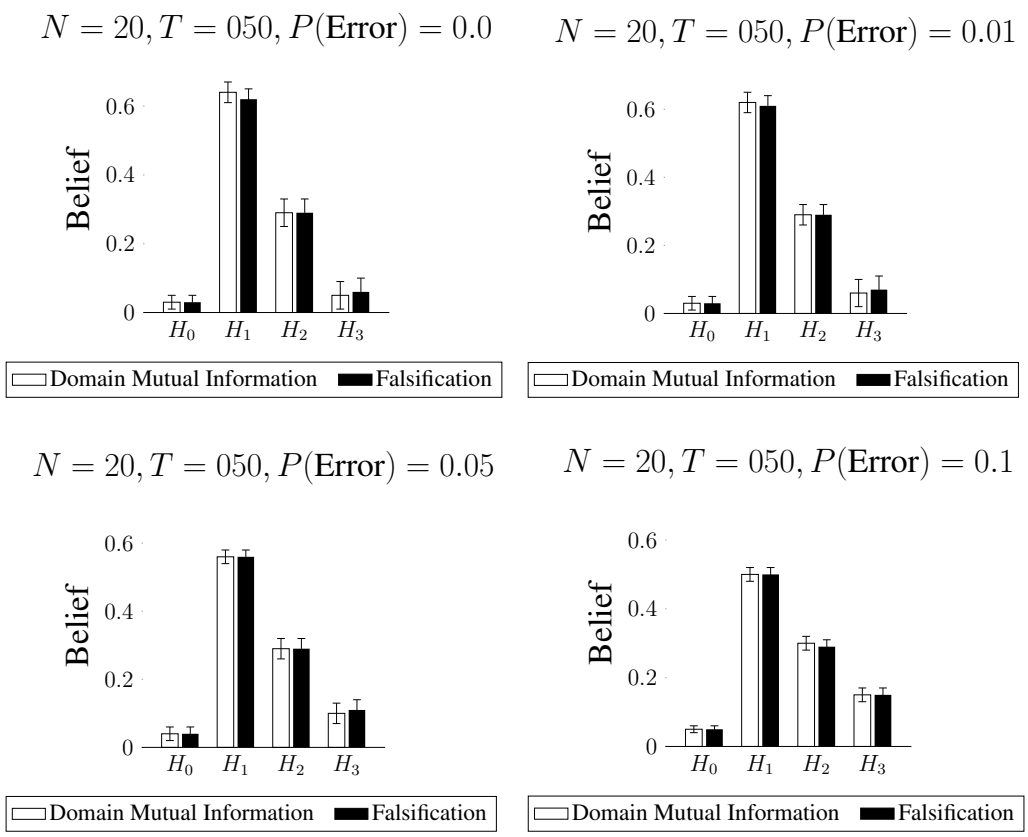


Figure E.13: Mixed quality hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 50, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

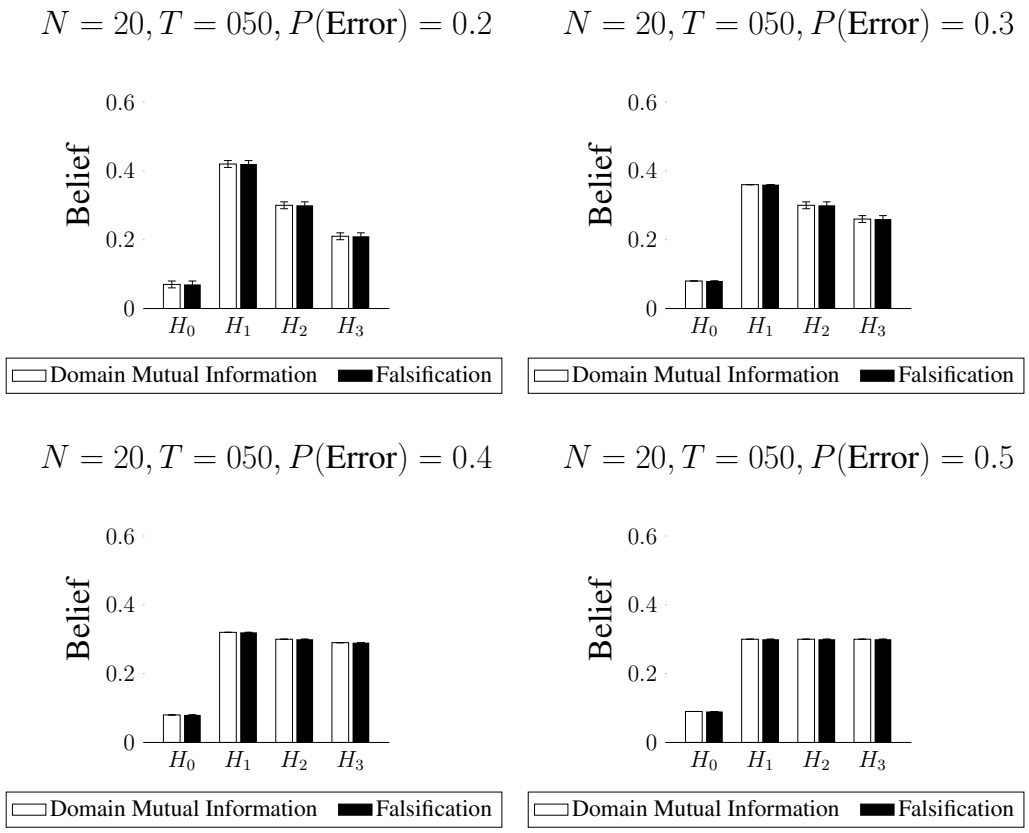


Figure E.14: Mixed quality hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 50, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

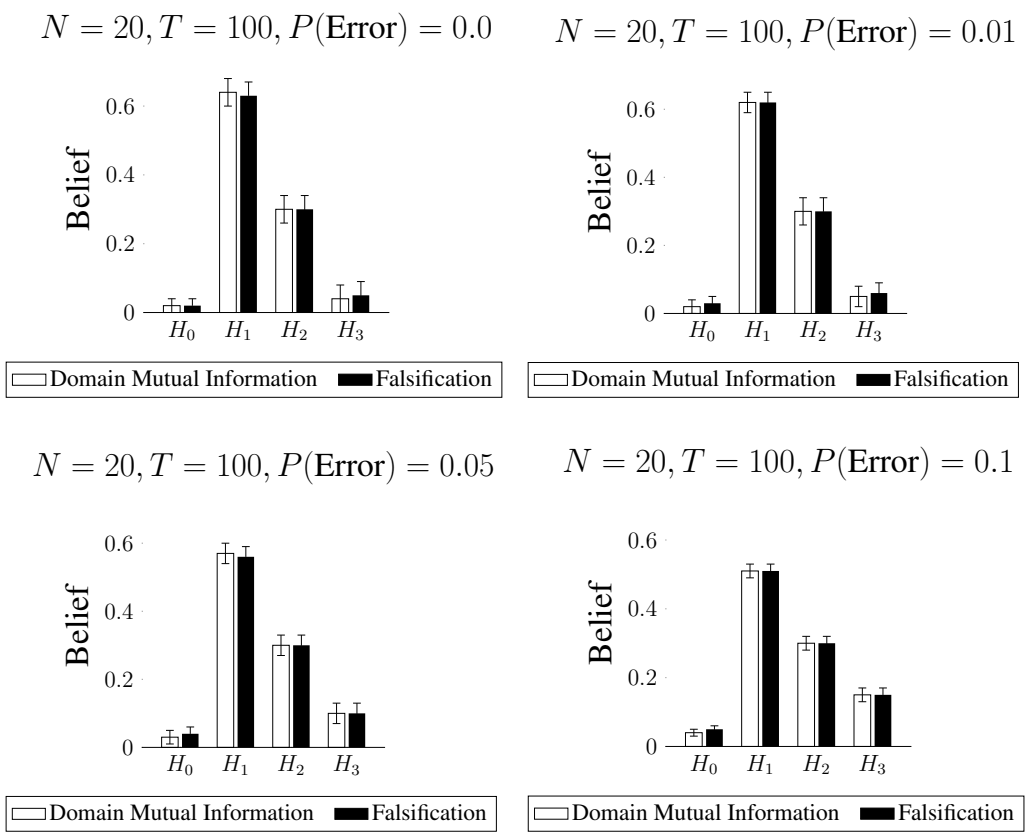


Figure E.15: Mixed quality hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 100, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

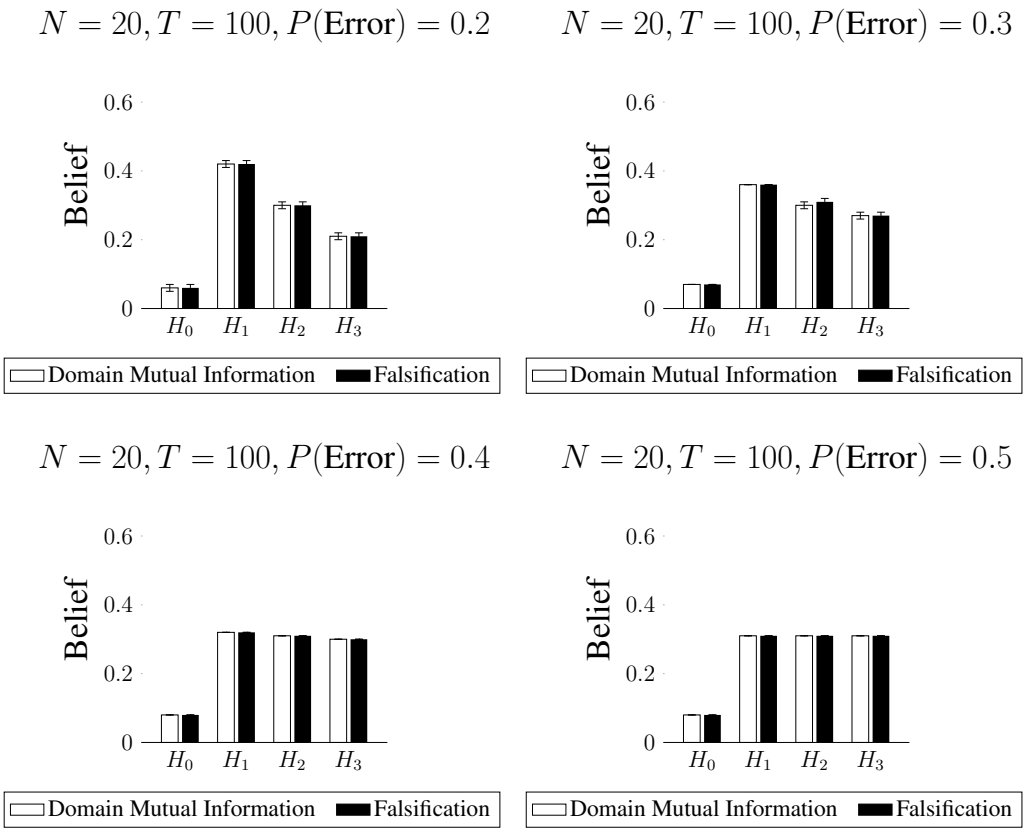


Figure E.16: Mixed quality hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 100, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

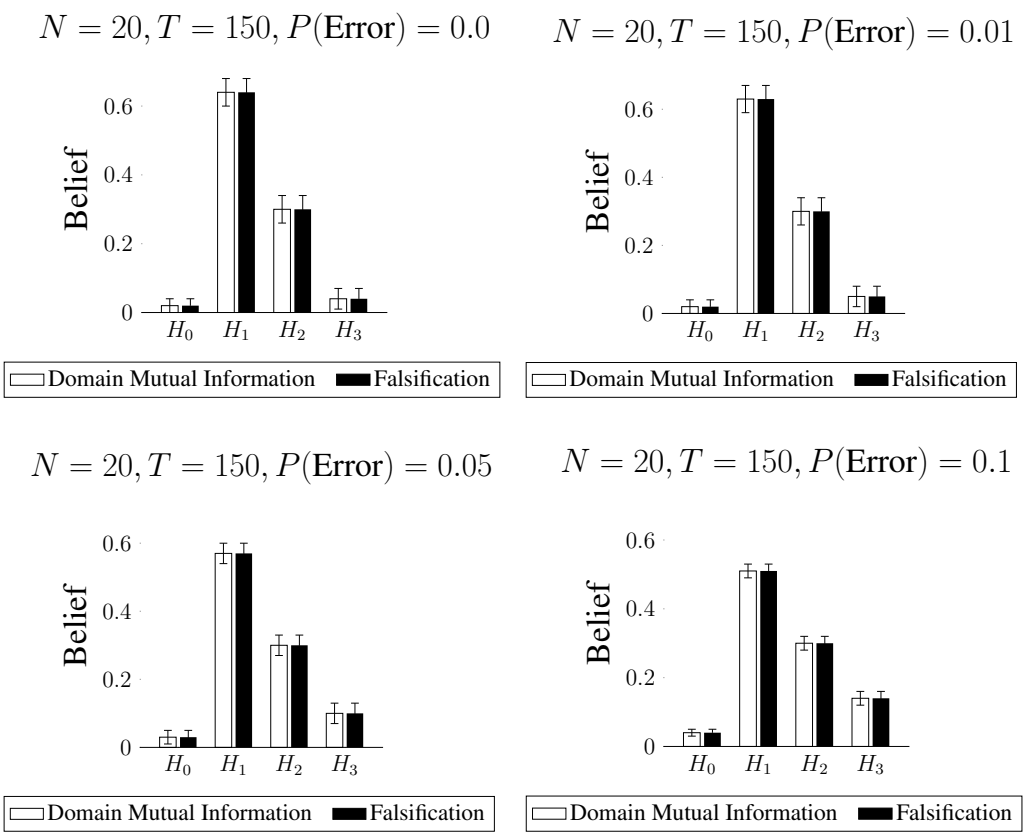


Figure E.17: Mixed quality hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 150, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

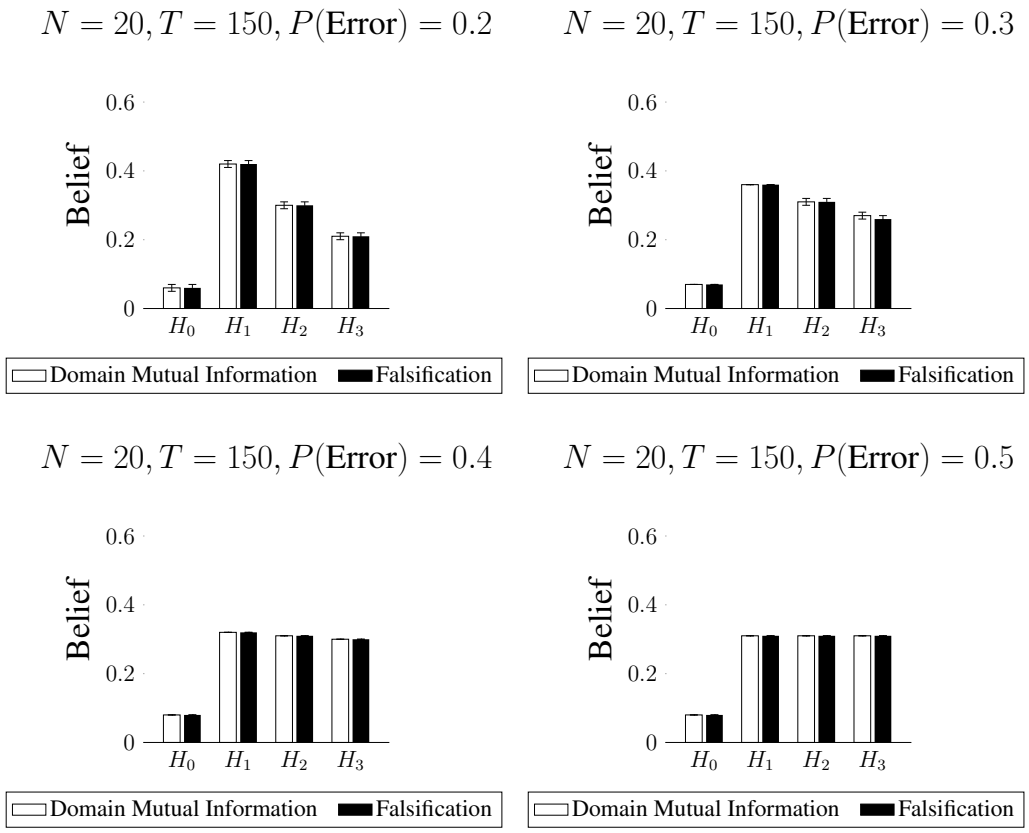


Figure E.18: Mixed quality hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 150, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

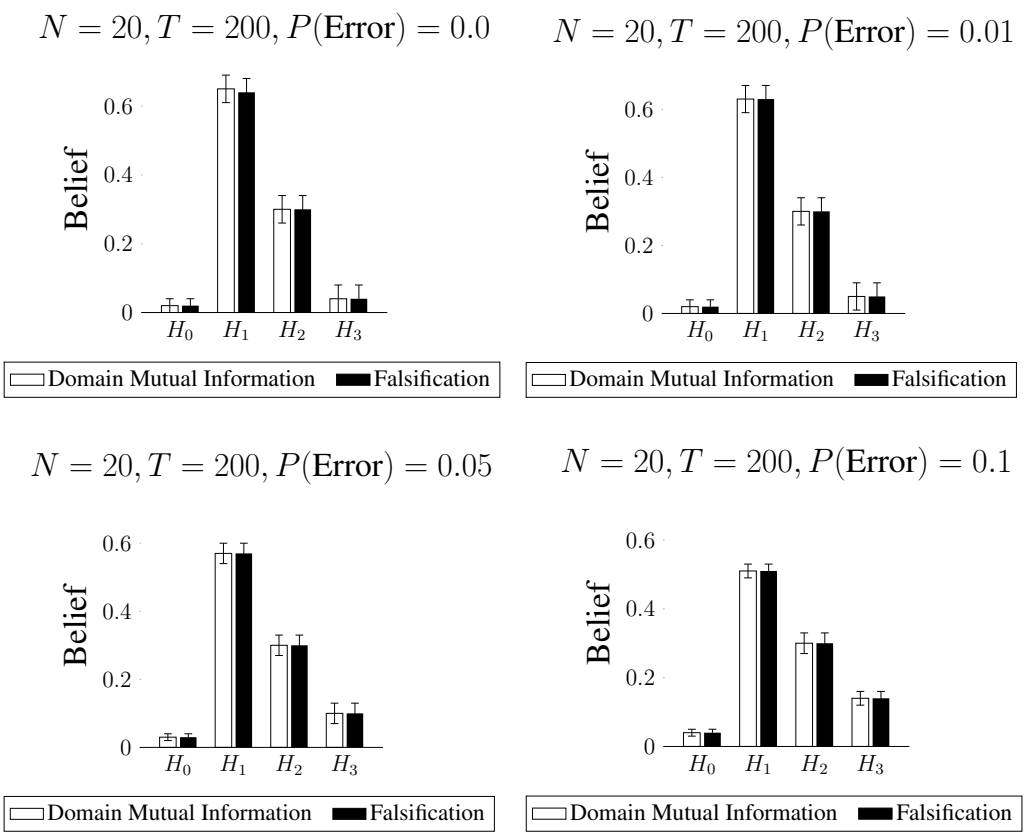


Figure E.19: Mixed quality hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 200, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

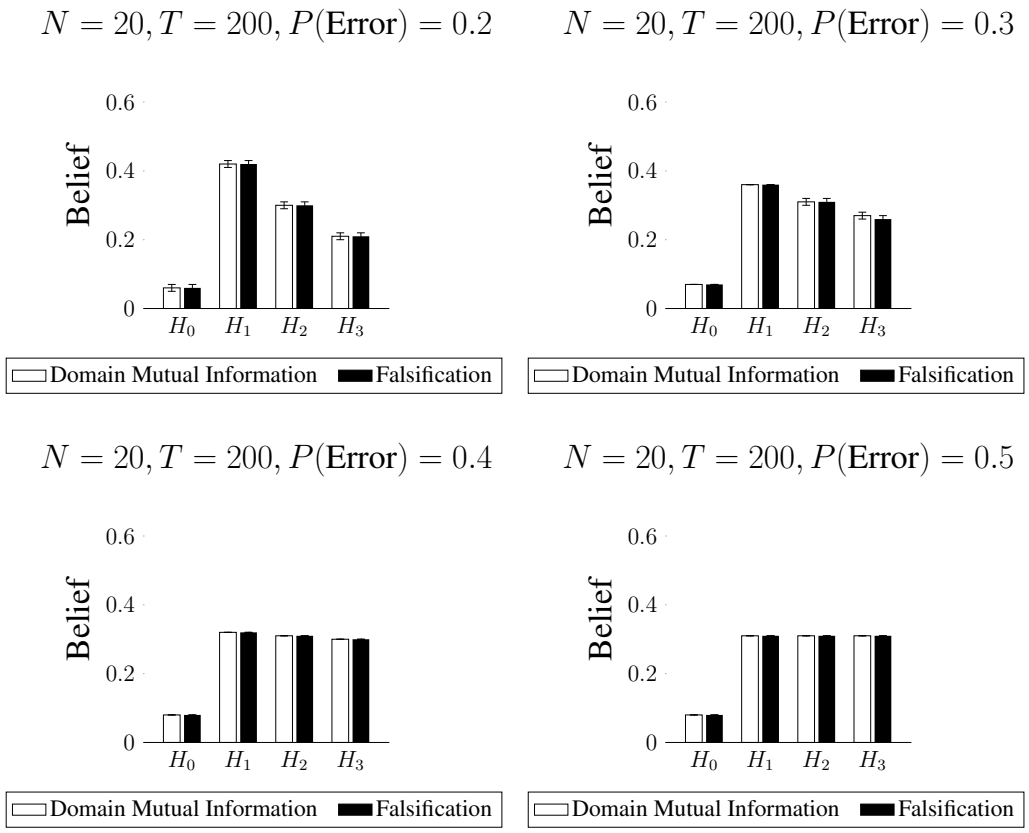


Figure E.20: Mixed quality hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 200, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

E.1.3 Experiment 3 - All Bad Hypotheses

Fig. E.21to Fig. E.30 give the final belief distributions of the two algorithms. We can see that as the noise level increases, the ability to distinguish the best hypotheses decreases.

When the noise level hits an error rate of 0.2, both algorithms begin to select one of the incorrect hypotheses as the best hypothesis. However, in this case both algorithms select the least-bad of the proposed hypothesis, H_1.

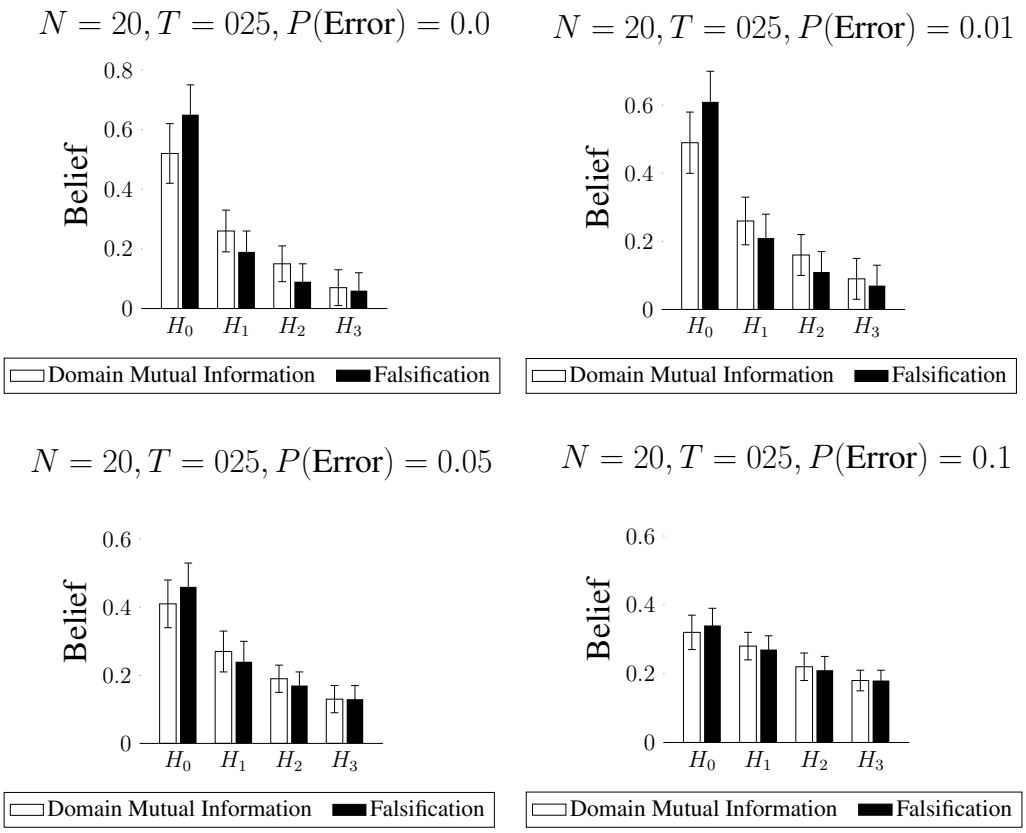


Figure E.21: All poor hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 25$, $P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

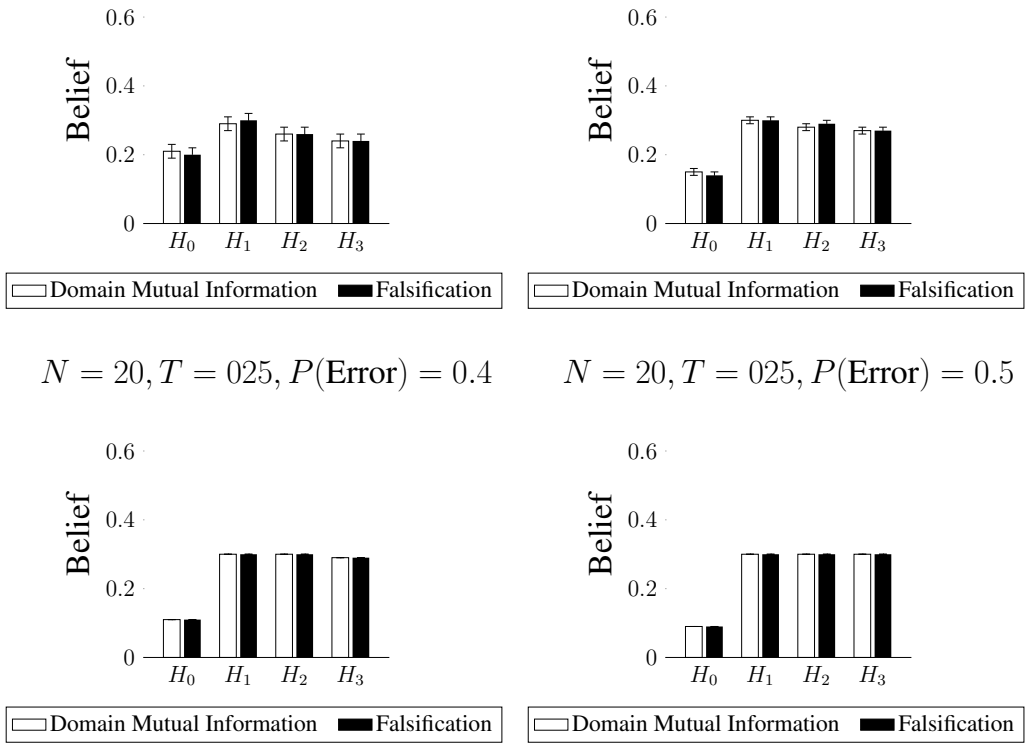


Figure E.22: All poor hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 25, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

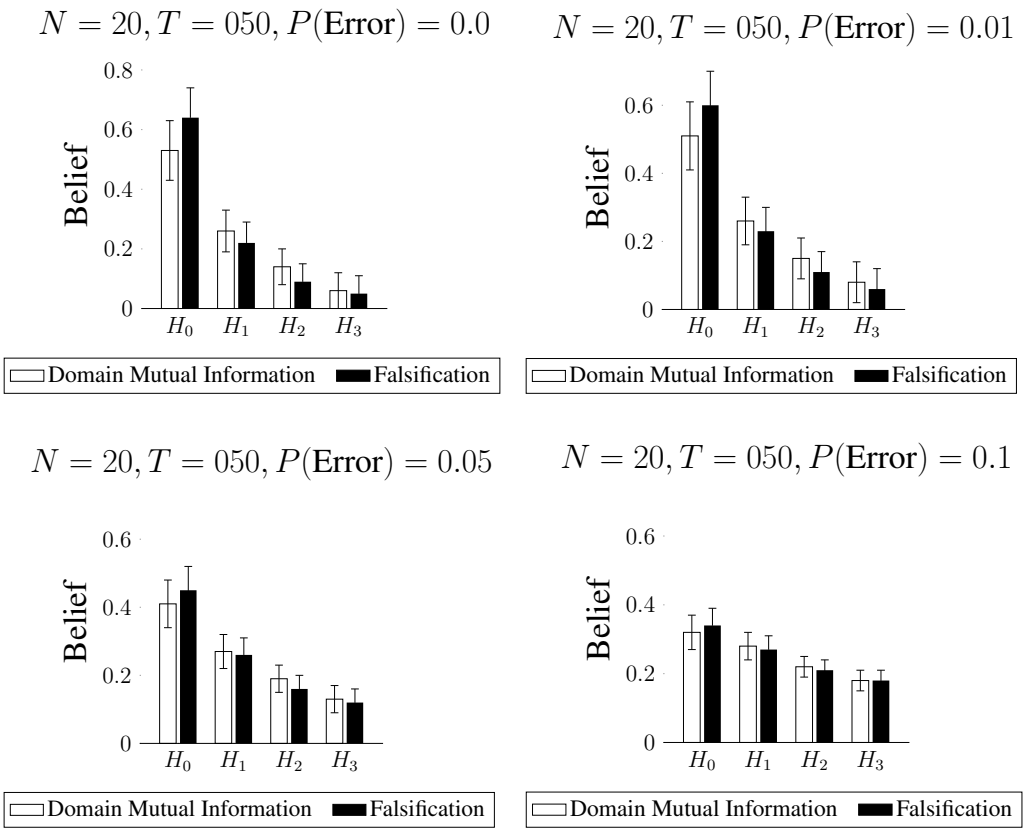


Figure E.23: All poor hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 50, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

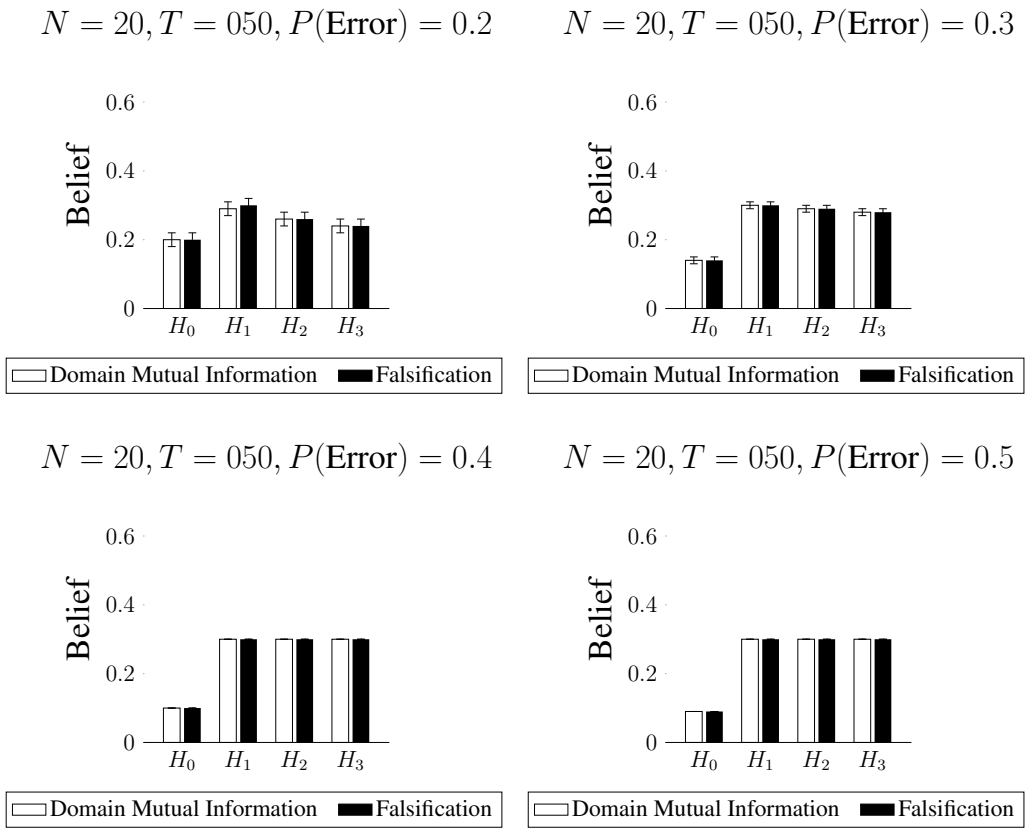


Figure E.24: All poor hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 50, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

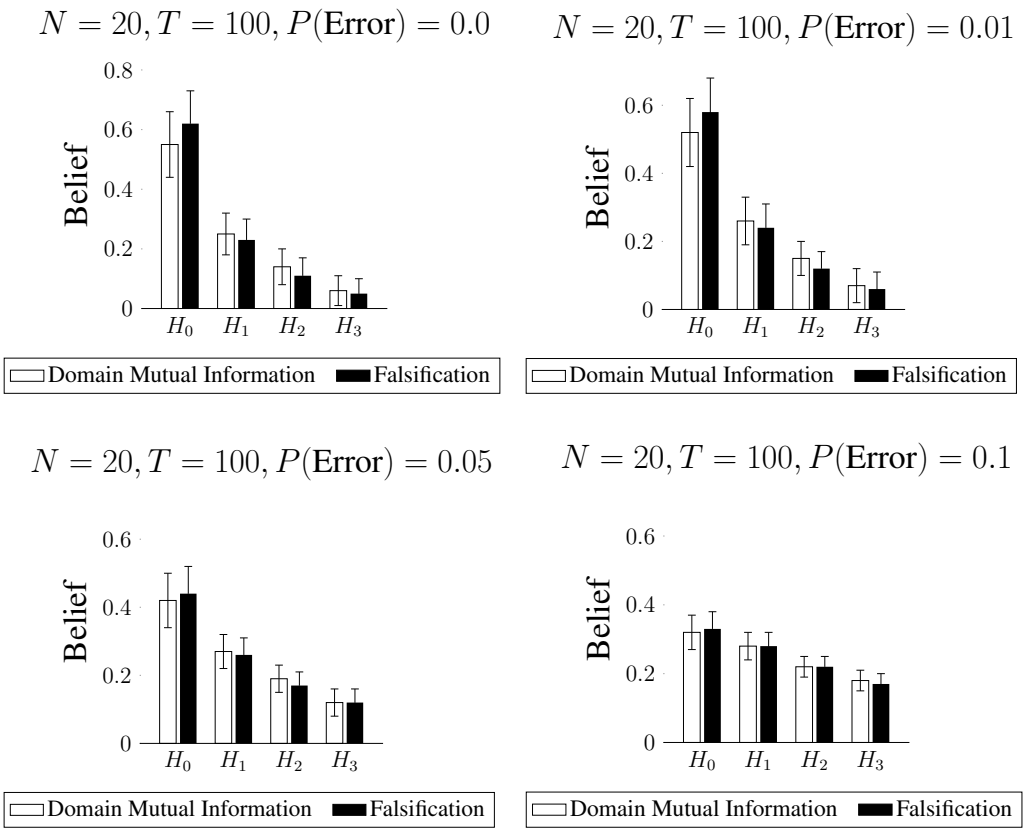


Figure E.25: All poor hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 100, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

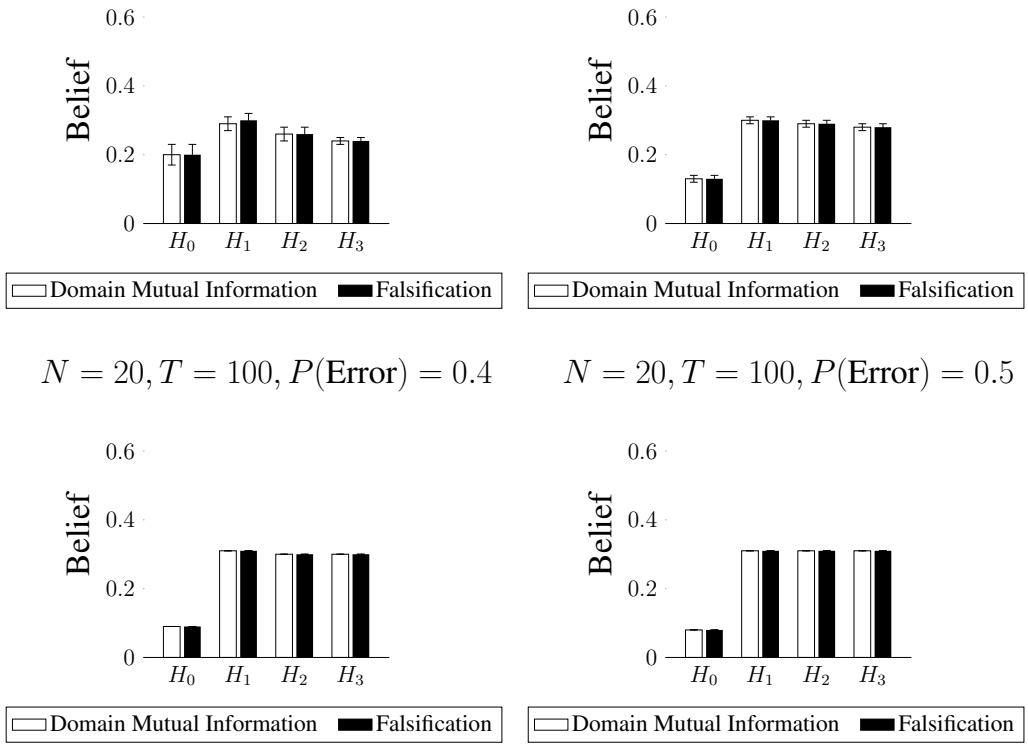


Figure E.26: All poor hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 100, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

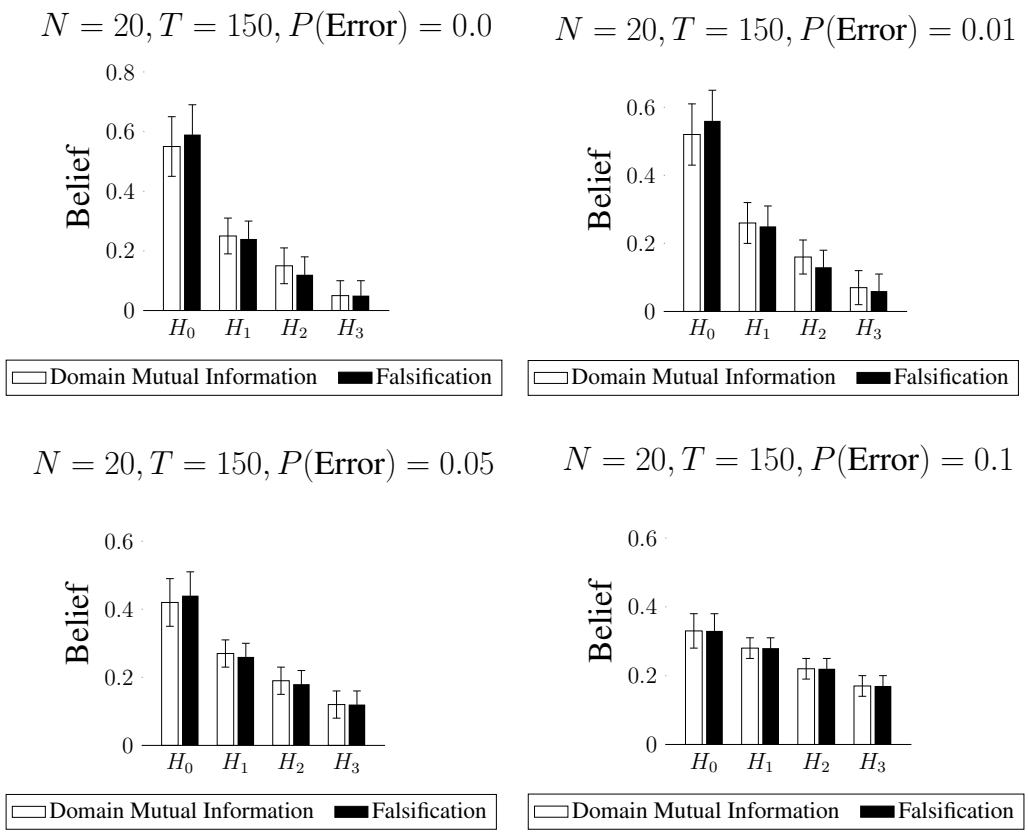


Figure E.27: All poor hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 150, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

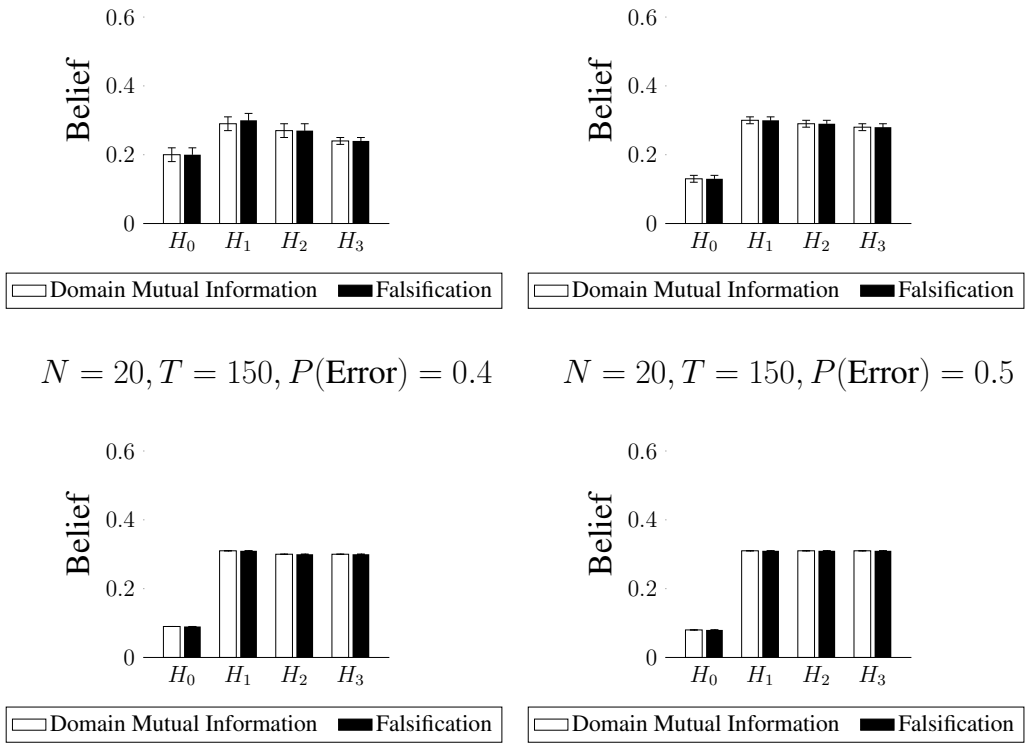


Figure E.28: All poor hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 150, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

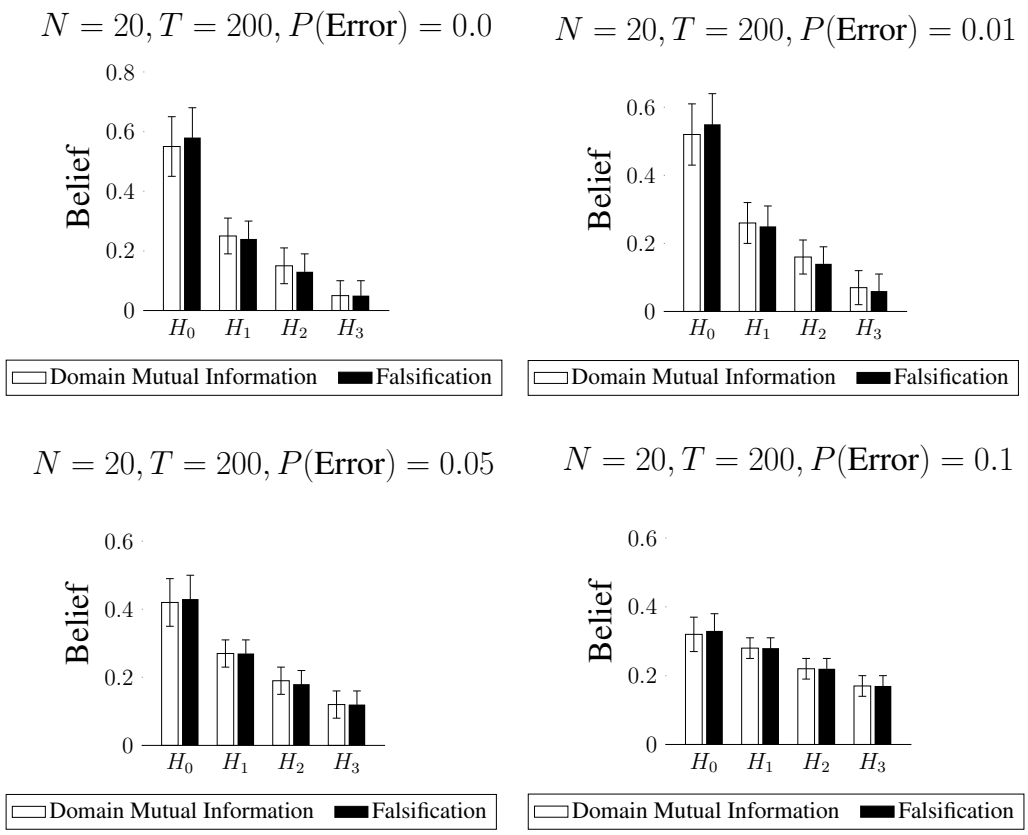


Figure E.29: All poor hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 200, P(\text{Error}) \in \{0, 0.01, 0.05, 0.1\}$. Both algorithm select the correct hypotheses. Initially the performance of the algorithms does not vary much as noise as increased.

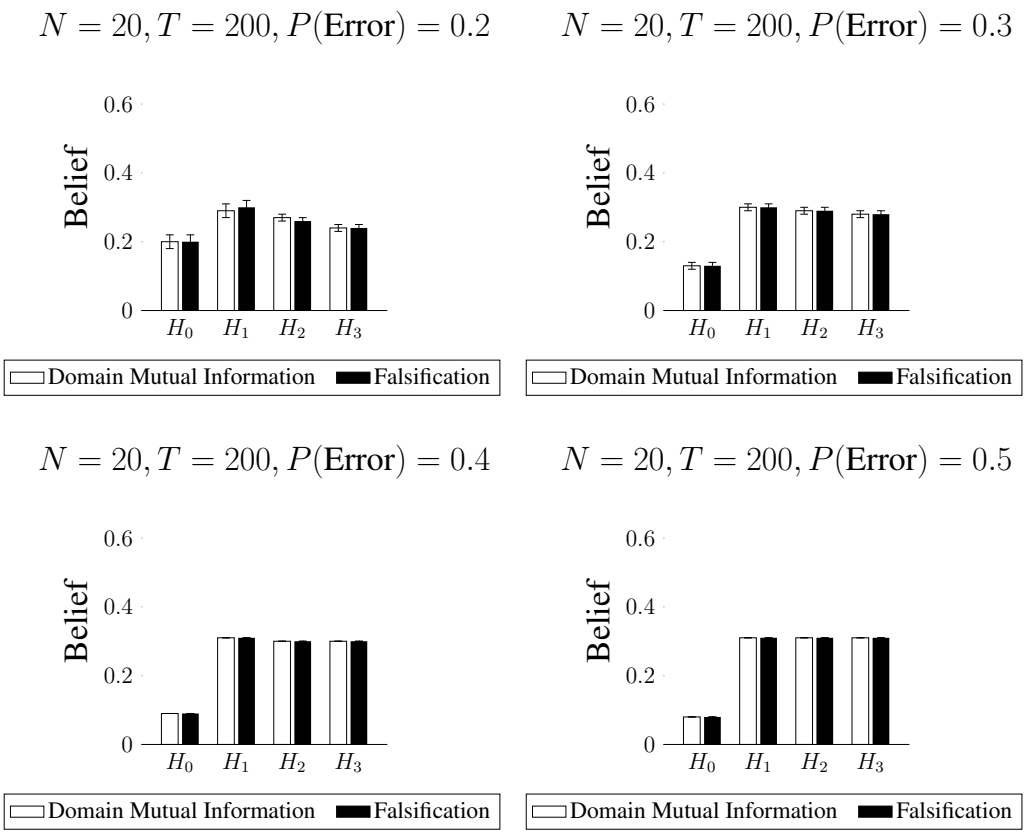


Figure E.30: All poor hypotheses. Belief distributions learned by the Domain Mutual Information and Falsification sampilng algorithms. $Budget = 200, P(\text{Error}) \in \{0.2, 0.3, 0.4, 0.5\}$. As noise increases the performance of the algorithms converges, and it becomes more difficult to distinguish the correct hypothesis.

E.2 Statistical Significance Data

In this section we present in tabular form the data which supports the plots in Chapter 5. Positive values of Cohen's d mean that the falsification algorithm has superior performance. Negative values means the mutual information algorithm is better.

E.2.1 Experiment 1 - All Good Hypotheses

Table E.1: Change in belief in the best hypotheses as sensor noise changes, all good hypotheses, sampling budget = 25.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	0.001	N/A
0.01	0.002	N/A
0.05	0.003	0.66
0.1	0.003	**0.80**
0.2	0.001	N/A
0.3	0.002	N/A
0.4	0.000	N/A
0.5	0.0	N/A

Table E.2: Change in belief in the best hypotheses as sensor noise changes, all good hypotheses, sampling budget = 50.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	0.001	N/A
0.01	0.002	N/A
0.05	0.002	N/A
0.1	0.002	N/A
0.2	0.002	N/A
0.3	0.002	N/A
0.4	0.000	N/A
0.5	0.0	N/A

Table E.3: Change in belief in the best hypotheses as sensor noise changes, all good hypotheses, sampling budget = 100.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	0.001	N/A
0.01	0.002	N/A
0.05	0.002	N/A
0.1	0.002	0.65
0.2	0.001	0.70
0.3	0.001	N/A
0.4	0.000	N/A
0.5	0.0	N/A

Table E.4: Change in belief in the best hypotheses as sensor noise changes, all good hypotheses, sampling budget = 150.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	0.001	N/A
0.01	0.002	N/A
0.05	0.001	N/A
0.1	0.002	0.63
0.2	0.001	**0.93**
0.3	0.001	0.70
0.4	0.000	N/A
0.5	0.0	N/A

Table E.5: Change in belief in the best hypotheses as sensor noise changes, all good hypotheses, sampling budget = 200.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	0.002	0.67
0.01	0.002	0.73
0.05	0.002	0.70
0.1	0.001	0.68
0.2	0.001	**1.09**
0.3	0.000	N/A
0.4	0.000	N/A
0.5	0.0	N/A

E.2.2 Experiment 2 - Mixed Quality Hypotheses

Table E.6: Change in belief in the best hypotheses as sensor noise changes, mixed quality hypotheses, sampling budget = 25.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	-0.017	**-1.03**
0.01	-0.013	**-0.82**
0.05	-0.010	-0.79
0.1	-0.002	N/A
0.2	-0.000	N/A
0.3	-0.001	**-0.87**
0.4	-0.000	N/A
0.5	0.0	N/A

Table E.7: Change in belief in the best hypotheses as sensor noise changes, mixed quality hypotheses, sampling budget = 50.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	-0.013	**-1.54**
0.01	-0.010	**-1.22**
0.05	-0.005	**-0.87**
0.1	0.001	N/A
0.2	0.001	N/A
0.3	-0.001	N/A
0.4	0.000	N/A
0.5	0.0	N/A

Table E.8: Change in belief in the best hypotheses as sensor noise changes, mixed quality hypotheses, sampling budget = 100.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	-0.006	N/A
0.01	-0.004	N/A
0.05	-0.002	N/A
0.1	0.001	N/A
0.2	0.001	N/A
0.3	0.000	N/A
0.4	0.000	N/A
0.5	0.0	N/A

Table E.9: Change in belief in the best hypotheses as sensor noise changes, mixed quality hypotheses, sampling budget = 150.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	-0.005	-0.69
0.01	-0.003	N/A
0.05	-0.002	N/A
0.1	0.001	N/A
0.2	0.001	N/A
0.3	0.000	N/A
0.4	0.000	N/A
0.5	0.0	N/A

Table E.10: Change in belief in the best hypotheses as sensor noise changes, mixed quality hypotheses, sampling budget = 200.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	-0.001	N/A
0.01	0.000	N/A
0.05	0.000	N/A
0.1	0.002	0.63
0.2	0.000	N/A
0.3	0.000	N/A
0.4	0.000	N/A
0.5	0.0	N/A

E.2.3 Experiment 3 - All Bad Hypotheses

Table E.11: Change in belief in the best hypotheses as sensor noise changes, all poor hypotheses, sampling budget = 25.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	0.131	**2.55**
0.01	0.113	**2.19**
0.05	0.050	**1.40**
0.1	0.014	N/A
0.2	-0.006	-0.79
0.3	-0.004	**-0.90**
0.4	-0.000	N/A
0.5	0.0	0.0 N/A

Table E.12: Change in belief in the best hypotheses as sensor noise changes, all poor hypotheses, sampling budget = 50.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	0.104	**2.67**
0.01	0.091	**2.43**
0.05	0.043	**1.51**
0.1	0.017	**1.14**
0.2	-0.002	N/A
0.3	-0.001	N/A
0.4	0.000	N/A
0.5	0.0	N/A

Table E.13: Change in belief in the best hypotheses as sensor noise changes, all poor hypotheses, sampling budget = 100.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	0.071	**3.01**
0.01	0.061	**2.58**
0.05	0.025	**1.33**
0.1	0.010	**1.03**
0.2	-0.001	N/A
0.3	-0.000	N/A
0.4	0.000	N/A
0.5	0.0	N/A

Table E.14: Change in belief in the best hypotheses as sensor noise changes, all poor hypotheses, sampling budget = 150.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	0.044	**2.88**
0.01	0.037	**2.77**
0.05	0.015	**1.95**
0.1	0.006	**1.09**
0.2	-0.001	N/A
0.3	-0.000	N/A
0.4	0.000	N/A
0.5	0.0	N/A

Table E.15: Change in belief in the best hypotheses as sensor noise changes, all poor hypotheses, sampling budget = 200.

$P(\text{Error})$	$\Delta P(H^*)$	Cohen's d
0.0	0.037	**4.05**
0.01	0.031	**4.19**
0.05	0.014	**2.85**
0.1	0.007	**1.91**
0.2	0.001	N/A
0.3	0.000	N/A
0.4	0.000	N/A
0.5	0.0	N/A

Bibliography

Bruce Abramson. The expected-outcome model of two-player games. 1987. 4.2

Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007. 4.2, 4.3.2

Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *COLT*, pages 39–1, 2012. 5.1

Pablo F Alcantarilla, Simon Stent, German Ros, Roberto Arroyo, and Riccardo Gherardi. Street-view change detection with deconvolutional networks. In *Robotics: Science and Systems*, 2016. 4.2

Alnur Ali, Rich Caruana, and Ashish Kapoor. Active learning with model selection. In *AAAI*, pages 1673–1679, 2014. 5.1

Daniel R Andrews, Anthony Colaprete, Jacqueline Quinn, Donald Chavers, and Martin Picard. Introducing the resource prospector (rp) mission. In *AIAA SPACE 2014 Conference and Exposition*, page 4378, 2014. 1.2.2, 3, 4.1, A.2

Akash Arora, P Michael Furlong, Robert Fitch, Terry Fong, Salah Sukkarieh, and Richard Elphic. Online multi-modal learning and adaptive informative trajectory planning for autonomous exploration. In *Field and Service Robotics*, pages 239–254. Springer, 2018. 2.3, 5.5

RE Arvidson, RG Bonitz, ML Robinson, JL Carsten, RA Volpe, A Trebi-Ollennu, MT Mellon, PC Chu, KR Davis, JJ Wilson, et al. Results from the mars phoenix lander robotic arm experiment. *Journal of Geophysical Research: Planets*, 114(E1), 2009. 2.1, 5.5

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003. 3.1.2, 4.5

Peter Auer and Ronald Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010. 3.1.2

Josep Aulinas, Yvan R Petillot, Joaquim Salvi, and Xavier Lladó. The slam problem: a survey. In *CCIA*, pages 363–371. Citeseer, 2008. 2.3

Rasmus Bååth. Bayesian first aid: A package that implements bayesian alternatives to the classical *.test functions in r. In *UseR! 2014 - the International R User Conference*, 2014. 5.3

Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 65–72. ACM, 2006. 3.1.2, 5.1

John E Bares and David S Wettergreen. Dante II: Technical description, results, and lessons

learned. *The International Journal of Robotics Research*, 18(7):621–649, 1999. 2.1

MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 39–52. Springer, 2010. 2.2.2, 3.1.1

Carl W Baum and Venugopal V Veeravalli. A sequential procedure for multihypothesis testing. *IEEE Transactions on Information Theory*, 40(6), 1994. 4.5

Jan Beirlant, Edward J Dudewicz, László Györfi, and Edward C Van der Meulen. Nonparametric entropy estimation: An overview. *International Journal of Mathematical and Statistical Sciences*, 6(1):17–39, 1997. 5.2.2

Asher Bender, Stefan B Williams, Oscar Pizarro, and Michael V Jakuba. Adaptive exploration of benthic habitats using gaussian processes. In *OCEANS 2010*, pages 1–10. IEEE, 2010. 2.3

Dean Bergman, Brian J Glass, Thomas Stucky, Kris Zacny, Gale Paulsen, and Chris McKay. Obtaining vibration data for autonomous health monitoring of interplanetary drills. In *AIAA SPACE 2016*, page 5485. 2016. 5.4

Jean-Loup Bertaux, Thomas Widemann, Alain Hauchecorne, VI Moroz, and AP Ekonomov. Vega 1 and vega 2 entry probes: An investigation of local uv absorption (220–400 nm) in the atmosphere of venus (so2 aerosols, cloud structure). *Journal of Geophysical Research: Planets*, 101(E5):12709–12745, 1996. 2.1

Jeffrey J Biesiadecki, P Chris Leger, and Mark W Maimone. Tradeoffs between directed and autonomous driving on the mars exploration rovers. *The International Journal of Robotics Research*, 26(1):91–104, 2007. A.3

Brian Bingham, Brendan Foley, Hanumant Singh, Richard Camilli, Katerina Delaporta, Ryan Eustice, Angelos Mallios, David Mindell, Christopher Roman, and Dimitris Sakellariou. Robotic tools for deep water archaeology: Surveying an ancient shipwreck with an autonomous underwater vehicle. *Journal of Field Robotics*, 27(6):702–717, 2010. 2.1

Space Studies Board, National Research Council, et al. *Vision and voyages for planetary science in the decade 2013-2022*. National Academies Press, 2012. 5

Rodney A Brooks. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159, 1991. 1

F Thomas Bruss et al. Sum the odds to one and stop. *The Annals of Probability*, 28(3):1384–1391, 2000. 3.1.1

Apostolos N Burnetas and Michael N Katehakis. Optimal adaptive policies for markov decision processes. *Mathematics of Operations Research*, 22(1):222–255, 1997. 3.1.2

Richard Camilli, Brian Bingham, Michael Jakuba, Hanumant Singh, and Jean Whelan. Integrating in-situ chemical sampling with auv control systems. In *OCEANS'04. MTTS/IEEE TECHNO-OCEAN'04*, volume 1, pages 101–109. IEEE, 2004. 2.2.2

Alberto Candela, David Thompson, Eldar Noe Dobrea, and David Wettergreen. Planetary robotic exploration driven by science hypotheses for geologic mapping. In *Proc. of IEEE/RSJ IROS*, 2017. 2.3

Rudolf Carnap. Testability and meaning. *Philosophy of science*, 3(4):419–471, 1936. 5

Rebecca Castano, Tara Estlin, Robert C Anderson, Daniel M Gaines, Andres Castano, Benjamin Bornstein, Caroline Chouinard, and Michele Judd. Oasis: Onboard autonomous science investigation system for opportunistic rover science. *Journal of Field Robotics*, 24(5):379–397, 2007. 2.2.1, 3.1, 3.1.4, 5.1

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011. 5.1

Eric L Charnov. Optimal foraging, the marginal value theorem. *Theoretical population biology*, 9(2):129–136, 1976. 3.1.3

Benjamin Charrow, Sikang Liu, Vijay Kumar, and Nathan Michael. Information-theoretic mapping using cauchy-schwarz quadratic mutual information. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4791–4798. IEEE, 2015. 2.3, 3.1.4

Steve Chien, Rob Sherwood, Daniel Tran, Rebecca Castano, Benjamin Cichy, Ashley Davies, Gregg Rabideau, Nghia Tang, Michael Burl, Dan Mandl, et al. Autonomous science on the eo-1 mission. 2003. 2.2.1

Steve Chien, Rob Sherwood, Daniel Tran, Benjamin Cichy, Gregg Rabideau, Rebecca Castano, Ashley Davis, Dan Mandl, Bruce Trout, Seth Shulman, et al. Using autonomy flight software to improve science return on earth observing one. *Journal of Aerospace Computing, Information, and Communication*, 2(4):196–216, 2005. 2.2.1, 5.1

Steve A Chien, Russell Knight, Andre Stechert, Rob Sherwood, and Gregg Rabideau. Using iterative repair to improve the responsiveness of planning and scheduling. In *AIPS*, pages 300–307, 2000. 2.2.1

Sanjiban Choudhury, Ashish Kapoor, Gireeja Ranade, Sebastian Scherer, and Debadeepta Dey. Adaptive information gathering via imitation learning. *arXiv preprint arXiv:1705.07834*, 2017. 2.3

Jacob Cohen. *Statistical power analysis for the behavioral sciences*. Academic press, 2013. 3.4

P. Coyle. Probabilistic Programming and PyMC3. *ArXiv e-prints*, July 2016. 4.3.2

Nichael Lynn Cramer. A representation for the adaptive generation of simple sequential programs. In *Proceedings of the First International Conference on Genetic Algorithms*, pages 183–187, 1985. 2.4

Carroll Croarkin, Paul Tobias, Chelli Zey, et al. *Engineering statistics handbook*. NIST iTL, 2002. 2.3

Jnaneshwar Das, Julio Harvey, Frédéric Py, Harshvardhan Vathsangam, Rishi Graham, Kanna Rajan, and Gaurav S Sukhatme. Hierarchical probabilistic regression for auv-based adaptive sampling of marine phenomena. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5571–5578. IEEE, 2013. 2.2.2, 2.3

Jnaneshwar Das, Frédéric Py, Julio BJ Harvey, John P Ryan, Alyssa Gellene, Rishi Graham, David A Caron, Kanna Rajan, and Gaurav S Sukhatme. Data-driven robotic sampling for marine ecosystem monitoring. *The International Journal of Robotics Research*, 34(12):1435–1452, 2015. 2.2.1, 2.2.2, 3.1.4, 4.2, 4.5

Lorraine Daston and Peter Galison. *Objectivity*. Zone Books, 2007. 1.1

Anca D Dragan Dorsa Sadigh, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems (RSS)*, 2017. 5.1

David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning*, June 2013. 2.4

Tara A Estlin, Benjamin J Bornstein, Daniel M Gaines, Robert C Anderson, David R Thompson, Michael Burl, Rebecca Castaño, and Michele Judd. Aegis automated science targeting for the mer opportunity rover. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3): 50, 2012. 2.2.1, 2.2.1, 3, 3.1.4, 5.1

Linda Neuman Ezell. Nasa historical data book. volume 2: Programs and projects 1958-1968. 1988. 2.1

Thomas S Ferguson. Who solved the secretary problem? *Statistical science*, pages 282–289, 1989. 2.2, 3.1.1, 4.2

Gabriele Ferri, Michael V Jakuba, and Dana R Yoerger. A novel trigger-based method for hydrothermal vents prospecting using an autonomous underwater robot. *Autonomous Robots*, 29 (1):67–83, 2010. (document), 2.2.2, 4, 4.2, 4.2, 4.1, 4.2, 4.2, 4.3.1, 4.3.1, 4.3.1, 4.3.1, 4.3.3, 4.3.4, 4.4.2, 4.4.4, 4.4.4

Greydon Taylor Foil. Efficiently sampling from underlying physical models. 2016. 3.1.4

Andrew Frank and Arthur Asuncion. Uci machine learning repository [http://archive. ics. uci. edu/ml]. irvine, ca: University of california. *School of Information and Computer Science*, 213, 2010. 2.2.1

Luigi Freda, Giuseppe Oriolo, and Francesco Vecchioli. An exploration method for general robotic systems equipped with multiple sensors. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5076–5082. IEEE, 2009. 2.3

Paul Furgale, Timothy D Barfoot, Nadeem Ghafoor, Kevin Williams, and Gordon Osinski. Field testing of an integrated surface/subsurface modeling technique for planetary exploration. *The International Journal of Robotics Research*, 29(12):1529–1549, 2010. 2.1

P Michael Furlong. Adaptive sample selection for hypothesis falsification. 2017. 5.1

P Michael Furlong and David Wettergreen. Sequential allocation of sampling budgets in unknown environments. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014a. 3.1.4

P Michael Furlong and David S Wettergreen. Budgeting samples for exploration in unknown environments. *i-SAIRAS*, 2014b. 3.1.4, 3.2.1

Daniel M Gaines, Tara Estlin, Steve Schaffer, Rebecca Castano, and Alberto Elfes. Robust and opportunistic autonomous science for a potential titan aerobot. 2010. 2.2.1

Martha S Gilmore, Rebecca Castaño, Tobias Mann, Robert C Anderson, Eric D Mjolsness, Roberto Manduchi, and R Stephen Saunders. Strategies for autonomous rovers at mars. *Jour-*

nal of Geophysical Research: Planets (1991–2012), 105(E12):29223–29237, 2000. 2.2.1

Yogesh Girdhar and Gregory Dudek. Modeling curiosity in a mobile robot for long-term autonomous exploration and monitoring. *Autonomous Robots*, 40(7):1267–1278, 2016. 2.2.2, 4.2

Yogesh Girdhar, Philippe Giguere, and Gregory Dudek. Autonomous adaptive underwater exploration using online topic modelling. In *International Symposium on Experimental Robotics (ISER)*, 2012. 2.2.2

Yogesh Girdhar, Philippe Giguère, and Gregory Dudek. Autonomous adaptive exploration using realtime online spatiotemporal topic modeling. *The International Journal of Robotics Research*, page 0278364913507325, 2013a. 2.2.2

Yogesh Girdhar, Philippe Giguere, and Gregory Dudek. Autonomous adaptive underwater exploration using online topic modeling. In *Experimental Robotics*, pages 789–802. Springer, 2013b. 3.1.4

Yogesh Girdhar, David Whitney, and Gregory Dudek. Curiosity based exploration for learning terrain models. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 578–584. IEEE, 2014. 2.3, 5.1

John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011. 5.1

Christian Gourieroux and Alain Monfort. *Statistics and econometric models*, volume 1. Cambridge University Press, 1995. 2.3

R.B. Grosse, R. Salakhutdinov, W.T. Freeman, and J.B. Tenenbaum. Exploiting compositionality to explore a large space of model structures. In *Uncertainty in Artificial Intelligence*, 2012. 2.4

Ian Hacking. *Representing and intervening: Introductory topics in the philosophy of natural science*. Cambridge University Press, 1983. 1, 5

Eric Hand. Philae probe makes bumpy touchdown on a comet. *Science*, 346(6212):900–901, 2014. 2.1

MH Hecht, SP Kounaves, RC Quinn, SJ West, SMM Young, DW Ming, DC Catling, BC Clark, WV Boynton, J Hoffman, et al. Detection of perchlorate and the soluble chemistry of martian soil at the phoenix lander site. *Science*, 325(5936):64–67, 2009. 3

Hank Heidt, Jordi Puig-Suari, Augustus Moore, Shinichi Nakasuka, and Robert Twiggs. Cubesat: A new generation of picosatellite for education and industry low-cost space experimentation. 2000. A.1

JL Heldmann, A Colaprete, A Cook, T Roush, M Deans, R Elphic, D Lim, JR Skok, NE Button, S Karunatillake, et al. Mojave volatiles prospector (mvp): Science and operations results from a lunar polar rover analog field campaign. In *Lunar and Planetary Science Conference*, volume 46, page 2165, 2015. 4.1

Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based robotic information gathering algorithms. *Int. J. Robot. Res.*, 33(9):1271–1287, 2014. 2.3

Geoffrey A Hollinger, Brendan Englot, Franz S Hover, Urbashi Mitra, and Gaurav S Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *Int. J. Robot. Res.*, 32 (1):3–18, 2013. 2.3

Guang-Bin Huang, Dian Hui Wang, and Yuan Lan. Extreme learning machines: a survey. *International journal of machine learning and cybernetics*, 2(2):107–122, 2011. 6.3

M Jakuba and D Yoerger. Autonomous search for hydrothermal vent fields with occupancy grid maps. In *Proc. of ACRA*, volume 8, page 2008, 2008. 2.1

Tackseung Jun. A survey on the bandit problem with switching costs. *De Economist*, 152(4): 513–541, 2004. 3.1.2

Pentii Kanerva, Jan Kristoferson, and Anders Holst. Random indexing of text samples for latent semantic analysis. In *Proceedings of the Cognitive Science Society*, volume 1, 2000. 6.3

Ross D King, Kenneth E Whelan, Ffion M Jones, Philip GK Reiser, Christopher H Bryant, Stephen H Muggleton, Douglas B Kell, and Stephen G Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, 2004. 2.4, 3.1

Ross D King, Jem Rowland, Wayne Aubrey, Maria Liakata, Magdalena Markham, Larisa N Soldatova, Ken E Whelan, Amanda Clare, Mike Young, Andrew Sparkes, et al. The robot scientist adam. *Computer*, 42(8), 2009a. 2.4

Ross D King, Jem Rowland, Stephen G Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N Soldatova, et al. The automation of science. *Science*, 324(5923):85–89, 2009b. 2.4

S Knight, Gregg Rabideau, Steve Chien, Barbara Engelhardt, and Rob Sherwood. Casper: Space exploration through continuous planning. *IEEE Intelligent Systems*, 16(5):70–75, 2001. 6.2

Nils Kolling, Timothy EJ Behrens, Rogier B Mars, and Matthew FS Rushworth. Neural mechanisms of foraging. *Science*, 336(6077):95–98, 2012. 3.1.3

SM Krimigis, TP Armstrong, WI Axford, CO Bostrom, CY Fan, G Gloeckler, and LJ Lanzerotti. The low energy charged particle (lecp) experiment on the voyager spacecraft. *Space Science Reviews*, 21(3):329–354, 1977. 2.1

SM Krimigis, JF Carbary, EP Keath, TP Armstrong, LJ Lanzerotti, and G Gloeckler. General characteristics of hot plasma and energetic particles in the saturnian magnetosphere: Results from the voyager spacecraft. *Journal of Geophysical Research: Space Physics*, 88(A11): 8871–8892, 1983. 2, 2.1

Anurag Kumar and Bhiksha Raj. Classifier risk estimation under limited labeling resources. *arXiv preprint arXiv:1607.02665*, 2016. 5.1

Clayton Kunz, Chris Murphy, Richard Camilli, Hanumant Singh, John Bailey, Ryan Eustice, Michael Jakuba, Ko-ichi Nakamura, Chris Roman, Taichi Sato, et al. Deep sea underwater robotic exploration in the ice-covered arctic ocean with auvs. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3654–3660. IEEE, 2008. 2.1

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances*

in applied mathematics, 6(1):4–22, 1985. 2.3, 3.1.2, 5.1

Pat Langley, Herbert A Simon, Gary L Bradshaw, and Jan M Zytkow. Scientific discovery: computational explorations of the creative process. 1987. 2.4

Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006. 2.1

Michael Lee, Matthew Hanczor, Jiyang Chu, Zhong He, Nathan Michael, and Red Whittaker. 3-d volumetric gamma-ray imaging and source localization with a mobile robot. *arXiv preprint arXiv:1802.06072*, 2018. 4.2

Leonid A Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3): 115–116, 1973. 2.4, 2.4

Leonid A Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984. 2.4

James Lind. *A Treatise on the Scurvy: In Three Parts, Containing an Inquiry Into the Nature, Causes, and Cure, of that Disease*. A. Millar, 1757. 1

Dennis V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, pages 986–1005, 1956. 1, 2.3, 3.2.1, 5.1

Mark W Lipsey, Kelly Puzio, Cathy Yun, Michael A Hebert, Kasia Steinka-Fry, Mikel W Cole, Megan Roberts, Karen S Anthony, and Matthew D Busick. Translating the statistical representation of the effects of education interventions into more readily interpretable forms. *National Center for Special Education Research*, 2012. 5.2.4, 5.3

James Robert Lloyd, David Duvenaud, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Automatic construction and Natural-Language description of nonparametric regression models. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2014. 2.4

Ralph D Lorenz, Tetsuya Tokano, and Claire E Newman. Winds and tides of ligeia mare, with application to the drift of the proposed time time (titan mare explorer) capsule. *Planetary and Space Science*, 60(1):72–85, 2012. A.2

Daniel L Ly and Hod Lipson. Learning symbolic representations of hybrid dynamical systems. *Journal of Machine Learning Research*, 13(Dec):3585–3618, 2012. 2.4, 2.4

Omid Madani, Daniel J Lizotte, and Russell Greiner. Active model selection. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 357–365. AUAI Press, 2004. 5.1

Mark Maimone, Andrew Johnson, Yang Cheng, Reg Willson, and Larry Matthies. Autonomous navigation results from the mars exploration rover (mer) mission. In *Experimental robotics IX*, pages 3–13. Springer, 2006. 1.3, A.3

Roman Marchant, Fabio Ramos, Scott Sanner, et al. Sequential bayesian optimisation for spatial-temporal monitoring. In *UAI*, pages 553–562, 2014. 4.2

George Mathew and Igor Mezić. Metrics for ergodicity and design of ergodic dynamics for multi-agent systems. *Physica D: Nonlinear Phenomena*, 240(4-5):432–442, 2011. 4.2, D.6.1, D.6.1, D.3, D.6.1, D.6.3

Lauren M Miller and Todd D Murphey. Trajectory optimization for continuous ergodic exploration on the motion group se (2). In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 4517–4522. IEEE, 2013a. 4.2, D.6.1

Lauren M Miller and Todd D Murphey. Trajectory optimization for continuous ergodic exploration. In *American Control Conference (ACC), 2013*, pages 4196–4201. IEEE, 2013b. 4.2, D.6.1, D.6.1

Lauren M Miller, Yonatan Silverman, Malcolm A MacIver, and Todd D Murphey. Ergodic exploration of distributed information. *IEEE Transactions on Robotics*, 32(1):36–52, 2016. 2.3, 3.1.4, 4.2, 4.2, 5.1, D.6.1, D.6.1, D.6.3

Philippe Morere, Roman Marchant, and Fabio Ramos. Sequential bayesian optimization as a pomdp for environment monitoring with uavs. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 6381–6388. IEEE, 2017. 4.2

Mark EJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46 (5):323–351, 2005. 3.3.2

Pedro A Ortega and Daniel A Braun. A minimum relative entropy principle for learning and acting. *Journal of Artificial Intelligence Research*, pages 475–511, 2010. 3.1.2, 5.1

Gerhard Paar, Mark Woods, Christiane Gimkiewicz, Frédéric Labrosse, Alberto Medina, Laurence Tyler, David P Barnes, Gerald Fritz, and Konstantinos Kapellos. Proviscout: a planetary scouting rover demonstrator. *Proc. SPIE Vol. 8301–IS&T/SPIE Electronic Imaging 2012: Intelligent Robots & Computer Vision*, 2012. 5.1

Gerhard Paar, Laurence Tyler, Dave Barnes, Mark Woods, Andy Shaw, Konstantinos Kapellos, Tomas Pajdla, Alberto Medina, Derek Pullan, Andrew Griffiths, et al. The proviscout field trials tenerife 2012–integrated testing of aerobot mapping, rover navigation and science assessment. In *Proc. 12th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2013)*, 2013. 2.2.1, 3.1.4

G Paulsen, S Yoon, K Zacny, D Wettergreeng, and NA Cabrol. The lita drill and sample delivery system. In *AGU Fall Meeting Abstracts*, 2013. 3

Charles Sanders Peirce and Joseph Jastrow. On small differences in sensation. 1884. 1

Peter Pirolli and Stuart Card. Information foraging. *Psychological review*, 106(4):643, 1999. 3.1.3

Karl Popper. *The logic of scientific discovery*. Routledge, 2005. 1, 5

Kevin S Pratt and Robin R Murphy. Protection from human error: Guarded motion methodologies for mobile robots. *IEEE Robotics & Automation Magazine*, 19(4):36–47, 2012. A.3

Ernst L'vovich Presman and Isaac Mikhailovich Sonin. The best choice problem for a random number of objects. *Teoriya Veroyatnostei i ee Primeneniya*, 17(4):695–706, 1972. 3.1.1

Jose C Principe. *Information theoretic learning: Renyi's entropy and kernel perspectives*. Springer Science & Business Media, 2010. 2.3, 6.3

Da Qi, Ross D King, Andrew L Hopkins, G Richard J Bickerton, and Larisa N Soldatova. An ontology for description of drug discovery investigations. *Journal of Integrative Bioinformatics*

(JIB), 7(3):156–168, 2010. 2.4

Paul Raeburn. *Uncovering the secrets of the Red Planet: MARS*. National Geographic, 1998. 2.1

Frank P Ramsey. Truth and probability (1926). *The foundations of mathematics and other logical essays*, pages 156–198, 1931. 5

Hadi Ravanbakhsh and Sriram Sankaranarayanan. Learning lyapunov (potential) functions from counterexamples and demonstrations. *arXiv preprint arXiv:1705.09619*, 2017. 5.1

Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952. 2.2, 3.1.2, 5.1

Amy L Rocha. The infinite secretary problem with recall. *The Annals of Probability*, pages 898–916, 1993. 4.2

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011. 2.2.2

RZ Sagdeev and VI Moroz. Project vega first stage-missions to venus. *Pisma v Astronomicheskii Zhurnal*, 12:5–9, 1986. 2.1

Christoph Sawade, Niels Landwehr, Steffen Bickel, and Tobias Scheffer. Active risk estimation. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 951–958. Citeseer, 2010. 5.1, 5.1, 5.2, 5.2.2

Jeffrey D Scargle. Studies in astronomical time series analysis. v. bayesian blocks, a new method to analyze structure in photon counting data. *The Astrophysical Journal*, 504(1):405, 1998. 4.2, 4.2

Jeffrey D Scargle, Jay P Norris, Brad Jackson, and James Chiang. Studies in astronomical time series analysis. vi. bayesian block representations. *The Astrophysical Journal*, 764(2):167, 2013. 4.2

Jürgen Schmidhuber. Discovering solutions with low kolmogorov complexity and high generalization capability. In *Machine Learning: Proceedings of the Twelfth International Conference*, pages 188–196. Citeseer, 1995. 2.4, 2.4

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009. 2.4

Mac Schwager, Philip Dames, Daniela Rus, and Vijay Kumar. A multi-robot control policy for information gathering in the presence of unknown hazards. In *Robotics Research*, pages 455–472. Springer, 2017. 2.3

Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2): 461–464, 1978. 2.4

Julie A Shah, Joseph H Saleh, and Jeffrey A Hoffman. Analytical basis for evaluating the effect of unplanned interventions on the effectiveness of a human–robot system. *Reliability Engineering & System Safety*, 93(8):1280–1286, 2008. A.1

Andy Shaw, Mark Woods, Ehsan Honary, Philip Rendell, Derek Pullan, Dave Barnes, Steve

Pugh, and Derek Long. Crest robotic scientist. *Towards Autonomous Robotic Systems (TAROS)*, 2007. 2.2.1

Kirstine Smith. On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of observations. *Biometrika*, pages 1–85, 1918. 1.1, 2.3, 3.1, 5, 5.1

Trey Smith. *Probabilistic Planning for Robotic Exploration*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2007. 2.3

Andrew Sparkes, Wayne Aubrey, Emma Byrne, Amanda Clare, Muhammed N Khan, Maria Liakata, Magdalena Markham, Jem Rowland, Larisa N Soldatova, Kenneth E Whelan, et al. Towards robot scientists for autonomous scientific discovery. *Automated Experimentation*, 2 (1):1, 2010. 2.4

Shannon V Spires and Steven Y Goldsmith. Exhaustive geographic search with mobile robots along space-filling curves. In *Collective robotics*, pages 1–12. Springer, 1998. 2.1

Mark Steyvers and Scott Brown. Prediction and change detection. In *Advances in neural information processing systems*, pages 1281–1288, 2006. 4.2, 4.2

Ellen Stofan, Ralph Lorenz, Jonathan Lunine, Edward B Bierhaus, Ben Clark, Paul R Mahaffy, and Mike Ravine. Time-the titan mare explorer. In *Aerospace Conference, 2013 IEEE*, pages 1–10. IEEE, 2013. 2.1, A.2

Yi Sun, Faustino Gomez, and Jürgen Schmidhuber. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *Artificial General Intelligence*, pages 41–51. Springer, 2011. 3.1.4

Yu A Surkov, LP Moskalyova, VP Kharyukova, AD Dudin, GG Smirnov, and S Ye Zaitseva. Venus rock composition at the vega 2 landing site. *Journal of Geophysical Research: Solid Earth*, 91(B13), 1986. 2.1, A.2

Wennie Tabib, Red Whittaker, and Nathan Michael. Efficient multi-sensor exploration using dependent observations and conditional mutual information. In *Proc. of IEEE SSRR*, pages 42–47, 2016. 2.3, 3.1.4, 5.5

Tong Tao, Yalou Huang, Fengchi Sun, and Tingting Wang. Motion planning for slam based on frontier exploration. In *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, pages 2120–2125. IEEE, 2007. 2.3

David R Thompson. *Intelligent Mapping for Autonomous Robotic Survey*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2008. 2, 2.3, 3.1.4, 5.1, 5.1, 5.2

David R Thompson, Trey Smith, and David Wettergreen. Information-optimal selective data return for autonomous rover traverse science and survey. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 968–973. IEEE, 2008. 4.2

David R Thompson, David S Wettergreen, and Francisco J Calderóon Peralta. Autonomous science during large-scale robotic survey. *J. Field Robot.*, 28(4):542–564, 2011. 2.3

David R Thompson, Nathalie A Cabrol, Michael Furlong, Craig Hardgrove, Bryan Kian Hsiang Low, Jeffrey Moersch, and David Wettergreen. Adaptive sensing of time series with application to remote exploration. In *Robotics and Automation (ICRA), 2013 IEEE International*

Conference on, pages 3463–3468. IEEE, 2013. 2.2.2, 3.1.4, 4.2, 4.5

David R Thompson, David T Flannery, Ravi Lanka, Abigail C Allwood, Brian D Bue, Benton C Clark, W Timothy Elam, Tara A Estlin, Robert P Hodyss, Joel A Hurowitz, et al. Automating x-ray fluorescence analysis for rapid astrobiology surveys. *Astrobiology*, 15(11):961–976, 2015a. 3.1.4

David Ray Thompson, David Wettergreen, Greydon T Foil, P Michael Furlong, and Anatha Ravi Kiran. Spatio-spectral exploration combining in situ and remote measurements. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3679–3685, 2015b. 2.3, 5.1

DR Thompson, AC Allwood, DL Bekker, NA Cabrol, T Fuchs, and KL Wagstaff. Texturecam: Autonomous image analysis for astrobiology survey. In *Lunar and Planetary Institute Science Conference Abstracts*, volume 43, page 1659, 2012. 2.2.1

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933. 3.1.2, 5.1

Fabio Lorenzo Traversa, Chiara Ramella, Fabrizio Bonani, and Massimiliano Di Ventra. Memcomputing np-complete problems in polynomial time using polynomial resources and collective states. *Science advances*, 1(6):e1500031, 2015. 6.3

VM Vakhnin. A review of the venera 4 flight and its scientific program. *Journal of the Atmospheric Sciences*, 25(4):533–534, 1968. 2.1

R. J. Vanderbei. The optimal choice of a subset of a population. *Mathematics of Operations Research*, 5(4):481–486, 1980. doi: 10.1287/moor.5.4.481. URL `http://dx.doi.org/10.1287/moor.5.4.481`. 3.1.1

John Vickers. The problem of induction. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2016 edition, 2016. 2.4, 5

Arnoud Visser, Merlijn Van Ittersum, Luis A González Jaime, Laurenţiu A Stancu, et al. Beyond frontier exploration. In *Robot Soccer World Cup*, pages 113–123. Springer, 2007. 2.3

Michael D Wagner, Dimitrios Apostolopoulos, Kimberly Shillcutt, Benjamin Shamah, Reid Simmons, and William Red Whittaker. The science autonomy system of the nomad robot. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1742–1749. IEEE, 2001. 2.2.1, 3.1, 3.1.4

Kiri L Wagstaff, Nina L Lanza, David R Thompson, Thomas G Dietterich, and Martha S Gilmore. Guiding scientific discovery with explanations using demud. In *AAAI*, 2013. 2.2.1, 6.3

Abraham Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945. 3.2.1, 4, 4.2, 4.3.1

Kimberley A Warren-Rhodes, Kevin L Rhodes, Stephen B Pointing, Stephanie A Ewing, Donnabella C Lacap, Benito Gómez-Silva, Ronald Amundson, E Imre Friedmann, and Christopher P McKay. Hypolithic cyanobacteria, dry limit of photosynthesis, and microbial ecology in the hyperarid atacama desert. *Microbial Ecology*, 52(3):389–398, 2006. 3

Kimberley A Warren-Rhodes, Jennifer L Dungan, Jennifer Piatek, Kristin Stubbs, Benito

Gómez-Silva, Yong Chen, and Christopher P McKay. Ecology and spatial pattern of cyanobacterial community island patches in the atacama desert, chile. *Journal of Geophysical Research: Biogeosciences (2005–2012)*, 112(G4), 2007. 3

David Wettergreen, Nathalie Cabrol, James Teza, Paul Tompkins, Chris Urmson, Vandi Verma, Michael Wagner, and William Whittaker. First experiments in the robotic investigation of life in the atacama desert of chile. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 873–878. IEEE, 2005. 3

MA Wiering and Jürgen Schmidhuber. Solving pomdps with levin search and eira. 1996. 2.4

Troy Wilson and Stefan B Williams. Adaptive path planning for depth-constrained bathymetric mapping with an autonomous surface vessel. *Journal of Field Robotics*, 2017. 2.1, 4.2, 4.2, 6.3

Mark Woods, Andy Shaw, Phil Rendell, Ehsan Honary, Dave Barnes, Steve Pugh, Dave Price, Derek Pullan, and Derek Long. Crest autonomous robotic scientist: Developing a closed-loop science exploration capability for european mars missions. In *i-SAIRAS: International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2008. 2.2.1

Mark Woods, Andy Shaw, Dave Barnes, Dave Price, Derek Long, and Derek Pullan. Autonomous science for an exomars rover–like mission. *Journal of Field Robotics*, 26(4):358–390, 2009. 2.2.1

Mark Woods, Andy Shaw, and Philip Rendell. High-level autonomy and image prioritisation for long distance mars rovers. In *Proc. 11th Symposium on Advanced Space Technologies in Robotics and Automation, ESA/ESTEC, The Netherlands*, pages 12–14, 2011. 2.2.1

Hauke Worpel and Axel D Schwope. Background subtraction and transient timing with bayesian blocks. *Astronomy & Astrophysics*, 578:A80, 2015. 4.2

Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151. IEEE, 1997. 2.3

Dana R Yoerger, Albert M Bradley, Barrie B Walden, Marie-Helene Cormier, and Wfiam BF Ryan. Fine-scale seafloor survey in rugged deep-ocean terrain with an autonomous robot. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1787–1792. IEEE, 2000. 2.1, 4.2

Dana R Yoerger, Michael Jakuba, Albert M Bradley, and Brian Bingham. Techniques for deep sea near bottom survey using an autonomous underwater vehicle. *The International Journal of Robotics Research*, 26(1):41–54, 2007. 2.2.2, 4.2

A Steven Younger, Emmett Redd, and Hava Siegelmann. Development of physical super-turing analog hardware. In *International Conference on Unconventional Computation and Natural Computation*, pages 379–391. Springer, 2014. 6.3